N° d'ordre : 3818

THÈSE

PRESÉNTÉE À

L'UNIVERSITÉ DE BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Imen JEMILI**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Clusterisation et conservation d'énergie dans les réseaux ad hoc hybrides à grande échelle

---

Soutenue le : 13 Juillet 2009

Après avis des rapporteurs :

Béchir El Ayeb	Professeur
David Simplot-Ryl	Professeur

Devant la commission d'examen composée de :

Richard Castanet	Professeur	Co-Directeur de thèse
Mohamed Mosbah	Professeur	Co-Directeur de thèse
Abdelfettah Belghith	Professeur	Co-Directeur de thèse
Francine Krief	Professeur	Présidente
Béchir El Ayeb	Professeur	Rapporteur
David Simplot-Ryl	Professeur	Rapporteur

-2009 -

Résumé :

Dans le cadre des réseaux ad hoc à grande envergure, le concept de clusterisation peut être mis à profit afin de faire face aux problèmes de passage à l'échelle et d'accroître les performances du système. Tout d'abord, cette thèse présente notre algorithme de clusterisation TBCA 'Tiered based Clustering algorithm', ayant pour objectif d'organiser le processus de clusterisation en couches et de réduire au maximum le trafic de contrôle associé à la phase d'établissement et de maintenance de l'infrastructure virtuelle générée. La formation et la maintenance d'une infrastructure virtuelle ne sont pas une fin en soi. Dans cet axe, on a exploité les apports de notre mécanisme de clusterisation conjointement avec le mode veille, à travers la proposition de l'approche de conservation d'énergie baptisée CPPCM 'Cluster based Prioritized Power Conservation Mechanism' avec deux variantes. Notre objectif principal est de réduire la consommation d'énergie tout en assurant l'acheminement des paquets de données sans endurer des temps d'attente importants aux niveaux des files d'attente des nœuds impliqués dans le transfert. Nous avons proposé aussi un algorithme de routage LCR 'Layered Cluster based Routing' se basant sur l'existence d'une infrastructure virtuelle. L'exploitation des apports de notre mécanisme TBCA et la limitation des tâches de routage additionnelles à un sous ensemble de nœuds sont des atouts pour assurer le passage à l'échelle de notre algorithme.

Mots clés : Réseaux ad hoc, clusterisation, ensemble dominant connexe, conservation d'énergie, routage.

Abstract :

Relying on a virtual infrastructure seems a promising approach to overcome the scalability problem in large scale ad hoc networks. First, we propose a clustering mechanism, TBCA 'Tiered based Clustering algorithm', operating in a layered manner and exploiting the eventual collision to accelerate the clustering process. Our mechanism does not necessitate any type of neighbourhood knowledge, trying to alleviate the network from some control messages exchanged during the clustering and maintenance process.

Since the energy consumption is still a critical issue, we combining a clustering technique and the power saving mode in order to conserve energy without affecting network performance. The main contribution of our power saving approach lies on the differentiation among packets based on the amount of network resources they have been so far consumed. Besides, the proposed structure of the beacon interval can be adjusted dynamically and locally by each node according to its own specific requirements.

We propose also a routing algorithm, LCR 'Layered Cluster based Routing'. The basic idea consists on assigning additional tasks to a limited set of dominating nodes, satisfying specific requirements while exploiting the benefits of our clustering algorithm TBCA.

Key words : Ad hoc networks, clustering, connected dominating set, power saving, routing.

A la mémoire de mon père,
A ma mère à qui je dois tout mon bonheur
pour son soutien sans faille et sa confiance,
A mes sœurs Alia et Asma,
A mes frères Sami, Salah et Naceur,
A mes anges Cyrine et Nadine,
A tous ceux qui me sont chers,
Qu'ils trouvent dans ce mémoire la preuve de mon affection et attachement.
Imen

Remerciements

Je souhaite remercier Francine Krief, Professeur à l'Ecole Nationale Supérieure d'Electricité, Informatique, Télécommunications, Mathématique et Mécanique de Bordeaux, pour m'avoir fait l'honneur d'accepter la position de Président du jury.

Je remercie également Béchir El Ayeb, Professeur à l'Université de Monastir, et David Simplot-Ryl, Professeur à l'Université de Lille 1, pour avoir accepté la charge de rapporteur et pour leurs remarques pertinentes concernant mon mémoire.

Mon parcours durant cette thèse n'aurait jamais pu aboutir sans l'intervention de plusieurs personnes, auxquelles j'adresse ma gratitude la plus sincère. Tout d'abord, j'exprime mes profonds remerciements à mes directeurs de thèse, au sein du pôle HANA et au sein du LaBRI. Mes premiers remerciements vont au Professeur Abdelfettah Belghith pour son soutien et sa vision pragmatique des problèmes. En de nombreuses occasions, son recul et sa vision d'ensemble du monde des réseaux se sont avérés essentiels pour trouver de nouvelles solutions et dégager des perspectives de recherche. C'est grâce au Professeur Mohamed Mosbah que je me suis intéressée à l'algorithmique distribuée. Outre ses remarques et suggestions techniques, académiques et professionnelles, qui m'ont été inestimables, son suivi personnel m'a beaucoup touché. Je tiens aussi à remercier le Professeur Richard Castanet, mon directeur de thèse, pour m'avoir accordé l'occasion de réaliser cette thèse et d'avoir soutenu mon travail tout au long de ces années.

Merci également à tous ceux qui m'ont accompagné et soutenu dans cette grande expérience de la vie que constitue une thèse. Mes remerciements vont aussi à tous mes amis du pôle HANA, Hanen, Naouel, Achraf, Wafa et Rafaa, qui ont contribué de prés ou de loin dans ce travail, par leur amitié et l'entraide qui régnait dans notre groupe. Je tiens à saluer mes meilleures amies pour leur encouragement, en particulier Hanen. Je ne peux oublier mes amis d'outre-mer, je m'adresse particulièrement à Hejer pour son soutien et sa présence et toutes les personnes que j'ai connues durant mes séjours au LaBRI. Je n'oublie pas le personnel administratif du LaBRI que j'ai du solliciter à maintes reprises et aussi au sein de l'ENSI.

Je ne peux oublier finalement de saluer l'encouragement de ma famille, tantes et cousines qui n'ont cessé de croire en moi. J’adresse enfin mes remerciements à tous ceux qui, par leur courtoise et sympathie, m’ont aidé, encouragé et soutenu.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

S'offrir un accès ubiquitaire à l'information, sans se soucier des contraintes temps et lieu, est devenu possible grâce à l'inexorable évolution technologique ayant permis l'émergence d'un nouvel environnement sans fil ad hoc. La volonté d'assouvir les attentes d'utilisateurs, de plus en plus nomade et toujours plus avides de services, a contribué à proliférer l'intérêt porté à ces réseaux. Le domaine de recherche, en pleine effervescence, essaie d'apporter des solutions aux défis intrinsèques à un tel environnement. Ces défis natifs demeurent encore un handicap pour le développement d'applications pour les réseaux sans fil intégrant des réseaux ad hoc et leur déploiement réel.

Initialement étudiés à des fins militaires, les réseaux ad hoc sont promis à un large spectre d'utilisation. En effet, grâce à leur aptitude d'auto configuration et l'absence d'une infrastructure fixe, ces réseaux constituent une solution pratique et intéressante offrant mobilité, flexibilité et faible coût de déploiement et d'utilisation. Ces caractéristiques attrayantes nous donnent la possibilité de s'affranchir des contraintes de câblage et d'infrastructures de communication lourdes, tout en offrant un système d'échange de données très utile permettant à un ensemble de noeuds de communiquer, par exemple dans le cadre d'un groupe de travail. Ainsi, Chaque station est apte à établir une communication avec n'importe quelle autre station. Une telle communication sera directe lorsque la station destination se trouve dans le voisinage immédiat de la station émettrice. Or, la limitation du rayon de propagation des transmissions des hôtes mobiles est l'une des propriétés induites par l'utilisation du support sans fil. Ainsi, le recours aux services des autres noeuds intermédiaires sera nécessaire lorsque les parties communicantes sont distantes. Dans une telle situation, le réseau ad hoc doit collaborer et doit se répartir les tâches afin de fournir un service réseau autonome et performant, de façon totalement distribuée. Ainsi, pour l'envoi d'un paquet, chaque noeud du réseau doit participer afin de trouver la localisation de la destination recherchée et lui acheminer le paquet.

Toutefois, les réseaux ad hoc, en plein essor, soulèvent de nombreuses questions, notamment concernant le routage, la conservation d'énergie, la sécurité, la qualité de service, ... Malgré le nombre imposant des travaux déjà réalisés touchant à plusieurs de ces problématiques inhérentes aux réseaux ad hoc, les défis faisant face à de tels réseaux sont encore loin d'être résolus [30].

En effet, l'utilisation d'un médium sans fil pour la communication entre les différents noeuds restreint les ressources disponibles ainsi que la bande passante dédiée aux utilisateurs. La diffusion d'information est rendue malaisée à cause de l'instabilité et du manque de fiabilité des liens radio, vu qu'un tel support est sujet aux interférences et aux collisions. Étant donnée la nature partagée de ce médium radio, toute émission de données va être réceptionnée par plusieurs noeuds à des puissances variables. Certains voisins, autres que la destination, peuvent subir des collisions malgré l'existence de mécanismes chargés de gérer les collisions et le partage du médium. Les hôtes ne se trouvant pas dans le voisinage immédiat de l'émetteur peuvent être victimes d'interférences. Ces interférences s'ajoutent au bruit et détériorent les communications. Ainsi, le taux d'erreur s'accroît d'autant, et la transmission des paquets non récupérables diminue le débit de la liaison. Par addition, l'asymétrie des liens due au phénomène d'évanouissement constitue une autre caractéristique des réseaux ad hoc. Cette dernière constitue un véritable handicap pour l'acheminement des données. Ainsi, ces contraintes physiques liées à l'utilisation du médium sans fil introduisent un certain nombre de problématiques à prendre en considération lors de la conception de protocoles dédiés à ces réseaux particuliers.

Par ailleurs, un réseau ad hoc est doté de capacités qui lui permettent de s'organiser et s'auto-configurer de manière autonome sans avoir recours à une infrastructure fixe. L'absence de structure fait que l'ensemble des services, à l'instar du routage, est généralement réparti entre les entités. Pour assurer la connexité dans un réseau ad hoc, chaque noeud est tenu alors à jouer le rôle d'hôte et de routeur afin d'assurer l'acheminement de trafic transitant à travers le réseau. De ce fait, chaque noeud intermédiaire doit participer à la phase de découverte nécessaire pour l'établissement du chemin, au transfert des paquets échangés ainsi qu'à la maintenance de ce chemin. Or, l'absence d'une administration centralisée et d'une infrastructure fixe rend difficile la maintenance des routes établies suite aux déplacements éventuels des noeuds impliqués dans l'acheminement des paquets. En effet, le transfert de données peut aisément être interrompu par le départ d'un noeud sur la route utilisée. Vu les limitations des réseaux ad hoc, la construction et la maintenance des routes doivent être faites avec un minimum de contrôle et de consommation de la bande passante. Ces tâches sont d'autant plus complexes dans un environnement ad hoc mobile en constante évolution, étant donné que tout noeud peut se déplacer librement et indépendamment des autres. Les changements topologiques rendent la fonction d'établissement des routes une tâche difficile et engendrent des réparations fréquentes sur les routes déjà établies. L'implication continuelle de tout noeud à ces tâches additionnelles peut être pénalisante, étant donnée l'autonomie énergétique réduite des différents hôtes.

Motivations et objectifs

Beaucoup de propositions, répondant aux contraintes et objectifs précédemment évoqués, ont été définies en s'appuyant sur une vue à plat du réseau ad hoc. La mise en oeuvre des protocoles proposés est nécessairement distribuée et incombe à tous les noeuds du réseau. En

effet, tous les terminaux sont considérés comme égaux et doivent contribuer de façon solidaire à la gestion du réseau pour accomplir toutes les tâches, normalement fournies par l'infrastructure réseau. Or, la capacité réduite du médium de communication partagé requiert la minimisation du trafic de contrôle nécessaire au bon fonctionnement du réseau. Faire participer tous les noeuds aux tâches additionnelles (la recherche de routes, le transfert de données, la diffusion, ...) va contribuer à encombrer le canal avec des messages de contrôle et puiser des ressources limitées et partagées au détriment des paquets de données. Prenons l'exemple des protocoles de routage, le recours à la diffusion est adopté dans les approches proactives et réactives [1, 47]. Le routage proactif se base sur le maintien d'une table de routage actualisée à travers la diffusion périodique des informations topologiques. Alors que le routage réactif entame une phase d'inondation pour la recherche d'une route donnée avant l'établissement de toute communication entre toute paire de noeuds.

Avec l'état imprévisible des liens et les changements continus de la topologie, un routage dynamique requiert d'énormes ressources pour véhiculer la signalisation indispensable aux mouvements des noeuds. Tout protocole essaie de s'adapter continuellement et rapidement à ces changements dès la détection d'une défaillance de route afin d'offrir des performances optimales sur la durée. Cette surabondance de trafic de contrôle additionnel dégrade considérablement les performances du réseau, notamment lorsque la vitesse des noeuds est relativement élevée. Jusqu'à présent, la plupart des protocoles de routage ad hoc se sont montrés plutôt inaptes à faire face à une forte mobilité.

Or, l'utilité de tels réseaux se concrétise lors de leur déploiement dans les aéroports, les hôtels, les espaces de rencontres entre professionnels et les campus universitaires. Un tel déploiement fait intervenir un grand nombre d'utilisateurs nomades. Pour cette raison, les performances du réseau ne doivent pas pour autant chuter de façon drastique lorsque le nombre de participants augmente. Il doit passer à l'échelle, et présenter un trafic de contrôle négligeable afin de ne pas perturber le trafic de données. Or, adopter une vue plate du réseau et attribuer des rôles égalitaires à tous les noeuds présentent des limitations au passage à l'échelle, vu la croissance du trafic de contrôle avec le nombre de noeuds.

Dans ce contexte, le concept de clusterisation peut être mis à profit afin de faire face aux problèmes de passage à l'échelle et d'accroître les performances du système [27, 73]. La structuration d'un réseau permet de rendre son exploitation plus aisée et de supporter plus éfficacement les protocoles de haut niveau. C'est ainsi que plusieurs mécanismes de clusterisation ont vu le jour pour organiser un réseau ad hoc de grande envergure et donner une vue abstraite de ce dernier, disjointe de la topologie radio réelle. Par ailleurs, avec la croissance exponentielle des communications sans fil, une grande variété d'équipements portables a pénétré le marché (PC portables, PDA[1], ...). Cette hétérogénéité observée dans les réseaux ad hoc en terme de

[1] PDA : Personal Digital Assistant

ressources disponibles au niveau de chaque noeud (capacité de stockage, puissance de traitement, énergie, ...) offre une hiérarchisation native dans les rôles à attribuer aux noeuds du réseau. Les noeuds ayant des ressources abondantes et faiblement mobiles sont plus aptes à assurer des fonctionnalités additionnelles et à jouer le rôle de chef de groupe. Tirer parti de cette hiérarchie permet de prolonger la durée de vie du réseau et d'éviter l'extinction des noeuds n'ayant pas une autonomie en énergie élevée en leur épargnant la participation aux tâches additionnelles. La notion de clusterisation a été adoptée afin d'améliorer et d'optimiser les performances des protocoles pour les réseaux ad hoc, à l'instar du routage, la sécurité, la conservation d'énergie, ...

Etant donné l'intérêt porté au concept de clusterisation et ses apports indéniables pour améliorer le passage à l'échelle dans un environnement ad hoc, le choix du mécanisme de clusterisation s'avère important. De ce fait, un algorithme de clusterisation doit en premier lieu être capable de sélectionner les noeuds appropriés pour assurer les fonctionnalités du chef de groupe. Des candidats faiblement mobiles, avec une autonomie en énergie élevée, de grandes capacités de stockage et de mémoire, sont aptes à assurer des tâches additionnelles. Un tel choix pourrait refléter l'hétérogénéité naturelle des réseaux ad hoc et offrir une infrastructure virtuelle exploitant au mieux les ressources disponibles dans un tel environnement hétérogène, sans pénaliser les noeuds défavorisés. En second lieu, il doit essayer d'atténuer l'impact du trafic de contrôle échangé durant la phase de formation et de maintenance de la structure virtuelle pouvant dégrader les performances du système ou affecter les performances d'autres algorithmes reposant sur cette structure de clusterisation. Les spécificités et caractéristiques du médium sans fil et d'un environnement ad hoc doivent être prises en considération lors de la conception de l'algorithme de clusterisation. Finalement, l'algorithme doit s'adapter continuellement aux changements de la topologie et maintenir une infrastructure virtuelle actualisée et cohérente, sans nécessiter un trafic de contrôle important.

Plan de thèse

La thématique traitée dans cette thèse s'articule en trois grands volets : la proposition d'une technique de clusterisation, l'exploitation de cette dernière pour faire de la conservation d'énergie et la proposition d'un algorithme de routage. La deuxième partie introduit notre approche de conservation d'énergie, elle allie les apports de la clusterisation et du mode veille. L'algorithme de routage proposé jouit aussi des apports de la clusterisation offerts par notre mécanisme, introduit au niveau de la première partie. Ces contributions sont complétées par une évaluation de performances basée sur des simulations et des expérimentations.

Dans le chapitre 1, nous définissons la notion de clusterisation dans un environnement ad hoc. Une étude des approches de clusterisation existantes, ayant permis de faire dégager les carences de certaines de ces approches, est proposée aussi dans ce chapitre. Nous avons essayé d'établir un regroupement des approches selon une classification se basant sur la notion

de domination de graphe. Malgré les avantages indéniables de l'utilisation d'une infrastructure virtuelle pour l'amélioration des performances du système, la formation et la maintenance d'une telle dorsale requièrent du temps et des ressources. Ainsi, après avoir énuméré les domaines d'exploitation de l'infrastructure virtuelle générée grâce à une technique de clusterisation, nous avons essayé de cerner le coût requis pour sa formation et sa maintenance.

Le chapitre 2 introduit notre mécanisme `TBCA` [2] de clusterisation, qui prend en considération des spécificités et caractéristiques du médium sans fil. Ce dernier se déroule en trois étapes qui se chevauchent : élection des clusterheads, élection des gateways et la phase de vérification de la connexité de l'ensemble dominant formé par l'union des clusterheads et gateways élus. Cette dernière phase est requise afin de s'assurer de la formation d'un ensemble dominant connexe, apte à fournir un support efficace pour les protocoles de haut niveau. Notre approche a pour principal intérêt d'organiser le processus de clusterisation en couches, limitant ainsi à un instant donné le nombre de candidats au rôle de clusterhead. Exploiter la notion de collision pour accélérer la finalisation du processus de clusterisation constitue l'un des apports de notre mécanisme. Par ailleurs, une étude des performances générales au travers de simulations viendra compléter cette présentation dans le chapitre 3, outre les preuves de validité.

La formation et la maintenance d'une infrastructure virtuelle ne sont pas une fin en soi. L'objectif est d'exploiter cette dorsale pour assurer les services réseaux couramment utilisés et supporter plus efficacement les protocoles de haut niveau, à l'instar du routage, la sécurité, la conservation d'énergie, ... Dans cet axe, le deuxième volet s'intéresse à la mise à profit de notre mécanisme pour assurer une conservation d'énergie sans pénaliser les performances du réseau.

La problématique de conservation d'énergie dans les réseaux ad hoc sera abordée au niveau du chapitre 4. Malgré le progrès technologique, l'énergie demeure une ressource critique. En effet, l'interface réseau constitue le premier consommateur d'énergie. Outre l'énergie consommée durant les opérations d'émission et de réception, un noeud consomme de l'énergie même en étant inactif au niveau applicatif. L'écoute continue du canal et l'interception des paquets destinés à des noeuds voisins constituent un gaspillage inutile de l'énergie. Pour cet effet, le standard 802.11 définit deux modes de fonctionnement : le mode actif où toute station écoute constamment le canal et le mode économie d'énergie où la station est dans un état dormant. Une présentation du mécanisme `PSM` [3], opérant avec le mode veille afin de réduire l'écoute passive du canal, est proposée conformément au standard 802.11. Après avoir présenté les différentes améliorations apportées au mécanisme PSM, nous étudions les mécanismes ayant exploité conjointement le mode veille et une technique de clusterisation pour réduire la consommation d'énergie sans toutefois pénaliser les performances du réseau. Uniquement, quelques approches de conservation d'énergie ont essayé d'exploiter la notion d'ensemble dominant connexe pour atteindre cet objectif.

[2] TBCA : Tiered Based Clustering Algorithm
[3] PSM : Power Saving Mechanism

Une évaluation de l'impact du mécanisme de clusterisation adopté sur l'efficacité de l'approche de conservation d'énergie utilisée est aussi proposée au niveau du chapitre 4. Pour ce faire, on a intégré notre mécanisme de clusterisation dans une approche de conservation d'énergie existante se basant sur le concept d'ensemble dominant et on a comparé les performances des deux mécanismes.

Le chapitre 5 se consacre à la présentation d'une nouvelle approche de conservation d'énergie basée sur l'exploitation des apports de la clusterisation conjointement avec le mode veille, baptisée `CPPCM` [4] avec ses deux variantes. Notre objectif principal est de réduire la consommation d'énergie tout en assurant l'acheminement des paquets de données sans endurer des temps d'attente importants aux niveaux des files d'attente des noeuds impliqués dans le transfert. A travers cette nouvelle approche, on vise à réduire la consommation globale d'énergie par paquet et à augmenter le nombre de paquets délivrés avec succès. Agissant de la sorte, on est apte à apporter une certaine garantie de qualité de service pour certains types de trafic, à l'instar des trafics multimédia et temps réel. L'évaluation de performances réalisée confirme l'efficacité de cette approche.

La clusterisation est une solution préconisée pour le routage. Dans cet axe, la présentation d'un algorithme de routage se basant sur l'existence d'une infrastructure virtuelle fera l'objet du chapitre 6. L'exploitation des apports de notre mécanisme `TBCA` et la limitation des tâches de routage additionnelles à un sous ensemble de noeuds sont des atouts pour assurer le passage à l'échelle de notre algorithme.

Enfin, le dernier chapitre conclura cette thèse. Nous exposerons également quelques perspectives à l'ensemble des travaux présentés dans ce document. Ce travail a été réalisé dans le cadre de ma thèse en cotutelle au sein de l'équipe HANA de l'Ecole Nationale des Sciences de l'Informatique de Tunis et le laboratoire LaBRI de l'Université de Bordeaux 1.

[4] CPPCM :Cluster based Prioritized Power Conservation mechanism

Chapitre 1

Clusterisation dans les réseaux Ad hoc

1.1 Définition de la clusterisation

Fournir un accès ubiquitaire à l'information indépendamment des facteurs temps et lieu à un utilisateur, de plus en plus nomade, est devenu possible, en particulier avec l'émergence du nouvel environnement mobile ad hoc. Caractérisé par une absence d'infrastructure fixe, ce dernier permet de former des réseaux mobiles totalement dynamiques et spontanés. Initialement déployés pour servir dans le domaine militaire et dans les applications de tactique, ces réseaux constituent une solution facile pouvant fournir une infrastructure sans fil pour toute application où le déploiement d'une infrastructure filaire est trop contraignant ou non économique pour une utilisation temporaire. En effet, la mobilité accordée aux noeuds formant un réseau ad hoc, ainsi que leur aptitude à être déployé rapidement et leur faculté d'auto configuration leur permettent de former rapidement une infrastructure réseau apte à assurer l'échange de données entre plusieurs hôtes distants. Ces caractéristiques attrayantes ont contribué à proliférer l'intérêt porté à ces réseaux. Ce domaine a suscité l'intérêt d'un nombre imposant de groupes de travail et constitue toujours un axe de recherche majeur, le nombre des études réalisées en témoigne. Les études relatives aux problèmes rencontrés dans les réseaux ad hoc se sont intéressées à la sécurité, au routage, à la qualité de service, à la conservation d'énergie, ... [30], chacun de ces thèmes a fait l'objet de plusieurs travaux de recherche.

Ces problématiques sont toujours d'actualité compte tenu des spécificités de l'environnement radio (médium partagé, sensibilité aux interférences, ...) et la mobilité potentielle des utilisateurs qui rendent la qualité de transmission des informations loin de fournir les performances requises et de satisfaire les exigences de qualité indispensables pour les différents types de trafic combinant vidéo, voix, audio. Par ailleurs, la limitation des ressources en terme de bande passante et d'énergie, accentuée par la collaboration de tous les noeuds pour assurer la communication et maintenir la connectivité, impose aussi des défis additionnels pour ce type de réseaux.

La technique de clusterisation constitue une des alternatives visant à pallier à certaines

problématiques intrinsèques aux réseaux ad hoc de grande envergure. Elle consiste à agréger et à organiser les noeuds en un ensemble de groupes, chaque groupe étant contrôlé par un chef de groupe, appelé '*clusterhead*'. L'objectif est d'obtenir un ensemble de clusters couvrant l'ensemble de la population des noeuds, de manière à ce que les noeuds sélectionnés forment une infrastructure virtuelle permettant de surmonter les problèmes de passage à l'échelle et d'assurer une utilisation meilleure des ressources du réseau [25, 73].

Par ailleurs, l'hétérogénéité observée dans les réseaux ad hoc en terme de ressources disponibles au niveau de chaque noeud (capacité de stockage, puissance de traitement, ...) offre une hiérarchisation native dans les rôles attribués aux noeuds du réseau. Les noeuds ayant des ressources abondantes sont plus aptes à assurer des fonctionnalités additionnelles et à jouer le rôle de chef de groupe. Cette hiérarchisation permet d'avoir une vue abstraite de la topologie du réseau.

Plusieurs mécanismes de clusterisation ont été proposés dans la littérature. La classification des approches existantes peut se baser sur plusieurs critères. La première classification évidente peut recourir aux métriques et critères utilisés pour le choix des chefs de groupes. Dans cet axe, certains mécanismes de clusterisation ont choisi des critères simples, en se basant par exemple sur l'identifiant ou la connexité pour le choix des clusterheads [8, 12, 25, 34, 39, 68]. Alors que d'autres approches ont adopté des sélections plus élaborées en s'appuyant sur une combinaison de critères afin de sélectionner les noeuds les plus appropriés pour assurer les fonctionnalités du chef de groupe [10, 19, 22]. Une autre classification peut se baser sur le nombre de sauts qui sépare un noeud ordinaire du chef auquel il est rattaché. Dans cette classification, on retrouve deux catégories : une clusterisation à un saut et une clusterisation multi-sauts. Classer les approches existantes selon l'objectif final de la structure générée est aussi une autre classification possible [100].

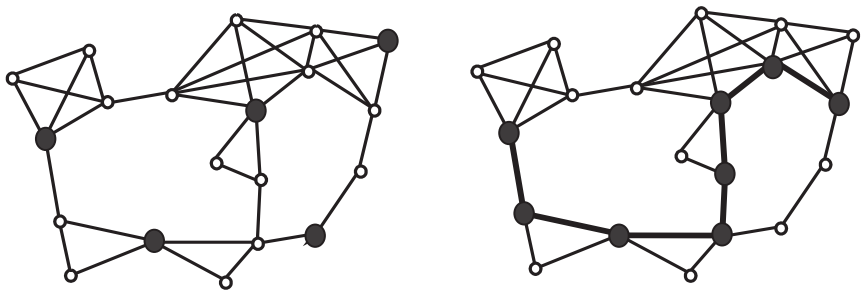
Dans la section suivante, on va présenter une classification inspirée de la notion de domination de graphes [27], en focalisant en particulier sur les approches proposées dans le but de construire un ensemble dominant connexe. Cette orientation a été motivée par le fait qu'une infrastructure offerte par un ensemble dominant connexe supporte plus efficacement les protocoles de haut niveau reposant sur cette dernière.

1.2 Classification des algorithmes de clusterisation

Plusieurs approches de clusterisation se sont inspirées de la notion de domination de graphes ou une de ses variantes. Généralement, pour modéliser un réseau ad hoc, on utilise un graphe non orienté $G = (V, E)$, où V représente l'ensemble des noeuds et l'ensemble E correspond à l'ensemble des arêtes représentant les liens existants entre les différents noeuds. Un lien relie deux noeuds s'ils se trouvent dans la portée l'un de l'autre.

Dans ce qui suit, on va introduire des définitions et des notations, qui nous sont nécessaires [71, 93, 27]. La notation $N(v)$ correspond au voisinage d'un noeud v. Tandis que le voisinage fermé, noté $N[v]$, d'un noeud donné v regroupe tous les sommets adjacents à ce dernier, en comptabilisant le noeud v aussi. Le voisinage fermé $N[S]$ de l'ensemble $S \subseteq V$ correspond à l'union $\bigsqcup_{v \in S} N[v]$. Alors que le voisinage ouvert $N(v)$ du noeud v du graphe G inclut les sommets adjacents à v, noté $N(v) = N[v] \setminus v$.

Un ensemble dominant du graphe $G = (V, E)$ est un sous ensemble $S \subseteq V$ tel que tout noeud $v \in V$ est soit un membre de l'ensemble S, soit un noeud adjacent à un noeud de S. Tout noeud de S domine ses voisins ordinaires, dits dominés. Ainsi, un ensemble de noeuds est dit **dominant**, si tout noeud du réseau est soit un noeud de l'ensemble, soit un voisin d'un noeud de l'ensemble dominant. L'exemple illustré dans la figure (1.1.a) montre un ensemble dominant formé par les noeuds noirs.



(a) Un ensemble dominant (b) Un ensemble dominant connexe

Figure 1.1 – Ensemble dominant

Un ensemble dominant est dit **indépendant** s'il n'y a pas deux noeuds de l'ensemble dominant qui soient adjacents. Ainsi, un ensemble indépendant S est défini de manière formelle :

$$\forall v \in S, \neg(\exists u \in S / u \in N(v)) \tag{1.1}$$

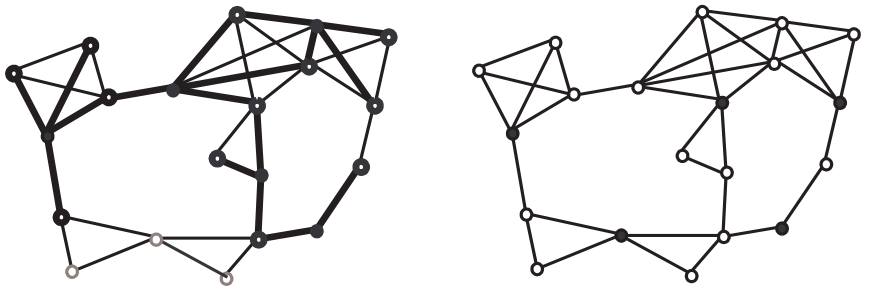
L'ensemble dominant S est dit **connexe** si tous les noeuds dominants sont directement connectés. Chaque noeud dominant peut jouer le rôle de clusterhead, son domaine inclura ses voisins ordinaires immédiats. On note que pour un ensemble dominant connexe d'un graphe G, le sous graphe induit $< S >$ est connexe. Le sous graphe $< S >$ inclut les noeuds dominants et l'ensemble des arêtes permettant de les connecter. La définition formelle de l'ensemble S est comme suit :

$$\forall u \in S, \exists v \in S / v \in N(u) \tag{1.2}$$

$$\forall (u, v) \in S^2, \exists c = chemin_{u \to v} / w \in c, w \in S \tag{1.3}$$

Les sommets d'un ensemble dominant connexe induisent donc un sous graphe pouvant servir comme une infrastructure virtuelle permettant de réduire considérablement les redondances lors des opérations de diffusion. L'ensemble des noeuds noirs, exposé dans la figure (1.1.b), forme un ensemble dominant connexe.

Alors qu'un ensemble dominant faiblement (*weakly dominating set*) est un sous ensemble $S \subseteq V$ ayant pour sous graphe induit $< S >_w = (N[S], E \cap (N[S] \times S))$, $< S >_w$ étant le sous graphe incluant tous les sommets de S, leurs voisins et toutes les arêtes du graphe initial ayant au moins un des deux sommets extrémités appartenant à S. S est dit un ensemble **dominant connexe faiblement** (*weakly-connected dominating set*) si S est dominant et $< S >_w$ est connexe. La figure (1.2) expose l'exemple d'un ensemble dominant connexe faiblement et le sous graphe induit.



(a) Un sous graphe faible induit dominant

(b) Un ensemble dominant connexe faiblement

Figure 1.2 – Ensemble dominant faiblement

Dans la suite de cette section, on présente une classification des différentes approches existantes établie en se basant sur la notion de domination de graphe. On va essayer de donner un aperçu exhaustif des différents mécanismes de clusterisation. Toutefois, l'intérêt sera focalisé en particulier sur les approches optant pour la construction des ensembles dominants connexes.

1.2.1 Clusterisation avec les ensembles dominants indépendants

Dans cette catégorie, on retrouve les approches ayant insisté sur le fait que l'ensemble formé par les chefs de groupe élus est nécessairement indépendant, en d'autres termes deux clusterheads ne peuvent pas être directement connectés.

Parmi les approches ayant opté pour la sélection des chefs de groupe suivant des critères simples, on retrouve les deux mécanismes proposés dans [39]. Le premier mécanisme `highest degree` se base sur le degré d'un noeud. Ainsi, chaque noeud diffuse la liste de ses voisins à son voisinage immédiat. Le noeud, ayant le nombre le plus élevé de voisins non couverts (n'ayant pas

encore décidé du rôle à assurer), se déclare comme clusterhead. En cas d'égalité, on a recours à l'identifiant comme deuxième critère de sélection pour briser la symétrie. Les voisins immédiats du clusterhead ne sont plus autorisés à participer au processus d'élection, ces derniers vont devenir membres du cluster formé. Un noeud, n'ayant pas encore élu un clusterhead, se considère comme un noeud non couvert. Un tel noeud vérifie son éligibilité au rôle de clusterhead, en tenant compte de ses voisins non couverts uniquement. Tout noeud, se trouvant à proximité de plusieurs clusterheads, peut jouer le rôle du gateway pour assurer la communication entre ses voisins clusterheads. Toutefois, le nombre de ré affiliation des noeuds est élevé à cause des déplacements éventuels des noeuds. Par conséquence, la probabilité de ré élection d'un ancien clusterhead est faible même s'il a perdu un seul noeud membre. Ainsi, dans un environnement dynamique, l'ensemble dominant est sujet à des changements fréquents défavorisant la stabilité de cet ensemble. Le deuxième mécanisme proposé dans [39], `lowest ID`, utilise l'identifiant comme critère de sélection des chefs de groupe. Le noeud, ayant le plus petit identifiant parmi son voisinage immédiat, se déclare comme clusterhead. Optant pour ce critère de sélection, les changements, affectant les statuts de clusterheads, seront moins fréquents. Toutefois, les noeuds, ayant des identifiants minimaux et ayant conservé leur statut de chef de groupe pour une longue période, risquent d'épuiser leur énergie rapidement. Aucune fonction de maintenance n'a été spécifiée pour ces deux mécanismes.

Une version modifiée de l'algorithme `lowest ID` a été proposée dans [68]. Cette version permet de résoudre certains problèmes spécifiques à ce dernier. Cet algorithme distribué est initié par tous les noeuds ayant un identifiant minimum parmi leurs voisins. En communiquant leurs décisions de créer des clusters, ils permettent à leurs voisins de sélectionner leur clusterhead, en choisissant celui ayant le plus petit identifiant parmi leurs voisins qui se sont déclarés clusterheads, et de diffuser leur décision. Si tous les voisins ayant les identifiants minimums d'un noeud ont déjà communiqué leur décision de clusterisation et aucun d'eux ne s'est déclaré chef de groupe, ce dernier décide de créer son propre cluster et informe ses voisins de son élection comme clusterhead. Ainsi, chaque noeud diffuse sa décision de clusterisation une fois tous les voisins ayant un identifiant plus petit ont déjà pris leur décision. Chaque noeud sera alors apte à déterminer son cluster et transmet uniquement un message durant tout l'algorithme. Une procédure de maintenance a été définie. Toutefois, dans certaines situations, elle peut conduire à des clusterheads isolés nécessitant des procédures additionnelles pour fusionner et réarranger les clusters. Par ailleurs, le rôle du clusterhead dans [68] est uniquement important pour la formation de la structure de clusterisation et n'entre pas en jeu dans les décisions de routage.

Pour partitionner les noeuds en clusters, une fonction poids a été définie dans [34], égale à la somme des degrés des voisins divisée par l'identificateur additionné au nombre total des noeuds. Ce mécanisme permet aux noeuds frontières de déterminer leur rôle en premier, toutefois, il tolère aussi l'existence des clusterheads isolés. Il requiert aussi la connaissance du nombre global des

noeuds formant le réseau.

Pour aboutir à une configuration stable et profiter de meilleures performances, `MOBIC` [11] adopte la mobilité comme critère de sélection de clusterheads, étant donnée qu'elle est la principale cause engendrant des changements de clusterheads et affectant les adhésions des hôtes à un cluster. Solliciter fréquemment la fonction de maintenance des clusters peut affecter les performances du réseau, en terme de débit et de délai d'acheminement. La métrique présentée dans `MOBIC` vise à capturer la mobilité dans le voisinage d'un noeud donné afin de s'en servir pour former des clusters stables. Elle est basée sur un rapport entre les mesures successives des puissances reçues au niveau de chaque noeud provenant de ses voisins. La métrique de mobilité proposée ne requiert aucune information sur la localisation absolue d'un noeud (obtenue grâce à un système GPS[5]), ni des informations sur la vitesse de déplacement de chaque noeud. Les auteurs proposent de constituer une idée claire sur la mobilité relative entre chaque paire de noeuds à partir de la mesure de la puissance du signal détectée faite pour deux transmissions successives de messages Hello périodiques provenant d'un noeud voisin. Grâce au calcul de la valeur de la mobilité locale, notée My, un noeud peut déterminer s'il est le plus apte à assurer le rôle de clusterhead. Une petite valeur de My par rapport à ses voisins montre que le noeud est le moins mobile vis-à-vis de ses voisins, donc il est le plus adéquat à devenir clusterhead. Alors qu'une valeur élevée de My par rapport à ses voisins indique qu'il est très mobile et se déplace continuellement. Le mécanisme `MOBIC` est adapté à des scénarios mettant en jeu des hôtes mobiles se déplaçant en groupe pour une longue période, le groupe de noeuds se déplace avec une même vitesse et dans la même direction. Ainsi, le clusterhead peut garder sa faible mobilité par rapport à ses voisins longtemps. Cependant, si les hôtes mobiles se déplacent de manière aléatoire et changent leur vitesse de temps à autre, les performances de `MOBIC` se voient dégrader. D'un autre côté, un noeud doit recueillir des informations de son voisinage afin de déterminer s'il est le moins mobile par rapport à son entourage immédiat. Le déplacement des noeuds peut entraîner la collecte d'informations erronées. `MOBIC` se base aussi sur la mobilité relative pour la formation des clusters initialement, et néglige les contraintes de mobilité dans la plupart des situations de maintenance ultérieurement.

L'algorithme `WCA`[6], proposé dans [22], est basé sur l'utilisation d'une métrique combinée tenant en considération plusieurs paramètres du système, à savoir le degré idéal d'un noeud, la puissance de transmission, la mobilité et l'énergie disponible dans la batterie au niveau du noeud, afin d'élire le noeud le plus approprié. Pour assurer l'équilibrage de charge entre les clusters et un accès efficace au médium, un seuil δ a été défini pour limiter le nombre de noeuds pouvant être gérés par un clusterhead. Ainsi, chaque membre du cluster aura sa part de ressources sans avoir à subir un temps d'attente long. Chaque noeud calcule une métrique W prenant en considération la différence Δ_v entre le degré du noeud et le seuil prédéfini δ, la somme des distances D_v le

[5]GPS : Global Positioning System
[6]WCA : Weighted Clustering Algorithm

séparant de tous ses voisins, la vitesse moyenne du déplacement du noeud et le temps cumulatif P_v, durant lequel le noeud a été clusterhead afin d'estimer la quantité de l'énergie de la batterie qui a été consommée. Chacun des paramètres impliqués dans la métrique possède un facteur poids choisi en fonction des besoins du système. Cette flexibilité dans le changement des poids associés aux différents paramètres permet l'application de l'algorithme avec plusieurs types de réseaux. Ainsi, le noeud ayant la plus petite valeur de W est choisi comme clusterhead et tous les voisins immédiats du clusterhead ne sont plus autorisés à participer à la procédure d'élection. Le clusterhead va assurer la communication entre les membres de son cluster. Pour cela, il utilise une faible puissance pour communiquer avec les noeuds de son voisinage, alors qu'il utilise une puissance plus élevée pour assurer la connexité des clusters et communiquer avec les clusterheads des clusters adjacents.

Or, la procédure d'élection de `WCA` requiert une grande quantité d'information afin de pouvoir calculer la métrique combinée pour faire le choix initial du clusterhead. Ceci nécessite une longue période sans déplacement pour tous les voisins. Par ailleurs, il n'est pas facile d'estimer la distance entre deux noeuds mobiles dans un environnement réel. En plus, se baser uniquement sur P_v ne peut pas garantir une bonne évaluation de la consommation d'énergie. En effet, la communication des données consomme une grande quantité d'énergie variant considérablement d'un noeud à un autre. L'activité du noeud durant cette période a un impact direct sur la consommation de l'énergie, étant donné qu'un noeud gaspille de l'énergie même en restant à l'écoute du canal. Par ailleurs, la procédure d'élection est invoquée initialement pour la formation des clusters. Toutefois, elle est non périodique, on fait appel à cette procédure que lorsque les clusterheads ne sont plus capables de couvrir l'entière population des noeuds. Lorsqu'un noeud se déplace dans une zone non couverte par un clusterhead, la procédure d'élection est déclenchée à nouveau. Ainsi, de nouveaux clusterheads sont élus en se basant sur la métrique combinée W afin de couvrir l'entière population. Ce type de reclusterisation détruit entièrement la structure de clusterisation actuelle et nécessite l'échange d'un grand nombre de messages afin d'aboutir à un nouvel ensemble dominant. `WCA` ne refait pas la re-clusterisation lorsqu'un noeud change son cluster d'attachement, chaque noeud doit estimer de manière continue la puissance du signal provenant de son clusterhead afin de pouvoir décider de changer d'affiliation ou pas. Malgré que ce mécanisme contribue à maximiser la stabilité des clusters, il permet aussi à des clusterheads de maintenir leur rôle sans avoir à re-vérifier la valeur W pour certaines situations de maintenance.

De leur côté, les auteurs de [10] ont abordé le sujet sous un angle différent en proposant une approche générale permettant de sélectionner les clusterheads en se basant sur un poids générique associé à chaque noeud. L'idée fondatrice est de permettre à chaque noeud de déterminer son rôle uniquement lorsque tous ses voisins ayant un poids plus grand que lui ont déjà décidé. L'apport de [10] est de pouvoir utiliser tout critère significatif pour le choix du clusterhead, on pourrait même inclure des paramètres relatifs à la mobilité des noeuds afin de choisir les noeuds les plus

stables pour assurer les fonctionnalités de chef de groupe.

La majorité des approches citées requièrent une phase de collecte d'information sur le voisinage immédiat. Certains mécanismes nécessitent parfois plusieurs échanges successifs de messages de contrôle [11, 22] contribuant ainsi à surcharger le réseau. Par ailleurs, la décision de tout noeud, pour certains mécanismes, repose sur les décisions prises dans le voisinage. Un tel fonctionnement présente des défis d'implémentation et de convergence certains. En effet, un noeud risque d'attendre indéfiniment, étant donnés les risques éventuels d'interférence et de collision qui n'ont pas été pris en considération dans ces approches. Finalement, aucune indication n'est fournie sur la procédure de sélection des noeuds intermédiaires qui vont assurer la communication entre chefs de groupe adjacents.

Le tableau récapitulatif suivant (1.1) essaie d'établir une comparaison entre la majorité des différentes approches exposées précédemment. Le premier critère de comparaison spécifie si l'algorithme utilise des messages de contrôle additionnels lors de la phase de clusterisation. Le deuxième critère indique si la phase de maintenance produit un effet de chaîne. Ceci se produit lorsque la mise à jour du statut d'un noeud entraîne la modification des statuts des noeuds avoisinants. Le troisième critère nous informe si le temps de finalisation du processus de clusterisation est constant, alors que le dernier critère quantifie le coût en nombre de messages échangés lors de la phase de clusterisation.

	Recours à des messages de contrôle pour la clusterisation	Effet de chaîne lors de la maintenance	Temps constant pour la clusterisation	Coût de la communication
Lowest ID [39]	Oui	Oui	Non	O(m*n)
Higest degree [39]	oui	Oui	Non	O(m*n)
[68]	Oui	Non	Non	O(n)
MOBIC [11]	Oui	Oui	Non	O((2*Δ+1+m)n)
WCA [22]	Oui	Oui	Non	O((Δ+1+ m)n)

Table 1.1 – Comparaison des algorithmes de clusterisation avec les ensembles dominants indépendants

On note que Δ fait référence au degré maximal d'un noeud, n est le nombre de noeuds formant le réseau et m représente le nombre moyen d'envoi de la décision d'un noeud.

1.2.2 Clusterisation avec les ensembles dominants

Le recours aux ensembles dominants indépendants, formés par les clusterheads, pose problème en cas de changements de la topologie. En effet, le rapprochement de deux clusterheads oblige l'un des deux à renoncer à son rôle de noeud dominant au profit de l'autre. Toutefois, les changements affectant l'ensemble des clusterheads ne s'arrête pas à ce niveau. Une telle modification peut entraîner une suite de changements pouvant se propager à travers le réseau entier.

En relaxant la condition d'indépendance pour l'ensemble dominant, on évite l'occurrence d'une telle chaîne de réaction en cas de changements de la topologie. Dans ce cas, on considère un simple ensemble dominant, sans contrainte d'indépendance.

Dans [66], les auteurs proposent un algorithme distribué rapide qui tient compte des interférences pour la construction d'un ensemble dominant dans un environnement asynchrone. Dans ce travail, on assume que les noeuds ne possèdent aucune information concernant leur voisinage immédiat et ni un mécanisme fiable de détection de collisions.

Les auteurs de [64] opèrent différemment en déclenchant une phase de formation de zones de manière distribuée. Ce processus est déclenché par les noeuds frontières ayant au plus un ou deux voisins à travers des fusions successives. Ce processus de fusion s'arrête lorsque la taille de la zone atteigne une certaine limite ou toutes les zones voisines ont déjà fusionné avec d'autres initiateurs. A chaque nouvelle fusion, un nouveau coordinateur de zone est sélectionné en privilégiant le noeud ayant le plus grand nombre de voisins non membres de la zone. Par la suite, la clusterisation s'effectue de manière indépendante au niveau de chaque zone en interdisant le partage de noeuds entre clusters. Une fois la formation des zones achevée, le coordinateur devient le gérant et initie le processus de clusterisation en diffusant un message aux noeuds membres. Chacun de ces derniers renvoie un nombre aléatoire, le noeud ayant tiré le nombre le plus grand sera élu comme clusterhead et ses voisins immédiats constitueront les membres. Par la suite, le gérant reprend le même processus pour les noeuds membres restants de la zone. Pour les opérations de maintenance, les messages de contrôle Hello sont échangés de manière régulière afin de détecter les changements qui ont survenus.

1.2.3 Clusterisation avec les ensembles dominants connexes

Le recours à la construction d'un ensemble dominant connexe semble une approche prometteuse pour fournir une infrastructure virtuelle apte à supporter de manière plus efficace les protocoles de haut niveau, à l'instar du routage. Le déploiement d'une telle structure pourrait être exploitée pour limiter les inconvénients de la diffusion, en attribuant ces tâches à ce sous ensemble de noeuds. Pour cet effet, le nombre de noeuds formant cette infrastructure virtuelle constitue un critère important. Sélectionner un grand nombre de noeuds dominants va anéantir les avantages de la clusterisation recherchés. Or, la construction d'un ensemble dominant connexe minimal d'un graphe est un problème NP complet [43]. Le recours à certaines heuristiques a été nécessaire pour arriver à approcher ce problème.

Les auteurs de [33, 80] proposent une série d'algorithmes de routage pour les réseaux ad hoc. L'idée de base consiste à déterminer le sous graphe qui forme un ensemble dominant connexe minimal. La connaissance du voisinage à 2 sauts est requise pour la construction de la dorsale, où on fait recours à la construction d'un arbre recouvrant minimal en se basant sur les poids des

arêtes pour l'augmentation de ce dernier.

Dans [20], la construction de l'ensemble dominant se base sur la construction d'un MIS (*Maximal Independent Set*). Un MIS consiste en un ensemble dominant dans lequel on n'a aucune paire de noeuds adjacents. Ainsi, toute paire de noeuds appartenant au MIS est séparée par au moins deux sauts. Pour la construction du MIS, l'affectation d'un rang à chaque noeud est nécessaire. Pour l'attribution de ce rang, on construit un arbre recouvrant (*spanning tree*) T ayant pour racine un noeud donné v, pouvant être choisi par un algorithme d'élection. Une fois la racine sélectionnée, les noeuds de l'arbre peuvent identifier leur niveau par rapport à cette dernière. En premier, la racine annonce son niveau 0. En interceptant le message d'annonce du niveau de son père, tout noeud calcule son niveau en incrémentant de un celui de son père dans l'arbre puis le diffuse à ses voisins. Chaque noeud sauvegarde le niveau de chaque voisin. Le niveau de chaque noeud dans l'arbre constitue le rang de ce dernier. La construction de l'ensemble dominant connexe est par la suite déclenchée par le noeud racine, initiant un processus de marquage des noeuds en blanc, gris et noir. Les noeuds marqués en noir et en gris forment l'ensemble dominant. Toutefois, la tâche du routage sera assurée uniquement par les noeuds marqués en noir et non tous les noeuds de l'ensemble dominant connexe.

Dans [87], un chef déclenche la construction du MIS à travers l'envoi de son message Hello. La sélection des autres noeuds dominants se base sur une fonction de poids, en privilégiant les noeuds ayant le poids le plus élevé dans le voisinage. L'algorithme distribué proposé dans [89] se déroule aussi en deux phases : la construction d'un MIS et la construction d'un arbre dominant. La première phase repose sur l'existence d'un arbre recouvrant T. La racine de cet arbre initie un processus de marquage des noeuds en gris et en noir en garantissant que la distance entre deux noeuds en noir est de deux sauts. Ces derniers forment le MIS. Par la suite, la racine de l'arbre T déclenche la construction d'un arbre dominant T^*.

Dans [94], on propose un algorithme distribué simple et efficace pour calculer un ensemble dominant connexe pour un graphe non orienté. Pour atteindre cet objectif, un processus de marquage des noeuds est exécuté au niveau de chaque hôte. Il permet de marquer tout noeud satisfaisant une certaine condition. Initialement, tout noeud échange la liste de ses voisins avec son voisinage. Un noeud, ayant deux voisins non connexes, se considère comme un noeud dominant. Une fois le processus de marquage achevé, la réduction de la taille de cet ensemble généré à l'issue de cette première phase de marquage se fait à travers deux règles. La règle 1 permet de démarquer tout noeud ayant son voisinage déjà couvert par un autre noeud dominant, possédant un identifiant supérieur au sien. Alors que la règle 2 permet d'exclure de l'ensemble dominant tout noeud ayant son voisinage couvert par deux autres voisins marqués, s'il a le plus petit identifiant. L'application de ces règles requiert la diffusion de l'état pour informer le voisinage de son statut. La finalité de la construction de l'ensemble dominant connexe est de servir dans le processus du routage. Ainsi, une fois l'ensemble dominant connexe déterminé, l'acheminement des paquets se

fera grâce aux noeuds dominants. Par ailleurs, les changements topologiques imposent parfois le re-calcul de cet ensemble dominant. Trois types de changement topologique ont été pris en considération, l'activation d'un nouveau noeud, l'arrêt d'un noeud et le mouvement d'un noeud. Pour assurer la maintenance de cette dorsale, tout noeud en mouvement est tenu à envoyer périodiquement un signal. Ainsi, le déplacement d'un noeud peut empêcher plusieurs noeuds de transmettre ou de recevoir leurs propres paquets. De même, le déplacement simultané de plusieurs noeuds peut affecter considérablement la structure de l'ensemble dominant connexe engendrant la reconstruction de ce dernier. Ainsi, la maintenance de cette structure est coûteuse en terme de transmission et de trafic de contrôle, vu qu'elle nécessite un échange considérable de messages additionnels. Dans [95], on propose une variante de ce mécanisme où on introduit le critère de l'énergie lors de l'étape d'élimination des noeuds de l'ensemble dominant.

L'algorithme de Wu et Li a été amélioré en terme de nombre de messages dans [82]. Cette approche a été étendue pour s'appliquer aux graphes non orientés à travers l'introduction de la notion de l'ensemble absorbant [96]. Ainsi, un réseau ad hoc ayant des liens unidirectionnels va être modélisé par un graphe orienté $G = (V, E)$ où V représente l'ensemble des noeuds et E constitue l'ensemble des arêtes orientées, vu l'existence de liens unidirectionnels. Une arête orientée de u vers v est notée par la paire ordonnée (u, v) dans G. Dans ce cas, u domine v et v est un absorbant de u. Ainsi, l'ensemble des voisins dominants est défini comme suit $N_d(u) = \{w : (w, u) \in E\}$. Alors que l'ensemble des voisins absorbants est défini comme suit $N_a(u) = \{v : (u, v) \in E\}$. L'union des deux ensembles constitue le voisinage du noeud u, $N(u) = N_d(u) \cup N_a(u)$. Le processus de marquage adopté étant le même, tandis que les règles 1 et 2 sont modifiées comme suit :

- Règle 1 : On considère deux sommets u et v de G'. Si $N_d(u) - v \subseteq N_d(v)$, $N_a(u) - v \subseteq N_a(v)$ et $id(u) < id(v)$, alors le noeud u peut se démarquer $(m(u) = F)$.
- Règle 2 : On suppose que v et w sont reliés par un lien bidirectionnel pour G'. Si $N_d(u) - \{v, w\} \subseteq N_d(v) \cup N_d(w)$, $N_a(u) - \{v, w\} \subseteq N_a(v) \cup N_a(w)$ et $id(u) = min\{id(u), id(v), id(w)\}$, alors le noeud u peut se démarquer $(m(u) = F)$.

Avec $G' = (V', E')$ est le sous graphe de G induit par V', $V' = \{v_1, v_2, ..., v_k\}$ étant l'ensemble des sommets du sous graphe fortement connexe G'.

Dans [31], une généralisation des deux règles 1 et 2 a été proposée afin de réduire la taille de l'ensemble dominant. Cette règle générique, appelée règle k, permet de démarquer des gateways déjà couverts par k autres gateways, le paramètre k pouvant être n'importe quel nombre. Pour cet effet, on considère $G' = (V', E')$ le sous graphe de G induit par V', $V'_k = \{v_1, v_2, ..., v_k\}$ étant l'ensemble des sommets du sous graphe fortement connexe G'. On note aussi $N_d(V'_k) = \cup_{v_i \in V'_k} N_d(v_i)$ et $N_a(V'_k) = \cup_{v_i \in V'_k} N_a(v_i)$. La règle k permet à un noeud marqué de passer à l'état F si l'ensemble de ses voisins, aussi bien son ensemble dominant que son ensemble absorbant, est déjà couvert par k autres noeuds et qu'il possède l'identifiant minimum. En d'autres termes,

si $N_d(u) - V'_k \subseteq N_d(V'_k)$, $N_a(u) - V'_k \subseteq N_a(V'_k)$ et $id(u) = min\{id(u), id(v_1), id(v2), ..., id(vk)\}$. Ainsi, les règles 1 et 2 se réduisent à des cas spécifiques de la règle k, où $|V'_k|$ est limité à 1 et 2 respectivement. Deux manières ont été présentées pour implanter cette règle : réduite (*restricted*) et non-réduite (*non-restricted*). Pour la première, un noeud peut se démarquer uniquement s'il est couvert par un groupe de voisins marqués connexes. Alors que pour le deuxième, un noeud peut être couvert par un groupe de noeuds, situés à 1 ou 2 sauts de lui, connexes ou connectés à travers d'autres noeuds marqués. La première approche est meilleure vu qu'elle nécessite uniquement la connaissance des informations concernant les voisins situés à deux sauts d'un noeud donné.

Une approche similaire a été adoptée dans [23]. L'élection des noeuds dominants, dits coordinateurs, se fait de manière distribuée en se basant sur des informations locales. Un échange périodique de messages Hello permet à tout noeud de collecter des informations sur son voisinage à deux sauts. Ainsi, tout noeud se déclare coordinateur s'il a deux voisins incapables de communiquer directement ou à travers un ou deux coordinateurs. Cette règle assure que la totalité du réseau soit couverte par suffisamment de coordinateurs. Chaque noeud est apte à vérifier son éligibilité périodiquement à devenir coordinateur grâce aux messages Hello reçus. De même, chaque coordinateur vérifie périodiquement s'il peut céder son rôle, s'il s'aperçoit que tous ses voisins peuvent communiquer directement ou via d'autres coordinateurs. Ainsi, la phase de clusterisation est basée sur l'envoi périodique des messages Hello pour maintenir la structure de clusterisation actualisée et prendre en compte les modifications survenant sur la topologie du réseau. De son côté, l'algorithme, présenté dans [69], repose sur la vérification de la connexité de certaines paires de voisins par chaque noeud afin de pouvoir se décider du rôle à jouer. Les auteurs de [77] proposent une variante de `SPAN` [23] où un noeud v se considère un coordinateur s'il a deux voisins incapables de communiquer directement ou à travers un noeud intermédiaire plus prioritaire que le noeud v.

Les messages échangés par `MTCDS`[7] [103] sont des variantes du message beacon auquel on a rajouté certaines informations. Le protocole `MTCDS` se déroule en deux phases : élection d'un initiateur et la construction d'un ensemble dominant connexe. Durant la première phase, le noeud ayant la plus petite adresse MAC[8] est élu comme initiateur. Ce dernier est responsable du déclenchement de la deuxième phase ayant pour finalité la formation d'un ensemble dominant connexe. L'initiateur diffuse un message de déclaration chaque *initMax* périodes beacon, couvrant ainsi ses voisins immédiats. Tout noeud couvert déclenche un temporisateur *DSTimer*, s'il possède encore des voisins non couverts. A l'expiration de ce temporisateur, le noeud rejoint l'ensemble dominant et diffuse cette information à son voisinage, s'il est toujours éligible. Un noeud éligible est un noeud couvert ayant encore des voisins non couverts. A chaque intervalle beacon, tout noeud dominant diffuse un message pour informer son entourage de sa présence. Le mécanisme

[7] MTCDS : Mac-layer Timer-based Connected Dominanting Set
[8] MAC : Medium Access Control

`TECDS`[9] [63] est une extension du protocole `MTCDS`, qui intègre la prise en compte du facteur énergie lors de la construction de l'ensemble dominant connexe. Ainsi, le noeud, ayant le niveau d'énergie le plus élevé et le degré le plus grand, est choisi comme initiateur. Le facteur d'énergie est aussi pris en considération lors du calcul du temporisateur *DSTimer*.

Dans [38], des techniques locales pour la construction d'un ensemble dominant connexe dans un réseau ad hoc sont présentées. Ces dernières tiennent compte des pertes de messages dues aux éventuelles collisions pouvant survenir sur un médium sans fil. Le premier des algorithmes distribués décrit requiert la connaissance du voisinage à 3 sauts, tandis que le deuxième exige la connaissance du degré maximal et la taille du réseau. Ce dernier repose aussi sur la collecte d'information relative au voisinage à deux sauts.

Certaines approches, précédemment citées, requièrent des informations globales ou une coordination globale pour la formation de l'ensemble dominant connexe, ce qui limite leur déploiement aux réseaux statiques ou à faible mobilité. D'autres approches assument la connaissance de certaines informations sur le voisinage à l'initialisation du réseau. Or, l'obtention de telles informations sur le voisinage requiert une phase préalable de collecte, la connaissance d'informations concernant le voisinage à h sauts est nécessaire pour certaines approches afin de permettre à tout noeud de décider de manière locale.

Toutefois, l'exécution distribuée de la majorité des approches existantes ne prend pas en considération les caractéristiques du medium sans fil. En effet, ces solutions sont conçues en accord avec l'hypothèse que tout paquet envoyé est nécessairement reçu au bout d'un certain laps de temps. Cette hypothèse se contredit avec les spécificités du médium sans fil, sujet aux interférences et collisions. Par ailleurs, les paquets de contrôle vont concourir pour accéder au canal conjointement avec le trafic de données. Les approches existantes n'ont pas pris en considération le cas d'éventuelles collisions pouvant survenir entre paquets de contrôle et paquets de données et l'impact de telles pertes sur la pertinence des informations collectées au niveau de chaque noeud et sur le fonctionnement du mécanisme de clusterisation.

Le tableau récapitulatif suivant (1.2) essaie d'établir une comparaison entre les différentes approches exposées précédemment :

[9] TECDS : Timer-based Energy aware Connected Dominanting Set

	Recours à des messages de contrôle pour la clusterisation	Effet de chaîne lors de la maintenance	Temps constant pour la clusterisation	Coût de la communication
(Das et al, Sivakumar et al)[33, 80]	Oui	–	$O(n^2)$	$O(n^2)$
(Bo et al)[20]	Oui	—	O(n)	O(nlog(n))
(Wu et al)[94]	Oui	Oui	$O(\Delta^3)$	O(2n)
(Stojmenovic et al) [82]	Oui	–	$O(n^2)$	Ω(n)
(Wan et al)[89]	Oui	—	O(n)	O(n log(n))
(Chen et al)[23]	Oui	Oui	$O(\Delta^3)$	O(2n)
(Rieck et al)[77]	Oui	Oui	$O(\Delta^3)$	O(2n)

Table 1.2 – Comparaison des algorithmes de clusterisation avec les ensembles dominants connexes

On note que Δ fait référence au degré maximal d'un noeud et n représente le nombre de noeuds formant le réseau. Par ailleurs, certaines approches n'ont pas proposé de phase de maintenance de l'ensemble dominant connexe généré.

1.2.4 Clusterisation avec les ensembles dominants connexes faiblement

Chen et Liestman [25] ont introduit l'utilisation d'un ensemble dominant connexe faiblement pour la clusterisation des réseaux ad hoc. Une suite d'algorithmes pour approximer la construction d'un ensemble dominant connexe faiblement minimal est présentée, se basant sur les propositions de Guha et Khuller pour les ensembles dominants connexes [42, 43]. Un ensemble dominant connexe faiblement se construit progressivement par l'ajout d'un sommet à l'ensemble des noeuds dominants, chaque nouveau candidat étant a au plus 2 sauts d'un autre dominant. Les auteurs proposent aussi une implémentation distribuée en créant plusieurs pièces et en les fusionnant dans un deuxième temps. Dans [26], un algorithme de construction d'un ensemble dominant faiblement par zone est présenté. Chaque zone est un sous graphe connexe du réseau renfermant au plus *2x* sommets, x étant un paramètre de contrôle de la taille des zones. L'algorithme proposé se déroule en 3 phases. La première phase consiste à partitionner le graphe en des zones disjointes à travers la construction d'une forêt recouvrante (*spanning forest*). A l'achèvement de cette étape, la racine de chaque arbre représente le chef de zone. L'identifiant du chef est connu par tous les sommets d'une zone donnée comme étant l'identifiant de la zone. Durant cette deuxième étape, un algorithme est exécuté au niveau de chaque zone de manière indépendante afin de construire un ensemble dominant connexe faiblement minimal. Le chef de zone supervise ce processus à travers la collecte d'informations au fur et à mesure et propage les décisions prises à travers la zone. Durant la dernière phase, un ensemble dominant connexe faiblement pour le graphe entier est formé par l'ajout de sommets supplémentaires à l'union des ensembles dominants connexes faiblement de toutes les zones. Une extension de cet algorithme a

été proposée dans [28] afin d'assurer la maintenance de l'ensemble dominant connexe faiblement. Cette extension peut être adaptée à un algorithme par zone pour la maintenance intra zone en limitant son exécution à une zone donnée. Cet algorithme peut être aussi appliqué pour la maintenance d'un ensemble dominant connexe faiblement dans un graphe arbitraire. Toutefois, il utilise la diffusion afin de propager les messages de contrôle. Certaines opérations de maintenance, à l'instar de la défaillance d'un lien ou d'un noeud, requièrent un échange excessif de messages de contrôle entre la racine (ou le chef de zone) et les noeuds impliqués dans l'opération de maintenance. Deux algorithmes distribués pour la construction d'un ensemble dominant connexe faiblement sont proposés dans [7] en se basant sur la notion de MIS.

L'utilisation des ensembles dominants connexes faiblement peut être appropriée pour le déploiement d'un réseau Bluetooth. Un réseau basique consiste en un noeud désigné comme maître entouré par des noeuds esclaves, le regroupement de plusieurs réseaux basiques peut être modélisé à travers la notion d'ensemble dominant connexe faiblement. Toutefois, le recours à des ensembles dominants connexes faiblement présente certaines lacunes pour un environnement ad hoc. En effet, le manque de communication directe entre les clusterheads élus nécessite le choix de noeuds supplémentaires intermédiaires afin de maintenir la connectivité entre ces clusterheads et d'assurer avec efficacité les opérations de routage et de diffusion.

1.2.5 Clusterisation avec les ensembles dominants à k sauts

Une caractéristique particulière unie toutes les approches présentées précédemment, tous les noeuds membres sont directement rattachés à leur clusterhead. Ainsi, la distance séparant deux noeuds membres au sein d'un même cluster ne peut pas dépasser 2 sauts. Par opposition, cette restriction n'est pas applicable dans la clusterisation multi sauts où les noeuds membres ne sont pas nécessairement directement reliés à leur clusterhead d'attache. Dans un tel contexte, le recours aux services de certains noeuds membres sera requis pour permettre au clusterhead d'atteindre ses membres éloignés.

Les auteurs de [92] proposent un protocole simple pour la diffusion en considérant l'ensemble des informations des voisins se trouvant à k sauts. Ainsi, une source, voulant initiée une opération de diffusion, sélectionne un ensemble de noeuds connexes dans son voisinage à k sauts. Ce dernier, baptisé ensemble de noeuds d'acheminement (*forward node set*), permet de couvrir tous les noeuds se trouvant dans le voisinage à k sauts de la source. Ces noeuds d'acheminement sont sélectionnés niveau par niveau, à partir de la source jusqu'à arriver aux noeuds situés à *k-1* sauts de cette dernière. Pour chaque niveau i, un ensemble de noeuds est sélectionné afin de courir tous les noeuds du niveau suivant *i+1*, tout en essayant de minimiser cet ensemble. L'ensemble des noeuds sélectionnés dans le voisinage à k sauts d'un noeud donné forme un ensemble dominant connexe. Pour ce protocole, les noeuds, sélectionnés au niveau *k-1* de la source, sont dits des noeuds de bordure. Ces derniers deviennent les nouveaux émetteurs et recalculent leur ensemble

de noeuds d'acheminement dans leur voisinage à k sauts. Ce protocole requiert la connaissance du voisinage à k sauts par chaque noeud pour l'assister dans sa stratégie de routage. En effet, la propagation d'un paquet d'une source vers la totalité du réseau s'effectue sur deux étapes. En premier, l'émetteur recherche des noeuds d'acheminement pour couvrir tous ses voisins dans son voisinage à k sauts. Les noeuds sélectionnés par l'émetteur ont la tâche de diffuser le paquet reçu, alors que les noeuds non sélectionnés dans le voisinage à k sauts de l'émetteur se limitent à recevoir le paquet. Deuxièmement, les noeuds de bordure choisis par l'émetteur choisissent à leur tour leur ensemble de noeuds d'acheminement afin que le paquet diffusé puisse être propagé vers les noeuds se trouvant hors de la zone de couverture à k sauts de l'émetteur. Ces étapes se répètent jusqu'à ce que le paquet diffusé traverse la totalité du réseau. Par ailleurs, afin de réduire la redondance au niveau des zones couvertes, chaque noeud de bordure t ne considère pas les parties de sa zone déjà couvertes par tout autre noeud de bordure ayant un identifiant plus petit. De même, une partie de la zone couverte par le voisinage à k sauts du noeud de bordure est déjà couverte par la source, puisque tous les deux se trouvent respectivement à k sauts l'un de l'autre, cette partie est aussi exclue. Cette optimisation nécessite aussi la connaissance du voisinage des autres noeuds de bordure.

Le mécanisme `ECS`, proposé dans [101], a pour objectif de réduire le nombre de clusters et d'éviter la formation des petits clusters. Il garantit que pour tout cluster nouvellement créé, le clusterhead associé sera au moins à 3 sauts des autres clusterheads existants, à travers l'introduction d'un nouveau statut *clusterguest*. Le mécanisme `ECS` a le privilège d'éliminer la phase initiale dédiée à la formation des clusters, il permet de réduire le chevauchement entre les clusters et de prolonger la durée de vie des clusters sans ajouter un trafic de contrôle excessif. En effet, la période initiale requise à la formation des clusters n'est pas nécessaire du moment que chaque noeud mobile peut se déclarer comme clusterhead sans échanger des informations avec son voisinage tant qu'il n'est pas couvert par un clusterhead. La stabilité au niveau du mécanisme `ECS` est maintenue à travers le changement du statut de clusterhead lors de l'occurrence de certaines situations. A travers ce changement événementiel des statuts des noeuds, le mécanisme vise à réduire la fréquence de changements des clusters. Au début de la formation des clusters, le statut de tous les noeuds est *non spécifié*. Tout noeud ayant ce statut, peut se déclarer clusterhead à travers l'envoi d'un message particulier à l'expiration d'un délai aléatoire. Toutefois, la réception d'une autre déclaration force tout noeud, n'ayant pas encore spécifié son statut, de joindre le cluster de l'émetteur de cette déclaration. Une telle décision sera annoncée lors de l'envoi du prochain message périodique Hello. Un noeud, ayant reçu plusieurs déclarations, se déclare comme *clustergateway*. L'échange de ces messages permet à tous les noeuds de maintenir une table d'informations, incluant les identifiants de tous les voisins et leur statut. On fait intervenir le nouveau statut introduit lors de la maintenance des clusters. Un noeud membre peut s'acquérir un tel statut s'il s'aperçoit qu'il est connecté uniquement à des noeuds membres, suite à des changements topologiques. Un clusterhead, ayant uniquement des noeuds membres

gateways, renonce à ce statut pour devenir *clusterguest* afin de réduire le nombre de clusters formés. Dans une telle situation, la distance séparant le clusterhead et ses membres peut excéder un saut. Ce mécanisme se base sur un échange périodique de messages Hello pour collecter des informations sur le voisinage, à l'instar de la majorité des approches existantes. Outre ces messages périodiques, il nécessite des messages spécifiques pour la déclaration des clusterheads. Un échange de messages spécifiques est aussi requis lors de la rencontre de deux clusterheads.

Le même objectif est recherché dans [50]. L'une des caractéristiques de cette proposition est l'absence de connaissance globale, chaque noeud effectue des actions locales disposant uniquement de certaines informations relatives à son voisinage immédiat. La principale motivation est d'éviter les clusterheads isolés tout en minimisant le nombre de clusters formés à travers l'utilisation d'une fonction poids reposant sur la connexité des voisins. L'idée de base consiste à permettre aux noeuds les plus internes d'initier le processus de clusterisation. Ces derniers semblent plus aptes à assurer les tâches de clusterheads, ils ont tendance à rester à l'intérieur du réseau même en cas de déplacement évitant ainsi les changements radicaux dans la structure de clusterisation et offrant par conséquence une plus grande stabilité.

Dans [24], les auteurs proposent de combiner deux approches très connues en adoptant la connexité comme critère primaire et l'identifiant le plus petit comme critère secondaire. Une généralisation a été définie pour permettre à un cluster de renfermer des noeuds se trouvant à k sauts de leur clusterhead d'attache.

Les auteurs de [8] proposent une heuristique distribuée pour l'élection des clusterheads garantissant que chaque noeud sera au plus à k sauts de son clusterhead, k étant un paramètre de l'heuristique. Cette heuristique tend aussi à élire les clusterheads existants même en cas de changements de la topologie, elle permet de répartir la charge équitablement entre les différents clusters formés. Une première phase permet de faire propager les identifiants les plus élevés dans le voisinage à k sauts, alors que la deuxième phase assure la propagation de l'identité des clusterheads élus. Aucun mécanisme de maintenance n'a été spécifié en cas de changements topologiques dus à la mobilité éventuelle des noeuds.

Une approche similaire a été proposée dans [19] où on présente un algorithme de clusterisation permettant de grouper les noeuds en des clusters en sélectionnant un chef dont les membres peuvent être à une distance de k sauts au plus. Une sorte d'arbre prenant racine au niveau du clusterhead est créée pour assurer certaines fonctions additionnelles. Pour chaque noeud, un facteur de disponibilité AF (*Availability Factor*) est évalué afin de prendre en considération trois caractéristiques distinctes, à savoir l'énergie, la connexité et la mobilité. Cette évaluation permet de quantifier les conditions opérationnelles du noeud afin de déterminer son rôle. L'objectif est d'exploiter au mieux l'hétérogénéité des noeuds grâce à une métrique appropriée afin d'affecter les rôles critiques (de chef de groupe par exemple) aux noeuds disposant des ressources abondantes, par conséquence ayant des conditions opérationnelles meilleures. L'algorithme opère de manière

distribuée, la décision au niveau de chaque noeud est prise à travers l'évaluation des ressources et propriétés de ses voisins. L'affiliation d'un noeud à un cluster est liée à la stabilité de la branche de l'arbre qui va lui permettre de se rattacher au clusterhead et à la valeur de la métrique du clusterhead lui-même. L'algorithme `A-C` a été défini pour partitionner le réseau en des clusters assez larges afin de les exploiter dans le routage. Il permet de mettre en place une structure d'épine à l'intérieur de chaque cluster à exploiter pour les communications inter-clusters. Pour atteindre le clusterhead, chaque noeud choisit un voisin, membre du même cluster, pour être son prochain saut (*Next Hop*). Ce dernier choisit à son tour un prochain saut, et le processus se répète jusqu'à ce que le prochain saut soit le clusterhead. De cette manière, le clusterhead, les noeuds choisis comme prochain saut (NH) et les liens les liant forment la structure d'épine à l'intérieur du cluster. L'intérêt de ce papier s'est porté sur les réseaux ad hoc d'intérieur permettant à des hôtes fixes et mobiles d'échanger des données.

L'algorithme `MobDHop` [35, 36] permet la formation de clusters de diamètre variable. A travers la capture de la mobilité du groupe, l'algorithme essaie de former des clusters multi sauts plus stables. Ainsi, les noeuds ayant des modèles de mobilité similaires sont groupés dans un même cluster, limitant les changements de statuts et garantissant une certaine stabilité aux clusters formés. Par opposition aux autres algorithmes de clusterisation multi sauts, `MobDHop` ne requiert que la connaissance d'informations relatives au voisinage à un saut. En effet, une première phase de `MobDHop` aboutit à la formation de clusters à un saut. Cette phase nécessite l'échange d'un certain nombre de messages Hello afin de permettre à tout noeud de calculer les métriques de mobilité sur lesquelles se base sa décision, à savoir l'estimation de la distance par rapport aux voisins au cours du temps et l'estimation de la distance moyenne du cluster. Grâce à ces dernières, on vise l'élection du noeud le plus stable parmi ses voisins pour le rôle de clusterhead. La deuxième phase de fusion permet aux noeuds nouvellement introduits au réseau ou aux noeuds non couverts suite aux changements éventuels de la topologie de joindre des clusters existants. Dans une telle situation, tout noeud essaie de choisir le voisin le plus stable pour se rattacher à lui.

La majorité de ces propositions de clusterisation exige une connaissance du voisinage situé à k sauts au niveau de chaque noeud ce qui nécessite des retransmissions de certaines informations utiles pour la prise de décision. Cette collecte d'informations est gourmande en ressources, pouvant surcharger le réseau. Elle requiert aussi un certain temps requis pour la propagation des informations pour atteindre tous les voisins éloignés de k sauts. Par ailleurs, la maintenance des clusters est très difficile et compliquée en particulier dans un environnement à mobilité élevée.

1.3 Intérêts de la clusterisation

Malgré les handicaps spécifiques intrinsèques aux réseaux ad hoc, ces derniers présentent des avantages indéniables, à l'instar de leur déploiement immédiat et leur faible coût d'utilisation. Grâce à ses caractéristiques attrayantes, les réseaux ad hoc trouvent un champ d'application direct dans le domaine militaire. Ainsi, les communications entre fantassins, véhicules et engins aéroportés deviennent autonomes et spontanées. Ils sont largement utiles lors de l'organisation d'événements tels que des conférences et des salons afin de proposer un réseau de partage d'informations. Ils peuvent également être utilisés dans les aéroports et les campus universitaires. En effet, les réseaux hybrides constituent la principale application civile des réseaux ad hoc. Ils étendent le concept des réseaux sans fil classiques en proposant la création de réseaux d'accès cellulaires multi sauts. De ce fait, la couverture radio est étendue, proposant un Internet ubiquitaire autorisant tant les connexions vers Internet qu'entre paires de clients.

Toutefois, le problème de passage à l'échelle demeure l'un des freins majeurs au développement d'applications ciblant une grande population pour les réseaux sans fil intégrant des réseaux ad hoc. Dans ce contexte, la structuration d'un réseau a été envisagée comme une solution prometteuse permettant de remédier à ce problème et de rendre l'exploitation du réseau plus aisée. Les réseaux ad hoc sont dotés de capacités d'auto organisation. Grâce au recours à la notion de clusterisation, cette auto organisation transparaît à travers une structure virtuelle apte à offrir les services réseaux couramment utilisés.

Grâce à l'organisation du réseau à travers la formation des clusters, les noeuds de chaque cluster sont directement super visionnés par leur chef de groupe (clusterhead). Ce dernier peut coordonner l'accès au canal, épargnant ainsi les ressources gaspillées dans les retransmissions dues aux collisions. Le chef de groupe aura la charge de coordonner les activités à l'intérieur de son cluster. Chaque chef de groupe va veiller à assurer et à garantir une meilleure qualité de service et une communication plus efficace entre les noeuds. L'infrastructure virtuelle formée par ces derniers permet de fournir une solution pour surmonter les problèmes de passage à l'échelle et pour assurer une utilisation meilleure des ressources du réseau [27, 73].

La clusterisation est aussi une solution préconisée dans le routage afin de minimiser l'utilisation des ressources. En effet, dans un réseau ad hoc, la collaboration de tous les noeuds est primordiale afin d'assurer les fonctionnalités normalement fournies par l'infrastructure réseau, à l'instar du routage. Par ailleurs, les protocoles de routage dédiés aux réseaux ad hoc peuvent être classifiés en trois catégories : le routage plat, le routage hiérarchique et le routage géographique. Ce dernier type de routage assisté par des informations de localisation géographique nécessite que chaque noeud soit équipé par le mécanisme GPS. Les approches de routage plat adoptent un mécanisme d'adressage plat, chaque noeud participant au routage joue un rôle égalitaire par rapport aux autres noeuds.

Pour le routage plat, les algorithmes de routage actuels se distinguent en trois catégories, les algorithmes proactifs, réactifs et hybrides [1, 47]. Plusieurs protocoles de routage proactifs proviennent des algorithmes de routage classiques à état des liens LS (*Link State*). Ces algorithmes essaient de maintenir une information actualisée au niveau de chaque noeud concernant une route donnée pour atteindre tout noeud dans le réseau. Ainsi, les routes sont déjà disponibles lorsqu'une source désire émettre un paquet vers une destination donnée. Cette vue de la topologie est acquise grâce aux échanges périodiques de messages de contrôle contenant les informations de topologie requises. Ce type de protocole puise continuellement une certaine partie de la bande passante du réseau, qui croît conjointement avec le nombre de noeuds. Pour un réseau fluide, un protocole proactif échange abusivement des informations pour entretenir des tables de routage de faible taille. L'investissement réalisé pour le maintien des tables de routage n'est pas justifié en présence d'une activité réseau réduite, étant données que les informations topologiques propagées peuvent ne jamais être utilisées.

Tandis que, les protocoles de routage réactifs se caractérisent par le fait qu'aucune activité de routage n'est entamée et aucune information de routage permanente n'est maintenue au niveau des noeuds du réseau, si aucune communication n'est en cours. Ces algorithmes réactifs utilisent le routage à la demande initié par la source. Lorsqu'une source requiert une route vers une destination donnée, elle entame un processus de découverte de route à travers le réseau grâce à la diffusion d'une requête, induisant un certain délai pour l'établissement d'une route donnée. Ce processus d'inondation, gourmand en terme de ressources, perturbe le fonctionnement normal du réseau. Une fois la route découverte, elle est maintenue le temps nécessaire de l'échange ou jusqu'à ce que la destination devienne inatteignable. Suite à la défaillance d'un lien sur la route établie ou le départ sans préavis d'un noeud impliqué dans le transfert, la réparation des routes peut générer à son tour un surcoût de trafic. Pour la détection des défaillances de liens, la plupart des algorithmes ont recours à un échange périodique de messages Hello, le départ d'un voisin peut être détecté suite à l'absence de messages Hello successifs en provenance du voisin en question [75]. Les algorithmes hybrides à leur tour maintiennent une information partielle concernant la topologie du réseau. Ces protocoles essaient de réaliser un compromis entre protocoles proactifs et protocoles réactifs tout en se conformant à l'environnement ad hoc.

Ainsi, le recours aux paquets de contrôle est nécessaire soit de manière périodique pour les algorithmes proactifs, soit durant la phase de découverte de routes, à laquelle participent tous les noeuds, pour les algorithmes réactifs. Les diffusions périodiques ou intempestives de ces paquets additionnels peuvent pénaliser le trafic de données. La situation est plus critique en particulier pour les réseaux de grande envergure, étant donnée que la quantité de trafic additionnel introduite est proportionnelle aux nombre de noeuds. En effet, le risque de collisions grandissant engendre plus de retransmissions induisant un gaspillage d'énergie inutile. Par ailleurs, dans une approche réactive, la non fiabilité du mécanisme de diffusion, due aux collisions et aux interfé-

rences, peut être un handicap pour détecter le plus court chemin, ou empêcher la détection d'une route quelconque. Pour les protocoles proactifs, la non fiabilité du mécanisme de diffusion peut rendre certaines informations topologiques obsolètes ou incomplètes. Ces algorithmes opérant avec un mode de routage plat souffrent du problème de passage à l'échelle du à leurs mécanismes intrinsèques. Les protocoles proposés se basent sur des techniques de diffusion, coûteuses en terme de bande passante et aptes à surcharger le réseau. Pour cela, le routage hiérarchique a été adopté afin d'améliorer le passage à l'échelle, de réduire le trafic de signalisation et de permettre par conséquence des économies en énergie [29, 46, 73]. Ce dernier est basé sur l'organisation des noeuds du réseau en groupes et l'affectation de rôles différents à ses membres. Grâce à ces protocoles, la taille de la table de routage et celle des paquets de mise à jour vont être réduites en incluant uniquement une partie du réseau. La plupart de ces protocoles sont basés sur une technique de clusterisation. Grâce à cette technique, on évite les diffusions fréquentes des informations de routage pouvant surcharger le réseau et dégrader considérablement les performances de transmission du système.

A cet effet, le recours à la notion d'ensemble dominant connexe (*Connected Dominating Set*) a été adopté par plusieurs approches [20, 23, 33, 80, 82, 94]. L'objectif est de réduire le trafic de contrôle associé au routage et d'arriver à une meilleure utilisation des ressources à travers l'attribution de la tâche de routage à un sous ensemble des noeuds du réseau, dits coordinateurs ou noeuds dominants. L'apport de l'utilisation d'un ensemble dominant dans le routage consiste à centraliser tout le réseau en un sous ensemble de noeuds dominants et connexes, qui seront responsables de maintenir les informations de routage pour une approche proactive. Ainsi, la génération et la diffusion des informations de routage peuvent être limitées à ce sous ensemble de noeuds [49, 74, 83]. Grâce à ces protocoles, on évite les diffusions fréquentes des informations de routage pouvant surcharger le réseau et dégrader considérablement les performances de transmission du système. La taille de la table de routage et celle des paquets de mise à jour vont être réduites en incluant uniquement une partie du réseau. Recalculer les tables de routage ne sera pas aussi nécessaire si les changements topologiques n'affectent pas le sous-ensemble chargé de ces tâches additionnelles. Quand un noeud mobile change son cluster d'attache, seulement les noeuds, faisant partie du même cluster du noeud ayant quitté, actualisent leurs informations. Donc, les changements locaux n'affectent pas le réseau entier mais uniquement une partie, ainsi l'information traitée et stockée par chaque noeud est réduite.

Alors que pour une approche réactive, l'apport de ce genre de routage réside dans le fait que la recherche sera limitée à cet ensemble dominant. Ainsi, on libère les autres noeuds des fonctionnalités additionnelles qu'un hôte est tenu à réaliser telles que la participation à l'établissement des routes, le transfert des paquets ainsi que la maintenance de ces routes établies. Les opérations de diffusion peuvent être aussi limitées à ce sous ensemble de noeuds sélectionnés [44, 92]. En effet, ces tâches constituent une lourde charge, en particulier avec la présence des contraintes

liées à l'énergie et aux ressources limitées.

En effet, l'énergie est une ressource rare, étant donnés que les noeuds sont munis de batteries autonomes. Plusieurs solutions ont été proposées pour le problème de conservation d'énergie dans un environnement ad hoc, à savoir le contrôle de puissance, le routage orienté conservation d'énergie et le mode veille. Les deux premières approches prennent en considération le facteur d'énergie lors de la prise de décision concernant la puissance de transmission à utiliser ou le choix d'un chemin à emprunter. Le mode veille autorise les noeuds à entrer en mode dormant afin de réduire le gaspillage de l'énergie du à l'écoute passive du canal et l'interception des paquets destinés à autrui. Toutefois, permettre aux noeuds de devenir inactifs modifie la topologie du réseau, générant ainsi des impacts négatifs sur les décisions du routage et sur les performances du réseau. Dans cet axe, la notion de clusterisation a été déployée conjointement avec le mode veille pour réduire la consommation d'énergie sans toutefois dégrader les performances du système [23, 97, 98]. Les noeuds dominants auront à assurer les tâches de routage et de transfert des paquets de données afin de permettre aux autres noeuds à entrer en mode conservation d'énergie dès que possible. Autoriser les noeuds ordinaires à devenir inactif permet d'éviter le gaspillage d'énergie. Dans de telles approches, la rotation du rôle de noeud dominant de manière équitable entre les différents noeuds s'avère nécessaire, vu qu'un noeud dominant assure plusieurs tâches additionnelles. Ainsi, on évite l'épuisement de l'énergie de certains noeuds particuliers, étant donnée que l'extinction de ces derniers peut causer le partitionnement du réseau.

1.4 Coûts de la clusterisation

La technique de la clusterisation est considérée comme une approche prometteuse pour remédier aux problèmes de passage à l'échelle. L'établissement et le maintien d'une dorsale ne sont pas une fin en soi. Plusieurs protocoles peuvent reposer sur l'infrastructure offerte par la technique de clusterisation. Pour cela, cette infrastructure doit être aussi stable que possible afin de supporter de manière efficace d'autres protocoles (conservation d'énergie, routage, ...).

La plupart des approches existantes dans la littérature optaient pour la construction d'un ensemble dominant connexe minimal. En effet, le nombre de noeuds formant l'ensemble dominant connexe est un paramètre important, étant donné qu'il nous donne une idée sur le nombre de noeuds impliqués dans le routage et allant assurer des fonctionnalités additionnelles au profit des autres noeuds. Dans ce contexte, un algorithme de clusterisation doit générer une structure aussi réduite que possible afin de minimiser le nombre de noeuds participant aux tâches additionnelles et par conséquence le trafic additionnel à faire acheminer lors des opérations de routage. Toutefois, maintenir un ensemble dominant connexe minimal est une tâche non triviale dans un environnement dynamique [43]. Avoir une certaine redondance dans cet ensemble autorise plus de flexibilité en cas de défaillance d'un lien à cause des changements imprévisibles de la topologie

dus à la mobilité accordée aux noeuds.

La clusterisation est généralement réalisée en deux phases, une phase de formation des clusters et une phase de maintenance. La phase de formation des clusters est déclenchée à l'initialisation du réseau. Compte tenu de l'hétérogénéité native en terme de ressources dans un environnement ad hoc, un algorithme de clusterisation doit être capable de sélectionner les noeuds appropriés pour assurer les fonctionnalités du chef de groupe, ces derniers doivent satisfaire certains critères définis par le mécanisme de clusterisation [11, 22, 39, 68]. Certaines approches requièrent l'échange de plusieurs messages spécifiques pour la collecte d'un certain nombre d'informations nécessaires pour la prise de décision sur le rôle à prendre. L'objectif est d'opérer de manière distribuée en ayant recours à des informations locales. Cette étape préalable de collecte nécessite du temps pour la propagation des informations, étant données que certaines approches reposent sur la connaissance du voisinage à h sauts, pour un h réduit. Ce trafic additionnel introduit peut pénaliser le trafic de données [37, 53]. Par ailleurs, aucun mécanisme n'a traité le cas d'occurrence de collisions entre les paquets de données et les paquets dédiés à la clusterisation, ni les collisions pouvant survenir entre les paquets dédiés à la clusterisation, étant donnée l'exécution distribuée faisant participer tous les noeuds du réseau. L'impact de telles situations sur la pertinence des informations collectées et sur le processus de clusterisation n'a pas été pris en considération. Cette situation est d'autant plus grave pour les mécanismes de clusterisation où la décision d'un noeud se base sur les décisions diffusées dans son voisinage. Un tel noeud peut attendre indéfiniment dans un réseau dense ou à grande envergure où la probabilité d'occurrence de telles collisions est très élevée. De ce fait, la formation d'une structure virtuelle non homogène peut avoir des répercussions négatives sur le fonctionnement des protocoles reposant sur cette infrastructure virtuelle et sur les performances du système.

La majorité de ces approches assument aussi que les noeuds restent statiques durant cette étape ou se déplacent à une mobilité réduite. Cette hypothèse garantit l'obtention d'informations exactes et pertinentes. Or, la libre mobilité accordée aux hôtes résulte en des changements topologiques imposant des mises à jour et des modifications pour maintenir les informations topologiques actualisées et une structure de clusterisation aussi homogène que possible [37, 40]. Cette structure offre un support pour d'autres protocoles et la maintenir actualisée servira à garantir le bon déroulement et fonctionnement de ces derniers.

Certaines approches n'ont pas proposé un mécanisme de maintenance de la structure générée et n'ont fourni aucune indication sur la manière à suivre en cas de changements affectant la structure générée [39, 89]. D'autres ont procédé différemment en faisant recours au même algorithme déployé pour la formation des clusters ou de l'ensemble dominant connexe. Une telle solution permet de reconstruire à zéro l'infrastructure virtuelle, affectant par conséquence sa stabilité et perturbant le fonctionnement des protocoles de haut niveau. Certaines propositions reposent sur un échange périodique de messages de contrôle afin de maintenir les informations sur le voisi-

nage actualisées et de détecter à temps les changements affectant la topologie [22, 23, 94, 96]. Ainsi, en cas de changements locaux, une telle proposition procède à des réparations locales. Ces interventions locales sur la structure de clusterisation peuvent déclencher une chaîne de modifications affectant la totalité de la structure. De même, plusieurs réparations locales simultanées peuvent affecter l'homogénéité de la structure globale et y introduire des incohérences, affectant les performances d'autres algorithmes reposant sur la structure de clusterisation.

Ainsi, outre les messages nécessaires pour l'établissement de la structure de clusterisation, le trafic de contrôle périodique requis pour la maintenance constitue une charge supplémentaire au réseau. On rappelle que le nombre de messages de contrôle croit proportionnellement avec le nombre de noeuds. Dans un réseau ad hoc dense et large, ce trafic considérable va puiser une partie des ressources limitées en terme de bande passante, de temps de traitement et d'énergie. Les messages de contrôle vont concurrencer conjointement avec le trafic de données pour accéder au canal, surchargeant ainsi le réseau. Dans un tel environnement, le risque de collisions croissant peut induire des retransmissions inutiles de paquets de données, ce qui contribue à accentuer la consommation inutile de l'énergie.

Dans [53], nous avons montré que le choix du mécanisme de clusterisation revêt une grande importance pour mieux profiter des apports de la clusterisation. En effet, l'établissement et le maintien d'une dorsale induit un coût en terme de trafic de contrôle introduit et de temps nécessaire pour la formation de cette dernière. L'algorithme de clusterisation doit être aussi efficace en terme de temps et de messages vu la limitation des ressources. Il doit essayer d'atténuer l'impact des changements de la topologie et du trafic de contrôle échangé lors des opérations de réorganisation et de maintenance pouvant dégrader les performances du système ou affecter les performances d'autres algorithmes reposant sur la structure de clusterisation. Les auteurs de [40] ont évalué les répercussions possibles de la mobilité sur l'infrastructure virtuelle dans le but de proposer des techniques simples pour limiter cet impact.

1.5 Conclusion

A travers ce chapitre, nous avons présenté un recueil des approches de clusterisation existantes, avec une tentative de classification de ces dernières en se basant sur la notion de domination de graphe. Nous avons détaillé le fonctionnement de certaines approches afin de mieux cerner les carences. L'inconvénient majeur de la majorité des approches est le coût induit pour l'établissement et la maintenance de la structure générée en terme de temps requis et de trafic de contrôle introduit. Les remarques exprimées dans la section précédente mettent en lumière la nécessité de fournir un mécanisme de clusterisation qui tient compte des spécificités du medium sans fil et des caractéristiques inhérentes d'un environnement ad hoc. Fournir un mécanisme de clusterisation efficace en terme de temps et de messages est une obligation afin de ne pas com-

promettre les fonctionnalités des protocoles reposant sur la dorsale générée. Dans ce contexte, nous avons proposé un mécanisme de clusterisation qui prend en compte ces contraintes, et qui sera présenté dans le chapitre suivant.

Chapitre 2

Proposition d'un algorithme de clusterisation distribué sans connaissance du voisinage

La clusterisation est une solution préconisée dans le routage afin de minimiser l'utilisation des ressources et d'éviter les diffusions fréquentes pouvant surcharger le réseau. En effet, la maintenance des routes établies nécessite des diffusions fréquentes des informations de routage, suite aux déplacements éventuels des noeuds impliqués dans l'acheminement des paquets. Ce trafic de contrôle additionnel est apte à surcharger le réseau et dégrader considérablement les performances de transmission du système, en particulier pour les réseaux de grandes envergures.

La construction d'un ensemble dominant connexe permet de fournir une dorsale apte à assurer ces fonctionnalités additionnelles. La majorité des approches proposées requiert l'échange de messages de contrôle additionnels afin de recueillir des informations concernant le voisinage immédiat d'un noeud. Cette phase de collecte préalable a pour objectif de permettre à ce dernier de décider de manière locale du rôle à jouer tout en assurant l'élection de noeuds satisfaisant un certain nombre de critères afin d'être capable d'assumer les fonctionnalités additionnelles. Certaines de ces approches nécessitent même la connaissance du voisinage à deux sauts ou le voisinage à h sauts du noeud, h étant un entier relativement petit dans la plupart des cas. Ainsi, l'échange de messages Hello ou autres types de messages de contrôle s'avère primordiale pour la collecte de ces informations requises pour le bon déroulement du processus de clusterisation. Ce trafic supplémentaire constitue aussi une surcharge pour le système pouvant détériorer ses performances et pénaliser le trafic de données transitant à travers le réseau. La majorité des travaux assume que tout paquet envoyé sera reçu par la destination au bout d'un certain laps de temps. Or, de telles hypothèses se contredisent avec la nature du support sans fil caractérisé par un accès partagé et sujet aux interférences et collisions. Malgré, ces caractéristiques intrinsèques aux réseaux ad hoc, la plupart des approches proposées ne prévoie aucun mécanisme pour le

traitement des collisions et perte de paquets.

Par ailleurs, la maintenance de la structure de clusterisation est requise pour refléter les changements topologiques dus aux déplacements éventuels des noeuds mobiles. Plusieurs approches ont recours aussi à un échange périodique de messages de contrôle pour collecter des informations nécessaires au maintien de la structure virtuelle générée par le processus de clusterisation. Grâce à cet échange périodique, ces approches essaient d'appliquer des réparations locales à la structure afin de maintenir et d'assurer une certaine stabilité à l'infrastructure virtuelle tout en évitant des modifications inutiles pouvant affecter la structure de clusterisation. Cependant, le maintien d'une structure de clusterisation globale cohérente à travers des réparations locales est parfois insuffisant, en particulier dans un environnement ad hoc à mobilité élevée. En effet, ces réparations locales tiennent en compte des informations sur le voisinage à h sauts collectées à travers des échanges périodiques de messages de contrôle. Ces informations peuvent être partielles ou obsolètes, vu les pertes de paquets dues aux collisions et interférences et le temps nécessaire à la propagation d'informations sur un voisinage à h sauts, h étant strictement supérieur à 1.

Dans cette partie, nous présentons un nouvel algorithme de clusterisation destiné pour les réseaux ad hoc `TBCA` (`Tiered based Clustering algorithm`) [54], le principe de base de ce mécanisme a été présentée dans [13, 52]. L'innovation apportée se résume dans le fait que l'algorithme de clusterisation opère sans connaissance préalable du voisinage. Notre objectif principal est d'alléger le réseau du trafic de contrôle additionnel afin de libérer plus de ressources au profit du trafic de données. Ainsi, on vise à atténuer l'impact du trafic de contrôle échangé durant la phase de clusterisation et de maintenance de cette structure pouvant dégrader les performances du système ou affecter les performances d'autres algorithmes reposant sur la structure de clusterisation.

L'idée fondatrice est d'organiser le processus de clusterisation en couches. Ainsi, on est apte à limiter le nombre de noeuds participants au processus de clusterisation, à un instant donné, et pouvant envoyer les messages de contrôle dédiés à cette phase. Grâce à cette organisation en couches, on réduit la probabilité d'occurrence de collisions pouvant ralentir ou perturber le bon fonctionnement de la phase de clusterisation. Pour atteindre cet objectif, l'algorithme permet aux voisins situés à deux sauts d'un clusterhead uniquement de se déclarer clusterhaed (**CH**). Par ailleurs, nous allons exploiter les collisions pouvant survenir pour accélérer le processus de clusterisation. Pour cet effet, le processus de clusterisation va être initié par un noeud donné, appelé **DN** '*Designated Node*'. Ce choix a été motivé par l'observation des domaines d'applications possibles des réseaux ad hoc incluant le domaine militaire, les réseaux hybrides, les opérations de secours et d'exploration, les conférences, ... Pour la majorité de ces applications potentielles, l'existence d'un membre gérant est requise pour assurer la coordination des opérations, la prise de décisions et la collecte d'informations dans certains cas. Dans ce qui suit, on assume l'existence

d'un noeud **DN** unique, ce dernier étant le premier noeud qui a formé le IBSS[10].

Au début de la phase de clusterisation, tous les noeuds sont marqués à **N** (rôle Non assigné), excepté le noeud **DN**. Une fois le processus de clusterisation achevé, tout noeud aura l'un des statuts suivants : noeud clusterhead (**CH**), noeud gateway (**GW**) ou noeud membre (**M**). Notre objectif final est de construire de manière distribuée et simple un ensemble dominant connexe pouvant être déployé rapidement pour le support des autres protocoles de haut niveau. Cet ensemble sera formé par l'ensemble des clusterheads et gateways élus.

On assume que tous les noeuds sont synchronisés grâce à un agent externe. Au début de chaque période de synchronisation, par exemple à chaque TBTT[11] marquant le début d'un intervalle '*Beacon*', le noeud DN envoie un paquet *cbeacon*[12] après une période d'écoute égale à SIFS[13] ou à un DIFS[14]. Dans notre approche, on va utiliser une période d'écoute SIFS pour l'envoi du premier *cbeacon* lorsqu'on n'a pas recours à un mécanisme de conservation d'énergie, tel que `PSM`. Dans le cas contraire, la période DIFS sera utilisée.

Les noeuds voisins du DN envoient après un SIFS un ACK_BT, ce qui engendre une collision, considérée comme un '*Busy Tone*' (**BT**). Ce dernier va être entendu par les voisins se trouvant à deux sauts du noeud DN. Ainsi, un noeud, ayant entendu un ACK_BT ou détecté une collision BT, est éligible à devenir un clusterhead (CH) vu que l'interception de l'ACK_BT ou de la collision BT l'informe de son éloignement du CH précédent de deux sauts. De ce fait, un tel noeud a le droit de concurrencer pour devenir un clusterhead à travers l'envoi de son paquet *cbeacon* à l'expiration de son temporisateur, s'il n'a pas entendu auparavant la déclaration d'un autre clusterhead. Le processus de clusterisation se poursuit ainsi couche par couche.

Il faut noter qu'un processus de délégation du rôle de DN a été mis en place pour permettre à un noeud DN, estimant qu'il ne pourrait plus assurer ce rôle, de passer la main à un autre noeud. Un mécanisme a été défini pour garantir l'élection d'un successeur, choisi parmi les gateways voisins du noeud DN, en cas d'une panne ou du départ éventuel de ce dernier.

2.1 Construction de l'ensemble dominant connexe

2.1.1 Déclaration des Clusterheads et des noeuds membres

Le processus de clusterisation va être initié par le noeud DN. Ce dernier est le premier noeud à se déclarer CH à travers l'envoi d'un *cbeacon* après un (D)SIFS de l'instant TBTT. Les autres noeuds ne calculent aucun délai pour l'envoi de *cbeacon*. Initialement, ces derniers sont

[10] IBSS :Independent Basic Service Set
[11] TBTT : Target Beacon Transmission Time
[12] Cbeacon : Clustering beacon
[13] SIFS : Short InterFrame Space
[14] DIFS : Distributed InterFrame Space

marqués à N, vu qu'ils n'ont pas encore décidé du rôle à jouer. En interceptant le *cbeacon*, les noeuds voisins du DN vont se joindre au cluster formé par ce dernier. Ils attendent un SIFS avant d'envoyer un ACK_BT engendrant une collision BT. Cette collision BT, entendue par les voisins à deux sauts du DN, sert à informer ces derniers qu'ils sont aptes à concurrencer pour être clusterheads et à déclencher le processus de clusterisation au niveau suivant. Ainsi, tout noeud ayant entendu un *cbeacon* n'est pas autorisé à participer à la phase d'élection des clusterheads, alors que tout noeud ayant entendu un ACK_BT correct ou intercepté une collision BT est éligible à devenir clusterhead. Un noeud éligible attend un SIFS puis calcule un délai aléatoire pour l'envoi de son *cbeacon*. Le calcul du délai aléatoire peut prendre en considération plusieurs critères, tels que la mobilité et l'énergie disponible afin de donner l'avantage aux noeuds ayant suffisamment de ressources et relativement stables par rapport aux autres afin de se déclarer en premier et d'assurer le rôle de clusterhead.

Dans notre approche, on a opté pour l'utilisation d'informations locales qui ne requièrent aucune étape préalable de collecte. Tout noeud i, éligible à devenir clusterhead (**candCH**), calcule une borne personnalisée D_1^i en prenant en considération le facteur d'énergie. Le calcul de cette borne se fait comme suit :

$$D_1^i = \lceil ((\frac{(D_{1min} - D_{1max})}{E_{max}}) * E_i) + D_{1max} \rceil \tag{2.1}$$

Avec :

D_1^i : La borne calculée par le noeud i

D_{1max} : La borne maximale imposée pour le calcul du délai aléatoire pour se déclarer clusterhead

D_{1min} : La borne minimale imposée pour le calcul du délai aléatoire pour se déclarer clusterhead

E_i : L'énergie disponible au niveau du noeud i

E_{max} : L'énergie maximale d'un noeud

La nouvelle borne D_1^i favorise la déclaration des noeuds ayant suffisamment d'énergie, en leur permettant d'avoir une borne D_1^i plus réduite que les autres noeuds ayant une quantité d'énergie inférieure. Ainsi, le délai aléatoire d_1 choisi par le noeud est un nombre aléatoire de slots compris entre 0 et cette nouvelle borne D_1^i calculée par le noeud. Le choix d'une borne inférieure différente de zéro a été nécessaire afin d'éviter le cas où tous les noeuds ont des quantités d'énergie très élevées à l'initialisation du réseau (voire maximale). Dans une telle situation, les candidats au rôle de clusterhead vont calculer chacun une borne D_1^i très petite augmentant ainsi les probabilités de choisir un même délai aléatoire et de renforcer par conséquence le risque d'occurrence de collisions lors de l'envoi de *cbeacon*. Fixer la borne inférieure à une valeur fixe, strictement supérieure à zéro, permet de remédier à ce problème.

Une fois le délai d_1 calculé, la décrémentation du temporisateur se fait conformément à l'algorithme de backoff relatif au mécanisme CSMA/CA[15]. Un noeud éligible renonce à être clusterhead à la réception d'un message *cbeacon* de la part de l'un de ces voisins avant l'expiration de son temporisateur. A l'occurrence d'un tel événement, il rejoint le cluster formé par l'émetteur du *cbeacon* et se déclare comme noeud membre. Dans le cas contraire, il se déclare clusterhead en envoyant un *cbeacon*. Les noeuds ayant acquis le statut de membre auront la tâche d'informer le niveau suivant à travers l'envoi des ACK_BTs. Ainsi, la propagation de la clusterisation se fait couche par couche.

Comme l'expose la figure suivante (2.1), le noeud DN (noeud 1) et ses voisins immédiats (noeuds 2, 3, 5, 6 et 7) forment le premier niveau, ou la première couche. A la réception du *cbeacon* provenant du DN, ces noeuds vont se déclarer membre et se joindre au cluster présidé par le DN. Ces derniers ne sont plus autorisés à candidater pour le rôle de clusterhead. Les ACK_BTs envoyés par ces noeuds membres vont informer les noeuds à deux sauts du DN qu'ils sont aptes à concurrencer pour le rôle de clusterhead, à l'instar des noeuds 4, 8, 9, 10 et 12 de l'exemple. De cette manière, on déclenche le processus de clusterisation au niveau de la couche suivante. Les voisins à deux sauts du DN (noeuds 4, 8, 9, 10 et 12), ayant entendu un ACK_BT correct ou intercepté une collision BT, sont les candidats éligibles pour le rôle de clusterhead au niveau de la deuxième couche. Les voisins immédiats de ces derniers, n'ayant pas encore participé au processus de clusterisation (noeuds 13, 14 et 15), vont se joindre aux clusters formés par les candidats élus lors de l'étape de déclaration des clusterheads. Ainsi, les clusterheads déclarés parmi les candidats (noeuds 4, 8, 9, 10 et 12) et leurs membres forment la deuxième couche.

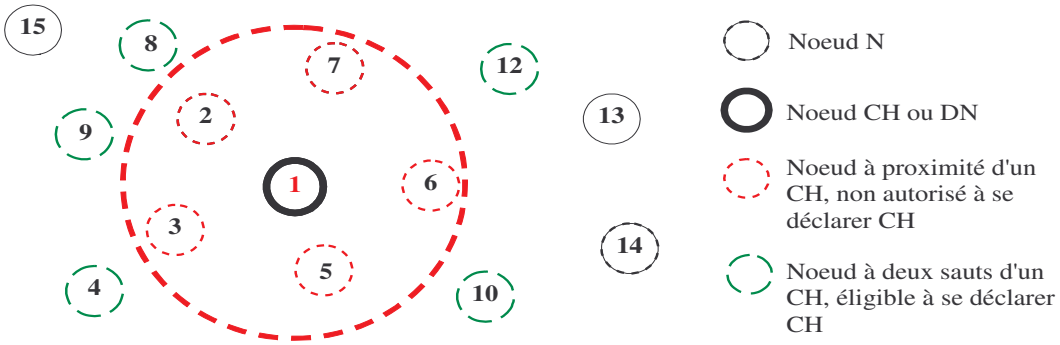


Figure 2.1 – Principe de fonctionnement de la clusterisation

Toutefois, un tel dénouement peut générer plusieurs collisions entre les *cbeacons* (dite collision CH) et les ACK_BTs. Par conséquence, les noeuds deviennent inaptes à se décider de l'action à entamer. Pour pouvoir différencier entre une collision CH (entre *cbeacons*), une colli-

[15]CSMA/CA : Carrier Sense Multiple Access with Collision Avoidance

sion BT (entre ACK_BTs) et une collision CH/ BT, on va procéder à la clusterisation du réseau ad hoc par couche centrée sur le noeud DN. On impose aussi la terminaison de la clusterisation entière d'une couche avant de passer à la suivante. De cette manière, on s'assure que tous les noeuds éligibles à devenir CH appartenant à une même couche se sont déclarés avant de permettre à ceux de la couche suivante de commencer le processus de clusterisation. Par ailleurs, afin d'éviter l'envoi simultané de *cbeacon* et d'ACK_BT, on impose l'envoi simultané des ACK_BTs à un instant précis pour chaque couche, une fois tous les clusterheads appartenant à ce niveau ont envoyé leur déclaration.

Pour atteindre cet objectif, à chaque TBTT, tout noeud divise son axe de temps en une succession de périodes, notées T_i, pour i = 1..N, N étant le nombre de phases requises pour permettre au processus de clusterisation de couvrir la totalité des noeuds. Ainsi, selon le contexte, l'instant t_i indique l'instant où la période i se termine, alors que la période T_i définit le temps nécessaire pour la clusterisation d'un niveau ou une couche i. Cette période T_i englobe une sous période pour les annonces de *cbeacon* suivie par une sous période pour l'envoi simultané des ACK_BTs. Cette dernière débute à l'instant t_{iBT}.

Afin de pouvoir déterminer les instants t_i et t_{iBT} de manière précise, on va calculer le temps maximal nécessaire pour l'envoi des *cbeacons* lors de la déclaration des candCHs et le temps requis pour la diffusion des ACK_BTs. Ainsi, la première sous période inclut une période d'écoute du canal, égale à SIFS, le backoff maximal pouvant être choisi pour transmettre un *cbeacon* et la durée de transmission d'un *cbeacon*, exception faite pour la première période T_1 où on ne comptabilise pas le backoff maximal vu que le *cbeacon* est envoyé par l'unique DN. Ainsi, chaque candidat au rôle de clusterhead attend une période SIFS avant de déclencher son temporisateur. Ce dernier est un nombre de slots aléatoire entre 0 et la borne personnalisée D_1^i, égale à D_{1max} dans le pire des cas. La décrémentation du temporisateur se fait conformément au mécanisme CSMA/CA, tant que le canal est libre. L'occupation du canal indique qu'un voisin est entrain d'émettre son *cbeacon*. A la réception d'un tel paquet, tout candidat renonce au rôle de clusterhead pour devenir un noeud membre.

La deuxième sous période est dédiée à l'envoi simultané des ACK_BTs. A l'instant t_{iBT}, tous les noeuds, ayant entendu un ou plusieurs *cbeacons* ou intercepté une collision durant la première sous période de T_i, envoient leur unique ACK_BT après une période SIFS de cet instant prédéfini t_{iBT}. De cette manière, on est capable d'éviter les collisions entre les *cbeacons* et les ACK_BTs afin de permettre à tous les clusterheads de se déclarer correctement, et, par conséquence, de différencier entre les collisions pouvant se produire entre *cbeacons* et collisions entre ACK_BTs.

Ainsi, si une collision survient avant l'instant t_{iBT} durant la première sous période de T_i, alors elle est considérée comme étant une collision CH, pouvant survenir entre deux ou plusieurs *cbeacons* provenant de noeuds voisins ou éloignés. Le cas d'une collision pouvant survenir entre

un *cbeacon* et une transmission de données, ayant débutée avant le TBTT de l'intervalle beacon en cours, ne se pose pas. On suppose qu'aucune transmission n'est autorisée durant le temps de clusterisation. Cette règle nous permet d'éviter le problème de collisions entre les paquets de données et les messages de contrôle dédiés à la clusterisation. Grâce à cette suspension momentanée du trafic de données, le processus de clusterisation peut se dérouler convenablement en s'assurant que tout paquet de contrôle associé à la phase de clusterisation sera reçu par le voisinage et que toute collision sera bien interprétée dans le cas contraire. Par ailleurs, on évite aussi les retransmissions des paquets de données.

Une collision, se produisant après l'instant t_{iBT}, est considérée comme étant une collision BT due à l'envoi simultané des ACK_BTs par tous les noeuds ayant intercepté un ou plusieurs *cbeacons* ou une collision CH durant la première sous période. L'envoi simultané d'ACK_BT permet d'informer les noeuds de la couche suivante qu'ils sont autorisés à initier le processus de clusterisation. Par ailleurs, le problème des collisions pouvant survenir entre les ACK_BTs et une transmission de données en cours ne se pose pas, vu que tout trafic de données va être suspendu avant d'initier le processus de clusterisation.

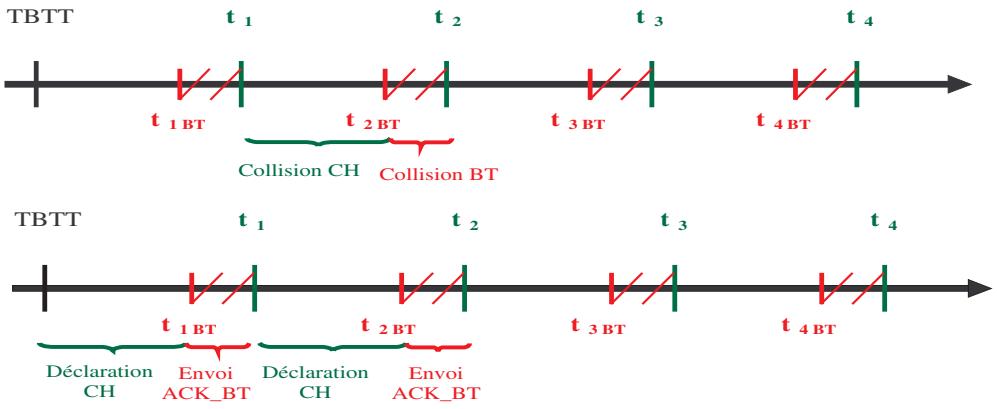


Figure 2.2 – Division du temps en sous périodes T_i et T_{iBT}

Pour exprimer les instants t_i et t_{iBT}, i =1 .. N, on considère les définitions suivantes :

Txcbeacon : Temps de transmission d'un *cbeacon*

TxBT : Temps de transmission d'un ACK_BT

SlotTime : Temps d'un slot

CWsize : Taille maximale de la fenêtre de contention en nombre de slot de temps

L'expression de t_i, pour i =1 .. N, est comme suit :

$$t_1 = SIFS + Txcbeacon + SIFS + TxBT \tag{2.2}$$

$$t_2 = t_1 + (SIFS + CWsize + Txcbeacon + SIFS + TxBT) \tag{2.3}$$

$$t_i = t_1 + (i-1) * (SIFS + CWsize + Txcbeacon + SIFS + TxBT), i = 2..N \tag{2.4}$$

L'expression de t_{iBT}, pour i =1 .. N, s'exprime comme suit :

$$t_{1BT} = SIFS + Txcbeacon \tag{2.5}$$

$$t_{2BT} = T_1 + (SIFS + CWsize + Txcbeacon) \tag{2.6}$$

$$t_{iBT} = T_{i-1} + (SIFS + CWsize + Txcbeacon), i = 2..N \tag{2.7}$$

Ainsi, le noeud DN est responsable de l'initiation du processus de clusterisation. Tout autre noeud va attendre que la vague de clusterisation l'atteigne afin de décider du rôle à jouer. Un tel noeud va se baser sur ce calcul pour déterminer les instants t_i et t_{iBT} relatifs à une couche i, marquant le début de la phase d'élection des clusterheads et la phase d'envoi des ACK_BTs au niveau de cette couche. Grâce au calcul continu des instants t_i et t_{iBT}, tout noeud, ayant reçu au moins un *cbeacon* ou a intercepté une collision CH, saura l'instant précis d'envoi de son ACK_BT. De même, tout noeud, ayant intercepté un ACK_BT correct ou une collision BT, sera notifié du commencement du processus de clusterisation au niveau de sa couche. Par ailleurs, chaque noeud est apte à connaître implicitement le nombre de pas qui l'éloigne du noeud DN. Grâce aux informations contenues dans les *cbeacons* correctement reçus, chaque noeud connaît ses clusterheads voisins.

2.1.2 Déclaration des noeuds gateways

2.1.2.1 Déclaration des candGW normaux

Durant le processus de clusteristion, un noeud, ayant entendu un *cbeacon*, devient un noeud membre (M) et ajoute l'émetteur du *cbeacon* reçu à la liste de ses clusterheads voisins (liste_CH). A la réception d'un ou plusieurs autres *cbeacons*, un tel noeud se déclare comme étant un candidat gateway (candGW). Un candidat gateway est un noeud éligible à assurer le rôle de gateway entre ses clusterheads voisins. Les *cbeacons* peuvent provenir de noeuds appartenant à la même couche ou être interceptés lors de la clusterisation de la couche suivante.

La figure (2.3) illustre la déclaration du noeud 6 comme étant un candidat gateway. En interceptant le *cbeacon* envoyé par le noeud DN 1, le noeud 6 devient un noeud membre (M). Les noeuds en pointillés (rouge), à l'instar des noeuds 2, 3, 5 et 7, ont aussi intercepté le même *cbeacon*, ils ne sont plus donc autorisés à candidater pour le rôle de clusterhead. Suite à la réception d'un autre *cbeacon* provenant du noeud 12 durant la phase de clusterisation de la couche suivant,

le noeud 6 se déclare comme candidat gateway, éligible à servir comme intermédiaire entre les clusterheads 1 et 12. Les candGWs 6 et 7 ont intercepté deux *cbeacons* durant la phase de clusterisation de leur couche et de la couche suivante. Alors que le candGW 14 a intercepté deux *cbeacons* uniquement durant la phase d'élection de CHs relative à sa couche.

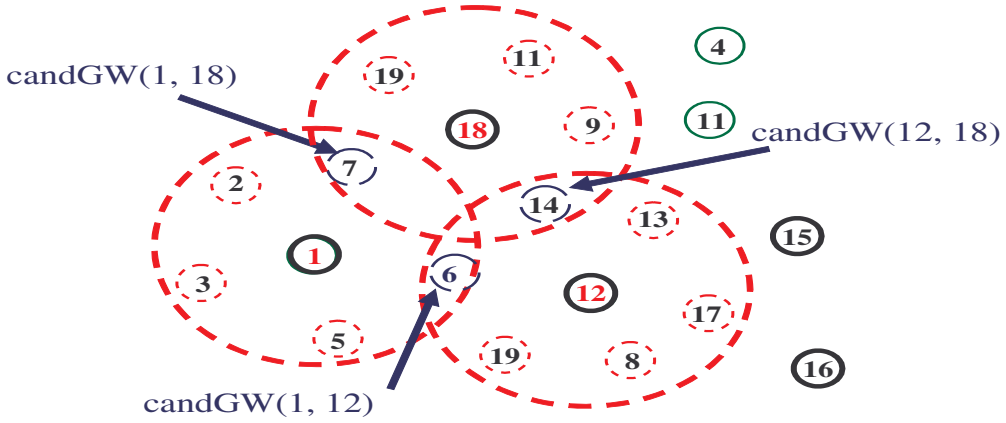


Figure 2.3 – Déclaration d'un candGW

Afin d'éviter de perturber le processus de clusterisation en cours au niveau des couches suivantes, un noeud candGW doit différer l'envoi de sa déclaration d'une période bien définie. Cette dernière sert à éviter les éventuelles collisions des déclarations des gateways avec les envois de *cbeacon* et d'ACK_BT durant la phase de clusterisation des niveaux suivants. Au bout de cette période d'attente, un noeud candGW est autorisé à envoyer sa déclaration qui pourra être correctement reçue par les noeuds clusterheads du niveau suivant sans se heurter à des envois de *cbeacon* ou d'ACK_BT. Le noeud candGW est tenu à spécifier ses clusterheads voisins dans sa déclaration.

Ainsi, un noeud du niveau i doit attendre une période de temps, notée TG_i, pour ne pas corrompre le déroulement du processus de clusterisation aux niveaux des couches i et $i+1$. En effet, sans cette attente, l'envoi de la déclaration de gateway pourra perturber le processus de clusterisation au niveau i en empêchant la bonne réception ou le bon envoi des *cbeacons* des candCHs encore en compétition. Une déclaration immédiate des candGWs pourra aussi brouiller les envois de *cbeacons* des noeuds éligibles du niveau $i+1$ en concurrence pour devenir clusterhead et empêcher la bonne réception de ces *cbeacons*. Le bon déroulement du processus de clusterisation au niveau d'une couche donnée n impose l'interdiction d'envoi de trafic au niveau des couches adjacentes.

Le calcul de cette période d'attente pour un noeud candGW de la couche i se fait grâce à la formule suivante :

$$TG_i = T_{i+2} - t \tag{2.8}$$

Avec t : instant de prise de décision du noeud pour devenir candidat gateway

On considère le noeud candGW k de la figure (2.4). Ce dernier s'est déclaré en tant que candGW suite à l'interception des *cbeacons* provenant des candCHs i et j. Ce candGW doit différer l'envoi de sa déclaration pour devenir GW de la période TG_i afin d'éviter tout risque de collision avec les *cbeacons* de noeuds candCHs de la couche suivante, tel que le noeud l. La période TG_i a été choisie de manière à éviter le risque d'interférence pouvant nuire à la bonne réception des *cbeacons* envoyés par les candCHs du niveau $i+2$, à l'instar du noeud o.

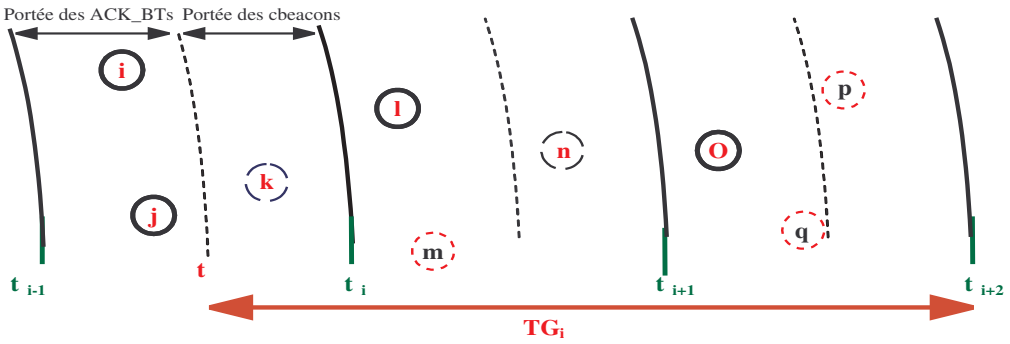


Figure 2.4 – Attente de la période TG_i

A l'expiration de cette période TG_i, les noeuds candGWs d'un même niveau vont entrer en concurrence pour envoyer leur déclaration pour se voir attribuer le rôle gatway. Chacun choisit un délai aléatoire au bout duquel il est autorisé à envoyer sa déclaration et à assurer le rôle de gateway à partir de cet instant précis. Afin de favoriser les noeuds candGWs se trouvant à proximité de plusieurs clusterheads, tout candGW calcule une borne D_2^i personnalisée conformément à la formule suivante :

$$D_2^i = (\frac{D_2}{Nombre\_CH\_voisins}) \tag{2.9}$$

Avec :

D_2^i : La borne calculée par le noeud i

D_2 : La borne maximale imposée pour le calcul du délai aléatoire pour se déclarer gateway

$Nombre\_voisins\_CH$: Le nombre de clusterheads voisins

Cette fonction peut être améliorée en prenant en considération d'autres critères tels que la mobilité, l'énergie restante, etc. Ainsi, tout candGW choisit un délai d_2, ce dernier est un nombre de slot aléatoire entre 0 et la borne personnalisée D_2^i. Étant donné qu'un noeud doit intercepter correctement au moins deux *cbeacons* pour se voir attribuer le statut de candGW, la borne personnalisée D_2^i, calculée par tout candGW, ne peut pas excéder $\frac{D_2}{2}$.

A l'expiration de ce délai d_2, le noeud candGW est autorisé à envoyer sa candidature, s'il est toujours éligible. En effet, un noeud candGW ou gateway, ayant intercepté la déclaration d'un gateway dominant, renonce au rôle de noeud intermédiaire et devient un noeud ordinaire membre. Un gateway dominant couvre plus de clusterheads, incluant les clusterheads voisins du noeud ayant retiré sa candidature, ce qui contribue à minimiser le nombre de gateways. Toutefois, durant l'opération de vérification d'éligibilité au rôle de GW, tout noeud candGW ou GW ne prend en considération que les déclarations provenant de candGWs du même niveau. Grâce à cette règle, on assure la déclaration de suffisamment de gateways pour assurer convenablement la communication entre clusterheads de niveaux adjacents.

2.1.2.2 Déclaration des candGWs avec un CH inconnu

Durant la phase de déclaration des candCHs, on ne peut pas éliminer tout risque d'occurrence de collisions. Une telle collision, dite collision CH, ne peut survenir que lors d'un envoi simultané de deux ou plusieurs *cbeacons* transmis par les noeuds éligibles à assurer le rôle de clusterhead. Il faut noter que de telles collisions peuvent être causées par des noeuds candCHs voisins ou éloignés. De ce fait, on distingue entre deux types de noeuds candGWs. Les noeuds ayant entendu uniquement des *cbeacons* correctement, ces derniers représentent des candGWs normaux. Le deuxième type de candGW regroupe les noeuds ayant intercepté une collision CH.

Par ailleurs, un noeud, ayant intercepté une collision CH durant la première sous période de T_i, ajoute à sa liste de CH un CH inconnu pour indiquer l'existence de deux ou plusieurs clusterheads voisins jusqu'alors inconnus. Toutefois, un tel noeud ne va pas se déclarer comme candidat gateway dans tous les cas, vu que le rôle essentiel des gateways est d'assurer l'interconnexion des clusterheads appartenant à deux niveaux distincts. L'interception d'une telle collision ne peut se produire que durant la première sous période de la phase de clusterisation (T_i), associée à la couche du noeud en question, ou lors de la première sous période de clusterisation du niveau suivant (T_{i+1}). On va essayer d'exposer les différentes situations possibles d'occurrence de collisions CH et expliquer les actions à entreprendre par chaque noeud en considérant son statut actuel.

a. Détection d'une collision CH par un noeud de niveau i durant la première sous période de T_i

Un noeud de niveau i, ayant détecté une collision CH durant la première sous période de T_i, peut avoir l'un de ses 4 états : candCH, noeud N, noeud M ou noeud candGW.

Cas d'un noeud candCH : Un candCH peut détecter une collision CH lors de l'envoi simultané de *cbeacons* par au moins deux autres candCHs avoisinants.

Proposition 1 : Un candCH n'a pas à se déclarer comme candGW en détectant une collision CH

Preuve : Un candCH est, par définition, un noeud apte à devenir un CH car il se trouve à 2 sauts des CHs qui se sont déclarés auparavant au niveau de la couche du niveau précédent. De ce fait, pour devenir un candCH, un tel noeud a nécessairement intercepté un ACK_BT correct ou une collision BT provenant des noeuds membres de la couche précédente. Pour mieux illustrer la situation, on considère l'exemple exposé dans la figure (2.5). Ainsi, la collision CH qui a été interceptée par un noeud candCH m du niveau i, a été causée par l'envoi simultané d'au moins 2 *cbeacons* générés par deux candCHs du même niveau, à l'instar des candCHs l et k. Cette collision serait aussi détectée par au moins un noeud de la couche précédente. En effet, les noeuds l et k sont deux candCHs. Pour se voir attribuer ce statut, ces derniers ont nécessairement intercepté au moins un ACK_BT envoyé par un noeud du niveau précédent ou ont détecté une collision BT due à un envoi simultané d'au moins deux ACK_BTs envoyés par les noeuds membres du niveau antérieur. Les noeuds du niveau antérieur *i-1*, ayant intercepté la même collision CH, pourront se déclarer candGW et concurrencer ultérieurement pour devenir GW et servir les CHs du niveau i.

c.q.f.d

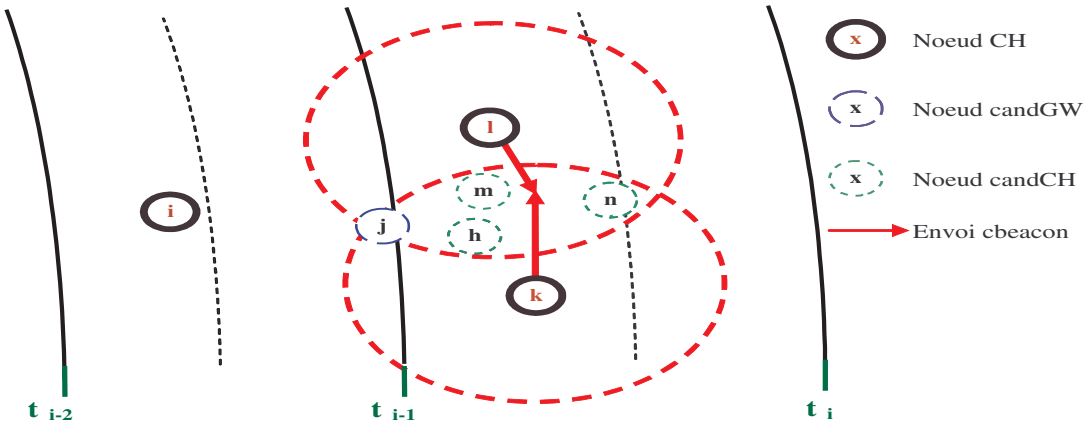


Figure 2.5 – Détection d'une collision par un noeud candCH

Ainsi, un candCH ne se déclare pas comme candGW lors de la détection d'une collision CH, comme prouvé précédemment. Un tel noeud se voit attribuer le statut de noeud membre (M), ajoute un CH inconnu à sa liste de CH et mémorise l'instant d'occurrence de la collision. En effet, une telle collision s'est produite entre au moins deux *cbeacons* provenant de candCHs appartenant à la même couche. Par conséquence, il est inutile d'essayer d'être un intermédiaire pour assurer la communication entre des CHs du même niveau que ce dernier, du moment qu'un candGW du niveau précédent *i-1* va nécessairement se déclarer ultérieurement pour devenir GW.

Par ailleurs, un noeud candCH, s'ayant vu attribué le statut de noeud membre (M) suite à la détection d'une telle collision, est apte à devenir un candGW s'il intercepte ultérieurement

un *cbeacon* ou une collision CH durant la phase de clusterisation de la couche suivante.

Cas d'un noeud N : A l'avènement d'une telle collision, un noeud de niveau i, initialement marqué à N, ajoute un CH inconnu (?) à la liste des clusterheads et mémorise l'instant d'occurrence de la collision. Toutefois, il se déclare comme étant un noeud membre (M), du moment que les clusterheads ayant causé la collision font partie de la même couche. Un tel noeud est apte à devenir un candGW s'il intercepte ultérieurement un *cbeacon* ou une collision CH durant la phase de clusterisation de la couche suivante.

Notre objectif principal est de minimiser le nombre de gateways tout en s'assurant de la déclaration d'assez de candGW pour assurer la communication inter couches. Ainsi, la communication entre deux clusterheads du même niveau peut se faire aisément à travers les gateways du niveau précédent. L'exemple illustré par la figure suivante (2.6) expose cette situation. Les noeuds n, m et h ont intercepté une collision entre les *cbeacons* provenant des CHs nouvellement déclarés l et k. La communication entre les clusterheads l et k peut se faire grâce au noeud j appartenant au niveau précédent T_{i-1} de la clusterisation. Cette tâche pourrait être aussi assurée par tout autre noeud candGW du niveau T_i, ayant entendu au moins deux *cbeacons* correctement durant la phase d'élection des clusterheads au niveau de la couche i ou s'étant déclaré pour jouer le rôle d'intermédiaire avec le niveau suivant.

Par ailleurs, comme le montre l'exemple, pour les réseaux denses, ces collisions CH peuvent être interceptées par plusieurs noeuds du même niveau, tels que les noeuds n, m et h. Si on autorise tous ces noeuds à se déclarer candGW, on va augmenter de manière considérable le nombre de gateways et on ne pourra pas recourir aux règles adoptées pour éliminer les gateways en surplus, vu qu'on n'a aucun critère pour les différencier de manière tranchante.

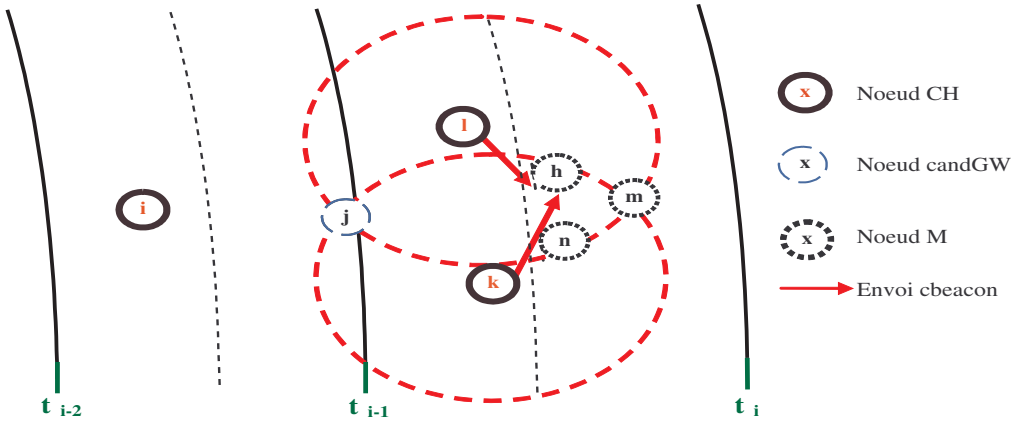


Figure 2.6 – Détection d'une collision par un noeud N

Cas d'un noeud candGW : Lorsqu'un noeud candGW intercepte une collision CH, il ajoute un CH inconnu à sa liste de CH et mémorise l'instant d'occurrence de collision. Ulté-

rieurement, un tel noeud candGW doit indiquer dans sa déclaration qu'il possède un CH inconnu. Il doit spécifier l'instant d'occurrence de la collision afin que tous les noeuds candGWs similaires puissent vérifier leur éligibilité à devenir gateway. Ainsi, les candGWs, ayant intercepté la même collision et couvrant les mêmes clusterheads, renoncent à envoyer leur déclaration et retrouvent le statut de noeud membre.

Cas d'un noeud M : Si le noeud en question est un noeud membre (M), il ajoute un CH inconnu à sa liste de CH et mémorise l'instant d'occurrence de collision. Il préserve son rôle de noeud membre. Il ne pourra se déclarer comme candGW que lorsqu'il reçoit un *cbeacon* ou détecte une collision CH durant la période T_{i+1}.

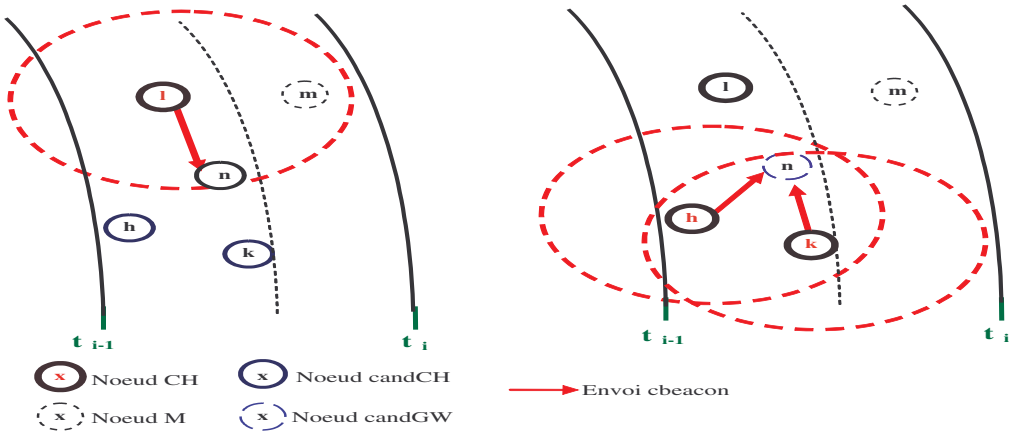


Figure 2.7 – Détection d'une collision par un noeud M

b. Détection d'une collision CH par le noeud de niveau i durant la première sous période de T_{i+1}

Lors de la détection d'une collision CH causée par les envois de *cbeacons* des noeuds candCHs du niveau *i+1*, la clusterisation au niveau i est achevée. Ainsi, une telle collision ne peut être interceptée que par un noeud membre (M) ou un noeud candGW. Un noeud candGW ajoute un CH inconnu à sa liste de CH et mémorise l'instant d'occurrence de la collision. Un noeud membre procède de la même manière tout en se déclarant comme candGW, du moment qu'il se trouve à proximité d'au moins deux clusterheads de niveaux différents, à l'instar du noeud h de l'exemple exposé à travers la figure (2.8).

Par ailleurs, les noeuds candGWs ayant un CH inconnu dans leur liste doivent se déclarer en dernier. A l'instar des candGWs normaux, ils doivent différer l'envoi de leur déclaration de la période TG_i. Par la suite, chacun calcule un délai aléatoire pour l'envoi de sa déclaration en choisissant un nombre de slot aléatoire entre $\frac{D_2}{2}$ et la borne maximale D_2. À l'expiration du délai choisi, un tel candGW envoie sa déclaration s'il est toujours éligible à devenir GW.

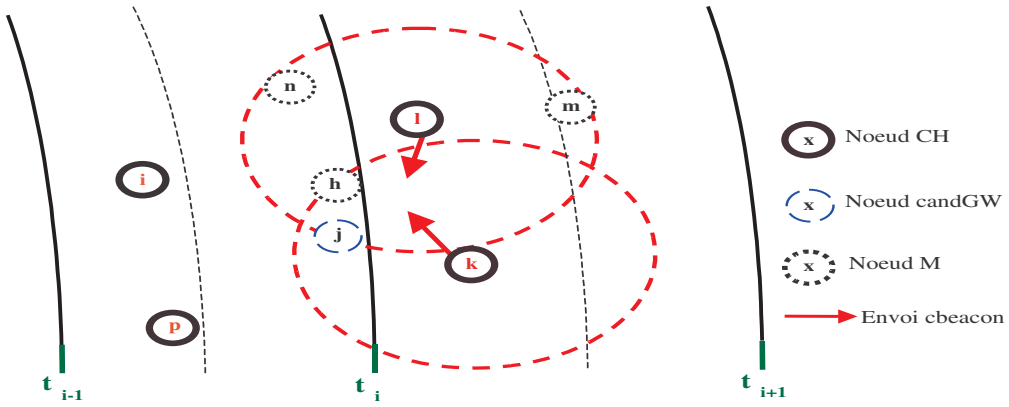


Figure 2.8 – Détection d'une collision CH durant T_{i+1} par un noeud M ou candGW du niveau i

Ces candGWs doivent indiquer le nombre de CH inconnus à proximité et l'instant d'occurrence des collisions associées. Pour chaque collision, on associe un CH inconnu. Grâce à ses informations, les autres candGWs peuvent vérifier s'ils ont intercepté les mêmes collisions en comparant les instants d'occurrence des collisions pour déterminer s'ils se trouvent à proximité des mêmes clusterheads. Ainsi, tout candGW est apte à vérifier son éligibilité à devenir GW. S'il est toujours éligible, il envoie sa déclaration à l'expiration du délai aléatoire choisi. Dans le cas où tous les clusterheads à proximité sont couverts, il renonce à devenir GW et retrouve un statut de noeud membre (M). Les règles adoptées par les candGWs normaux lors de l'étape de vérification d'éligibilité sont aussi appliquées pour ce type de candGW. Ainsi, de tels candGWs ne considèrent que les déclarations provenant des candGWs de même niveau afin d'assurer la couverture de clusterheads de niveau adjacents.

2.1.3 Phase de vérification

Une fois le processus de clusterisation achevé englobant la phase de déclaration des clusterheads et des gateways, ces derniers doivent former un ensemble dominant connexe. En effet, tout noeud du réseau pourra être soit un noeud voisin à un noeud de l'ensemble dominant en tant que noeud membre attaché à un clusterhead, soit un noeud de l'ensemble dominant connexe en s'appropriant le rôle de clusterhead ou de gateway. Cet ensemble aura la charge d'assurer certaines fonctionnalités généralement fournies par l'infrastructure réseau, tel que le routage. De ce fait, on doit s'assurer que cet ensemble est effectivement connexe pour ne pas avoir des répercussions négatives sur le fonctionnement des autres protocoles reposant sur cette structure générée par le processus de clusterisation.

Les deux phases de déclaration de clusterheads et de gateways visent à déterminer les

noeuds clusterheads et les gateways allant assurer la liaison entre ces derniers. Toutefois, dans certaines situations, elles s'avèrent insuffisantes vu le risque d'occurrence de collisions pouvant empêcher la bonne réception des messages *cbeacons* et des déclarations de gateways.

Pour une meilleure compréhension du problème, on considère l'exemple illustré par la figure (2.9). On suppose que la collision engendrée suite à l'envoi simultané des *cbeacons* des candCHs q et l et la collision produite entre les *cbeacons* provenant des noeuds k et n se sont produites au même instant. De ce fait, les noeuds j et h vont se déclarer des candGWs, du moment que la collision CH s'est produite durant la phase de clusterisation du niveau suivant. Chacun de ses deux candGWs mémorise l'instant d'occurrence de la collision, ces deux collisions vont être perçues comme étant une seule collision. Ultérieurement, le candGW j réussit à envoyer en premier sa déclaration pour devenir gateway. A la réception de cette déclaration, le noeud candGW h renonce à devenir GW en s'apercevant que l'instant d'occurrence de la collision indiquée dans la déclaration et celui qu'il a mémorisé sont identiques. Ainsi, il présume qu'il se trouve à proximité des mêmes clusterheads que le noeud gateway j et renonce à candidater au rôle de gateway pour devenir un noeud membre. Or, en renonçant à devenir gateway, les clusterheads l et q vont rester non servis par aucun gateway pour assurer la communication avec les clusterheads du niveau i, vu qu'uniquement le noeud h pouvait assurer cette tâche. La gravité du problème est accentuée, en particulier, dans le cas où aucun gateway du niveau *i+1* ne se déclare ultérieurement pour connecter ces clusterheads avec ceux du niveau *i+2*. Ainsi, la structure de clusterisation ne sera pas connexe, ce qui aura un impact négatif sur le routage et l'acheminement des paquets de données. En effet, ces tâches reposent essentiellement sur les noeuds clusterheads et gateways, l'ensemble dominant formé par ces derniers doit être connexe afin que toutes les destinations soient atteignables en empruntant des chemins n'incluant que les noeuds de l'ensemble dominant.

Afin d'éviter ce problème de déconnexion entre les clusterheads de niveaux adjacents, une phase de vérification s'avère nécessaire. Cette étape est entamée par tous les noeuds clusterheads, les clusterheads n'ayant aucun gateway pour les raccorder aux clusterhads du niveau précédent doivent se manifester. La vérification de la connexité par rapport au niveau précédent nous permet d'éviter de faire un traitement particulier au dernier niveau, contrairement à la vérification de la connexité par rapport au niveau suivant qui nécessite la détection de la dernière couche de clusterisation pour ne pas réaliser cette phase de contrôle de la connexité.

Toutefois, les clusterheads d'un certain niveau i ne vont pas attendre la finalisation de tout le processus de clusterisation pour entamer la phase de vérification, vu qu'on ne peut pas disposer de cette information. Tout clusterhead devra attendre la période nécessaire pour la déclaration des gateways avant de vérifier s'il a au moins un gateway lui permettant de rester connecté au niveau précédent ou pas. S'il ne possède aucun gateway, soit à cause d'une perte d'information suite à des collisions survenues lors de la déclaration des gateways du niveau

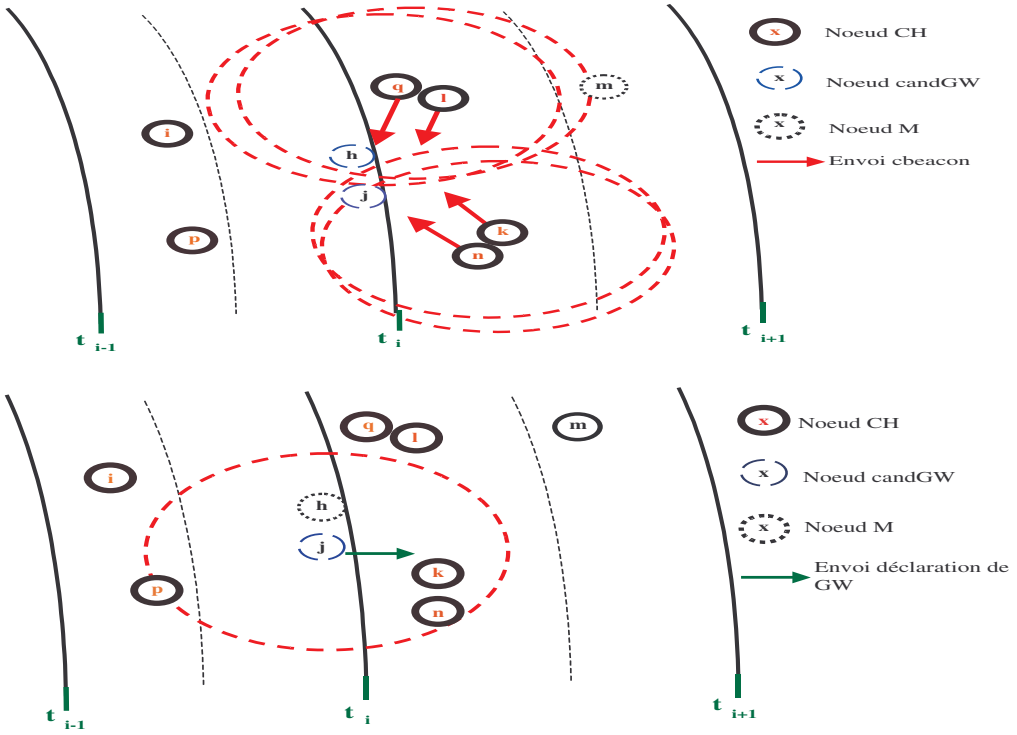


Figure 2.9 – Cas de de CHs de niveaux i et $i+1$ déconnectés

précédent, soit suite à une erreur de jugement sur l'éligibilité d'un candGW, il ré envoie un *cbeacon* pour pousser des noeuds membres du niveau précédent à se déclarer comme gateway ou obliger des gateways du niveau précédent à se manifester. Ainsi, en interceptant ce *cbeacon*, un noeud GW du niveau précédent devra ré envoyer sa déclaration. Tandis qu'un noeud membre se déclare comme candGW et calcule un délai aléatoire pour envoyer sa déclaration comme gateway. Cette phase de déclaration de gateways subit les mêmes règles que la précédente. On note que les noeuds déjà gateways seront favorisés lors du ré envoi de leur déclaration afin d'éviter des déclarations de noeuds supplémentaires parfois inutiles.

Par conséquence, tout noeud clusterhead devra calculer, au moment de sa déclaration comme CH, le temps nécessaire à attendre avant d'effectuer cette vérification de la connexité. Cette période d'attente de tout clusterhead, notée **P_attente**, doit inclure la période TG_{i-1}, nécessaire à tout candGW du niveau précédent avant de pouvoir concurrencer pour devenir gateway, Toutefois, un clusterhead de niveau i ne doit pas aussi nuire et empêcher le bon déroulement des déclarations faites par les candGWs du niveau i. Pour cette raison, on doit prendre en considération aussi la période d'attente des candGWs du niveau i.

On considère l'exemple illustré par la figure (2.10). Lors de la déclaration du noeud m comme étant un clusterhead durant la période T_{i+1}, le noeud k se voit attribuer le statut de

candGW. Toutefois, pour pouvoir envoyer sa déclaration, il devra la différer de la période d'attente $TG_i = T_{i+2}$ - t_1, t_1 étant l'instant de déclaration du noeud m comme CH permettant au noeud k d'acquérir le rôle de candGW. Or, à proximité du clusterhead m, il peut y avoir des candGWs du même niveau *i+1*. Ces derniers pourront ultérieurement concurrencer pour devenir gateway et jouer le rôle d'intermédiaire avec les clusterheads du niveau suivant *i+2*, à l'instar du noeud n qui a eu ce statut suite à la déclaration du CH o du niveau *i+2*. De ce fait, le noeud m devra prendre en considération aussi l'existence de ces candGWs pour ne pas perturber le bon envoi de leur déclaration ultérieurement. Par ailleurs, cette période **P_attente** doit aussi

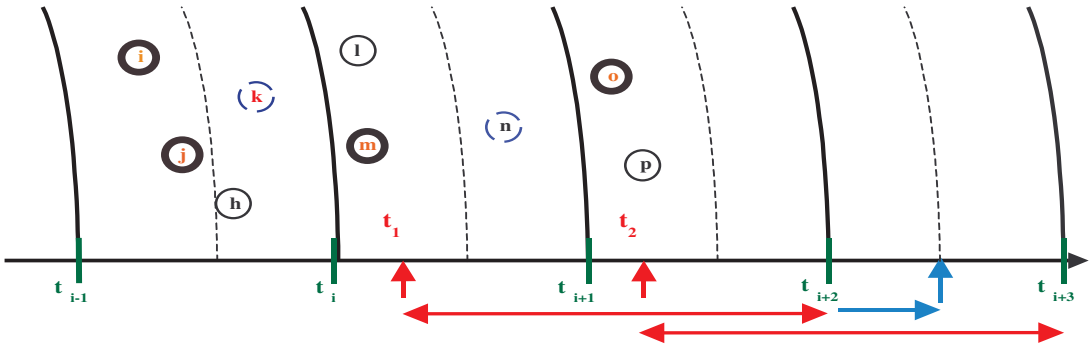


Figure 2.10 – Exemple explicatif de la période d'attente d'un CH

prendre en compte le temps nécessaire à la déclaration des gateways se trouvant à proximité du clusterhead en question. Ainsi, on doit calculer le temps nécessaire à la déclaration du nombre maximum de gateways se trouvant à proximité d'un clusterhead de niveau i, ces derniers peuvent appartenir au même niveau i ou au niveau précédent *i-1*. Le calcul de la période **P_attente** pour un noeud clusterhead du niveau i devra inclure le temps d'attente avant l'initiation de la phase de déclaration des candGWs du niveau *i-1* et du niveau i et le temps nécessaire pour la déclaration du nombre maximum de candGWs se trouvant à proximité du CH en question. On fait remarquer que l'envoi des déclarations de candidature pour s'attribuer le rôle de gateway est conforme à la procédure de backoff normal. Tout noeud ne pourra pas choisir un délai aléatoire supérieur à CWsize.

La détermination de la période **P_attente** par tout clusterhead doit se faire de manière à éviter le problème de collisions et d'interférences pouvant survenir lors de la manifestation de deux CHs non couverts appartenant à deux niveaux adjacents. Le déroulement de la phase de vérification au niveau de deux couches adjacentes à des instants rapprochés peut nuire soit à la bonne réception des *cbeacons* envoyés par les CHs non couverts, soit à la bonne réception des déclarations des gateways. Ainsi, le calcul de **P_attente** se fait conformément à la formule (2.10) pour tout noeud CH de niveau i impair, tandis que les CHs appartenant à un niveau i

pair devront adopter la formule (2.11) :

$$P\_attente = (T_{i+2} - t) + (CWsize + (NbrMax\_gw\_CH * T_{candGW})) \quad (2.10)$$

$$P\_attente = (T_{i+2} - t) + (CWsize + (NbrMax\_gw\_CH * T_{candGW})) + (3 * T_i) \quad (2.11)$$

Avec :

t : Instant de déclaration du CH en question

T_{candGW} : Temps nécessaire pour la transmission d'une déclaration de gateway

CWsize : Taille maximale de la fenêtre de contention en nombre de slot de temps

$NbrMax\_gw\_CH$: Nombre maximum de gateway pouvant se déclarer à proximité d'un CH

Or, pour garder une certaine synchronisation entre les noeuds en terme de périodes T_i, on oblige tout clusterhead à effectuer cette étape de vérification au début d'une période T_i. Pour atteindre cet objectif, on va exprimer aussi le temps nécessaire à la phase de déclaration des candGWs en nombre de période T_i. Le calcul du nombre de période T_i requis pour cette étape est comme suit :

$$Nbr\_T_{iGW} = \lceil (\frac{CWsize + (NombreMax\_gw/CH * T_{candGW})}{T_i}) \rceil \quad (2.12)$$

Avec :

T_i : Période nécessaire à la clusterisation du niveau i

Nbr_T_{iGW} : Nombre de périodes T_i nécessaires pour la déclaration des candGWs se trouvant à proximité du noeud CH ayant fait la vérification

$NombreMax\_gw/CH$: Nombre maximal de noeuds pouvant réagir au *cbeacon* envoyé par le clusterhead pour le rôle de gateway

On rappelle que toute période T_i est déterminée comme suit :

$$T_i = SIFS + CWsize + Txcbeacon + SIFS + TxBT \quad (2.13)$$

Par conséquence, la formule de **P_attente** relative à un noeud CH de niveau i impair sera modifiée comme suit :

$$P\_attente = (T_{i+2} - t) + (Nbr\_T_{iGW} * T_i) \quad (2.14)$$

La formule de **P_attente** associée à un noeud CH de niveau i pair sera changée comme suit :

$$P\_attente = (T_{i+2} - t) + (Nbr\_T_{iGW} * T_i)) + (3 * T_i) \quad (2.15)$$

Une fois la période **P_attente** expire, tout clusterhead entame la phase de vérification de connexité. En détectant une déconnexion par rapport au niveau précédent, il ré émet un *cbeacon*. Au début de la période T_i suivante, les gateways du niveau précédent, ayant intercepté ce paquet, devront calculer un délai aléatoire en prenant $\frac{D_2}{2}$ comme borne maximale pour le choix du nombre de slots avant de ré émettre leur déclaration. Alors que les noeuds membres du niveau *i-1*, ayant acquis le statut de candGW suite à l'interception de ce paquet, devront choisir un délai aléatoire entre $\frac{D_2}{2}$ et D_2 pour pouvoir transmettre leur candidature. Ainsi, on privilégie les noeuds gateways déjà existants et on contribue à minimiser le nombre de gateway tout en essayant de maintenir l'ensemble des clusterheads et gateways connexe. Ainsi, une période T_i est nécessaire pour la ré émission des *cbeacons* des clusterheads non servis par de gateways du niveau antérieur. Il faut prévoir aussi un nombre suffisant de périodes T_i pour la manifestation des gateways existants ou la déclaration de nouveaux gateways.

Pour cela, le temps **Tphase_verif** nécessaire à la vérification inclura une période T_i pour la manifestation de CHs non servis, suivie par une période pour l'envoi de déclarations de gateways déjà existants ou des nouveaux noeuds gateways supplémentaires. Cette dernière est exprimée aussi en nombre de périodes T_i. Le calcul de **Tphase_verif** sera comme suit :

$$Tphase\_verif = T_i + (Nbr\_T_{iGW}\_sup * T_i)) \tag{2.16}$$

La variable Nbr_T_{iGW}_sup correspond au nombre de périodes T_i nécessaires pour la déclaration du nombre moyens de GW pouvant se manifester suite au rappel fait par le CH non servi. Cette dernière est déterminée comme suit :

$$Nbr\_T_{iGW}\_sup = \lceil (\frac{(CWsize + (NombreMoy\_gw/CH * T_{candGW}))}{T_i}) \rceil \tag{2.17}$$

Avec :

$NombreMoy\_gw/CH$: Nombre moyen de gateway pouvant ré envoyer leur déclaration ou se déclarer suite au rappel fait par un CH non servi du niveau suivant

Pour s'assurer de la déclaration d'un gateway, tout clusterhead non servi redéclenche le temporisateur relatif à **P_attente** au début de la période de vérification. Ce temporisateur sera réglé à la valeur de **Tphase_verif**. A l'expiration de ce dernier, le clusterhead re vérifie sa connexité par rapport au niveau antérieur. En cas de déclaration ou de manifestation de gateways durant **Tphase_verif**, le clusterhead peut reprendre son fonctionnement normal. Dans le cas contraire, le clusterhead peut réitérer ce processus jusqu'à n fois, n étant égale à $NombreMoy\_gw/CH$.

2.1.4 Maintenance

Le principal apport de notre approche réside dans son aptitude à fournir rapidement une infrastructure virtuelle pouvant être déployée immédiatement par les protocoles de haut niveau, les

résultats présentés dans la sous section (3.1) le confirmeront. Cette caractéristique nous permet de suspendre temporairement l'envoi des paquets de données durant la phase de clusterisation sans pénaliser les performances du système. Cette suspension temporaire relative aux paquets de données permet de garantir le bon déroulement du processus de clusterisation en évitant les collisions éventuelles entre les paquets de données et les paquets de contrôle dédiés à la clusterisation. De telles collisions peuvent nuire au bon fonctionnement de processus de clusterisation et engendrer des retransmissions inutiles des paquets de données.

L'achèvement rapide de la phase de clusterisation nous donne aussi le privilège de relancer le processus de clusterisation fréquemment sans dégrader les performances du système. Relancer le processus de clusterisation périodiquement pour assurer la maintenance de l'ensemble dominant connexe nous permet de prendre en considération des informations actualisées (le voisinage immédiat, l'énergie résiduelle au niveau de chaque noeud, ...) lors de la prise de décision sur le rôle à assurer.

Toutefois, afin d'éviter de reconstruire à zéro l'ensemble dominant connexe et d'assurer une certaine stabilité à l'infrastructure virtuelle générée par le processus de clusterisation, un mécanisme additionnel a été défini. Ce dernier autorise la ré élection des anciens clusterheads pouvant encore assumé ce rôle. Par ailleurs, un noeud dominant, en tant que CH ou GW, assure des fonctionnalités additionnelles pouvant contribuer à épuiser rapidement ses ressources. Pour ce fait, on a limité le nombre maximal de périodes de clusterisation consécutives lors de la ré élection des clusterheads afin d'assurer la rotation du rôle de clusterhead de manière équitable entre les différents noeuds. Pour cet effet, l'ajout de deux nouvelles bornes D_{1pmin} et D_{1npmin} à utiliser lors de la phase d'élection des clusterheads a été nécessaire.

Lors de la première exécution du processus de clusterisation, tous les candCHs utilisent la formule (2.1) pour le calcul de la borne personnalisée pour avoir la même chance d'acquérir le rôle de clusterhead. Durant les phases de clusterisation ultérieures, un noeud candCH calcule sa borne personnalisée D_1^i en ayant recours à la formule (2.18), s'il a déjà assuré le rôle de clusterhead durant la période antérieure. Toutefois, tout candCH ne doit pas excéder le nombre maximal de périodes de clusterisation successives en tant que clusterhead.

$$D_1^i = \lceil ((\frac{(D_{1pmin} - D_{1max})}{E_{max}}) * E_i) + D_{1max} \rceil \tag{2.18}$$

Les autres noeuds candCHs déterminent leur borne personnalisée grâce à la formule (2.1) tout en s'assurant que le nombre de slots choisi d_1 est supérieur ou égal à la borne D_{1pmin}. Le recours à cette nouvelle borne nous permet de privilégier les anciens clusterheads aptes à assumer encore les responsabilités assignées à un noeud dominant. Ainsi, on assure une certaine stabilité à l'infrastructure virtuelle tout en offrant une structure globale homogène.

Un noeud, ayant assuré le rôle de clusterhead durant n périodes de clusterisation successives, doit perdre le privilège qui lui est accordé. Cette distinction sera résiliée lorsque n atteint le seuil

maximal de périodes de clusterisation successives en tant que clusterhead. Un tel candCH devra adopté la formule (2.19) pour le calcul de sa borne personnalisée D_1^i, tout en s'assurant que le nombre aléatoire de slot choisi d_1 est supérieur ou égal à la borne D_{1min}. Agissant de la sorte, on assure une certaine rotation équitable des rôles des noeuds dominants (clusterhead ou gateway) entre l'ensemble des noeuds formant le réseau.

$$D_1^i = \lceil ((\frac{(D_{1npmin} - D_{1max})}{E_{max}}) * E_i) + D_{1max} \rceil \tag{2.19}$$

2.1.5 Preuves

La preuve que l'algorithme se termine et qu'il est exacte sera sur plusieurs étapes. On suppose que le réseau reste toujours connexe, même en cas de mobilité.

Proposition 1 : Tout noeud va forcément entendre un *cbeacon* ou un ACK_BT.

Preuve : D'après le principe de notre algorithme, on vise à permettre aux noeuds se trouvant à deux sauts d'un clusterhead d'un niveau i de candidater pour devenir clusterhead durant la phase de clusterisation du niveau suivant. De ce fait, au sein d'une même couche, on regroupe les clusterheads qui ont réussi à se déclarer et les noeuds membres, initialement marqués à N ou des anciens candCHs, ayant intercepté une collision CH durant la première sous période de T_i et/ou ayant reçu correctement au moins un *cbeacon*. Le processus de clusterisation évolue par couche en commençant par la première formée par le DN et les noeuds avoisinants. Ces derniers à leur tour informent ceux qui se trouvent à 2 sauts du DN à travers l'envoi d'un ACK_BT qu'ils peuvent initier la phase de clusterisation au niveau de la 2^{me} couche. Cette couche inclura les clusterheads déclarés et les noeuds ayant interceptés au moins un *cbeacon* et/ou une collision CH. Les noeuds membres et candGW auront la tâche d'informer les noeuds de la couche suivante, et ainsi de suite. De cette manière, l'information se propage de proche en proche et le nombre de noeuds non couverts diminue à chaque couche jusqu'à la couverture de tous les noeuds du réseau.

Ainsi, étant donné qu'un réseau ad hoc est modélisé par un graphe connexe, il existe forcément un chemin entre tout noeud x du réseau et le noeud DN. Si on considère que la distance séparant le noeud x du DN en nombre de sauts est $d(DN, x) = k$, alors la clusterisation ne va atteindre le noeud x que durant la $(\lfloor(\frac{k}{2}) + 1\rfloor)^{me}$ période T_i. A chaque période T_i, on élimine 2 types de noeuds, les clusterheads et les noeuds directement rattachés à eux. En d'autres termes, on supprime 2 sauts sur le chemin séparant x du DN, exception faite pour le niveau 1 où on n'élimine qu'un seul saut. De ce fait, le noeud x interceptera soit un ACK_BT ou une collision BT pour devenir éligible à concurrencer pour le rôle de clusterhead, soit au moins un *cbeacon* et/ou une collision CH pour acquérir le statut de noeud membre ou de candidat gateway.

Dans le cas (a) de l'exemple suivant (fig. 2.11), on illustre le cas où le noeud x est éloigné

d'un nombre de sauts impair du noeud DN, égal à 5. En appliquant la formule, on retrouve que la clusterisation va atteindre le noeud x au cours de la 3^{me} période T_i. Le cas (b) expose la situation où le nombre de sauts est pair, pour cet exemple 6 sauts. Par conséquence, le noeud x participera à la phase de clusterisation au cours de la 4^{me} période T_i. Dans l'exemple exposé, les noeuds en pointillés rouge correspondent aux clusterheads, tandis que les noeuds verts sont des candidats pour le rôle de gateway.

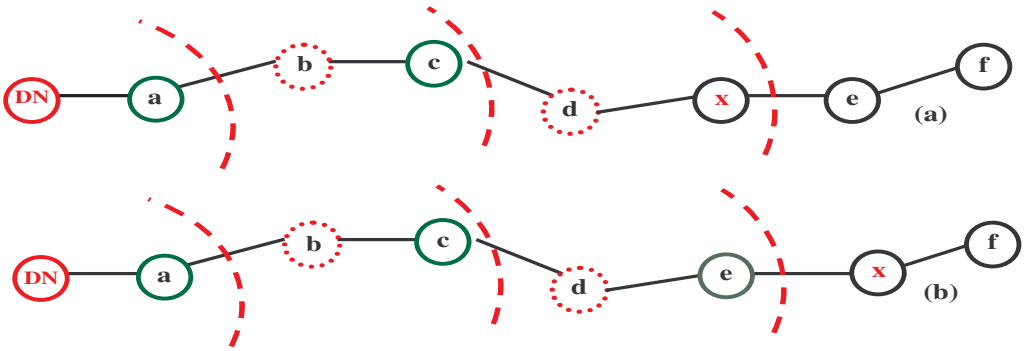


Figure 2.11 – Illustration de l'éloignement d'un noeud x du DN

De manière générale, le diamètre représente le plus long chemin entre 2 noeuds du réseau. Pour déterminer le temps nécessaire pour finaliser la phase d'envoi des *cbeacons*, noté $T_{cbeacon}$, on considère *long* la distance séparant le noeud DN et le noeud le plus éloigné par rapport à ce dernier, exprimée en nombre de sauts. En effet, le noeud DN constitue le repère à partir duquel débute le processus de clusterisation. Ainsi, le nombre de périodes T_i nécessaires pour finaliser cette étape, noté $Nb\_T_{cbeacon}$, est déterminé comme suit :

$$Nb\_T_{cbeacon} = \lfloor \frac{long}{2} \rfloor + 1 \tag{2.20}$$

Ce nombre de périodes T_i peut être borné, vu que dans le pire des cas, le noeud DN se trouve à l'extrémité du graphe. Ainsi, *long* sera assimilé au diamètre du réseau. Alors que pour le meilleur des cas, le noeud DN se trouvera au centre du réseau, donc *long* correspondra au $\frac{diamtre}{2}$.

$$\lfloor \frac{diametre}{4} \rfloor + 1 \leq T_{cbeacon} \leq \lfloor \frac{diametre}{2} \rfloor + 1 \tag{2.21}$$

Toutefois, on doit prendre en compte le fait que la première période T_1 subit un calcul différent des autres périodes T_i lors du calcul de $T_{cbeacon}$:

$$T_{cbeacon} = (\lfloor \frac{long}{2} \rfloor * T_i) + T_1 \tag{2.22}$$

c.q.f.d

Proposition 2 : L'ensemble formé par les clusterheads et les gateways forme un ensemble dominant connexe.

Preuve : Le rôle principal des gateways est d'assurer la communication entre clusterheads de couches adjacentes. Toutefois, on n'exclut pas la déclaration de gateways pour jouer le rôle d'intermédiaire entre au moins deux clusterheads du même niveau, cette déclaration est autorisée lorsque le noeud en question reçoit correctement des *cbeacons* de ses voisins clusterheads durant la phase de clusterisation de sa couche. Dans ce qui suit, on va exposer les cas possibles pour acquérir le statut de candGW et pouvoir par la suite concurrencer pour devenir gateway.

Le premier type de candGW représente les candGW normaux n'ayant intercepté que des *cbeacons* corrects. Pour le traitement de ces derniers, il n'y a aucune ambiguïté, vu que chaque candGW connaît ses clusterheads voisins. Ainsi, en les spécifiant dans la déclaration transmise, les autres candGWs similaires sont aptes à vérifier leur éligibilité de manière tranchante. Les candGWs, ayant réussi à se déclarer, peuvent jouer le rôle d'intermédiaire entre des clusterheads d'un même niveau ou des clusterheads de couches adjacentes. Toutefois, une règle a été adoptée pour minimiser le nombre de gateways, elle oblige les candGWs ou GWs ayant reçu une déclaration d'un GW dominant, de même niveau, à se retirer de la concurrence ou à renoncer à ce statut.

Le deuxième type de candGW regroupe les noeuds ayant intercepté au moins une collision CH. Toutefois, l'interception d'une collision CH n'implique pas directement la déclaration du noeud comme étant un candGW. En effet, si le noeud est un noeud membre (M), il ne se voit attribuer le statut de candGW que si la collision s'est produite durant la phase de clusterisation de la couche suivante. De cette manière, on n'autorise que la candidature de noeuds se trouvant à proximité de clusterheads de couches adjacentes. Si le noeud est déjà un candGW, alors soit il a intercepté correctement au moins deux cbeacons, remplissant ainsi la règle pour avoir le statut de candGW, soit il était un noeud M ayant intercepté une collision CH durant la clusterisation du niveau suivant. Ainsi, on ne tolère pour ce type de candGW que la candidature de noeuds pouvant assurer la communication inter couches. Une phase de vérification a été ajoutée afin de remédier aux erreurs de vérification de l'éligibilité pour ce type de candGW, vu que de tels candGWs ne disposent pas d'information exacte concernant les clusterheads avoisinants. Cette étape de vérification impose à tout clusterhead n'ayant pas de gateway le raccordant au niveau antérieur de ré envoyer son *cbeacon* afin d'inciter les noeuds du niveau précédent soit à se manifester s'ils sont déjà gateway soit à candidater s'ils sont des noeuds membres.

c.q.f.d

Proposition 3 : Le temps de clusterisation est O(d), d étant le diamètre du réseau.

Preuve : On pose T_{clus} le temps nécessaire pour la finalisation du processus de clusterisation pour un réseau donné, ce temps comprendra :

- Le temps nécessaire pour finaliser la phase d'envoi des *cbeacons*, on doit prendre en compte le fait que la première période T_1 subit un calcul différent des autres périodes T_i ;
- Le temps nécessaire pour la déclaration de candGW de la dernière couche. En effet, chaque candGW du dernier niveau devrait attendre deux périodes T_i avant de candidater pour devenir gateway ;
- Le temps nécessaire à la vérification de la dernière couche.

Ainsi, T_{clus} s'exprime comme suit :

$$T_{clus} = T_{cbeacon} + P\_attente + Tphase\_verif \tag{2.23}$$

D'après la sous section (2.1.3), le calcul de la période **P_attente** se fait différemment selon le niveau du noeud CH non couvert. La formule adoptée diffère pour les noeuds CHs de niveau pair de ceux appartenant à un niveau impair. Ainsi, le calcul de T_{clus} se fera conformément à la formule (2.24) si le niveau de la dernière couche est impair :

$$\begin{aligned} T_{clus} &= (\lfloor \frac{long}{2} \rfloor * T_i) + T_1 + ((2 * T_i) + (Nbr\_T_{iGW} * T_i)) \\ &\quad + (T_i + (Nbr\_T_{iGW}\_sup * T_i)) \end{aligned} \tag{2.24}$$

Dans le cas contraire, on utilise la formule (2.25) si le niveau de la dernière couche est pair :

$$\begin{aligned} T_{clus} &= (\lfloor \frac{long}{2} \rfloor * T_i) + T_1 + ((2 * T_i) + (Nbr\_T_{iGW} * T_i)) + (3 * T_i)) \\ &\quad + (T_i + (Nbr\_T_{iGW}\_sup * T_i)) \end{aligned} \tag{2.25}$$

Dans le pire des cas, la distance *long* sera assimilée au diamètre du réseau. Ainsi, d'après la formule (2.21), on obtient la formule (2.26) si le niveau de la dernière couche est impair et la formule (2.27) si le niveau de la dernière couche est pair :

$$T_{clus} = (((\lfloor \frac{diametre}{2} \rfloor) + 3 + Nbr\_T_{iGW} + Nbr\_T_{iGW}\_sup) * T_i) + T_1 \tag{2.26}$$

$$T_{clus} = (((\lfloor \frac{diametre}{2} \rfloor) + 6 + Nbr\_T_{iGW} + Nbr\_T_{iGW}\_sup) * T_i) + T_1 \tag{2.27}$$

Ainsi, la valeur de T_{clus} dépend étroitement du diamètre du réseau et non pas du nombre de noeuds formant ce dernier, étant donné que le diamètre varie d'un réseau à un autre et toutes les autres variables sont fixes. Lors du calcul de T_{clus}, on n'a considéré qu'une seule vague de vérification. Toutefois, un clusterhead, non servi après la première phase de vérification, peut réitérer le même processus pour inciter à nouveau les noeuds membres et les gateways du niveau antérieur.

On note que tout noeud n'est pas obligé d'attendre la finalisation complète du processus de clusterisation pour tout le réseau pour pouvoir relancer l'envoi des paquets de données suspendu durant la phase de clusterisation. Cette interruption momentanée du trafic de données a été

adoptée afin que la phase de clusterisation finisse rapidement et afin d'éviter toute collision entre les paquets de contrôle dédiés au processus de clusterisation et les paquets de données. Ces éventuelles collisions peuvent engendrer des retransmissions inutiles et empêcher le bon déroulement du processus de clusterisation.

Par ailleurs, tout noeud, ayant été atteint par la phase de clusterisation, est apte à estimer le nombre de périodes T_i à attendre avant de reprendre la transmission des données en attente, noté $T_{reprise}$. Tout noeud de niveau i, indépendamment de son statut, devra attendre 2 périodes dédiées à l'attente des candGWs du niveau i, suivies par le nombre de périodes nécessaires à la déclaration de ces derniers et par la phase de vérification réalisée par les clusterheads du niveau i incluant une période d'annonce *cbeacon* par les clusterheads non servis et une période de déclaration pour les gateways allant répondre à ce rappel. Ainsi, $T_{reprise}$, exprimé en nombre de périodes T_i, se calcule selon la formule (2.28) pour un noeud de niveau i impair, étant donnés que les clusterheads du même suivant pair doivent différer leur étape de vérification de $3 * T_i$ afin d'éviter la perturbation du déroulement de cette phase pour le niveau antérieur. Le calcul sera conformément à la formule (2.29) pour un noeud de niveau i pair, étant donné qu'un tel noeud réagisse à la réception de *cbeacon* provenant d'un niveau impair.

$$T_{reprise} = 3 + Nbr\_T_{iGW} + Nbr\_T_{iGW}\_sup + 4 \tag{2.28}$$

$$T_{reprise} = 3 + Nbr\_T_{iGW} + Nbr\_T_{iGW}\_sup + 1 \tag{2.29}$$

Par conséquence, un noeud de niveau i pourra reprendre son fonctionnement normal après $T_{reprise}$ périodes T_i.

c.q.f.d

2.2 Délégation du rôle de DN

Le recours à la clusterisation avait pour objectif de sélectionner un ensemble dominant de noeuds qui vont offrir une sorte d'infrastructure virtuelle pouvant assurer certaines fonctionnalités, telles que le routage et l'acheminement des données. Ainsi, les noeuds clusterheads et gateways auront des tâches additionnelles, ce qui contribue à accélérer l'épuisement de leur ressource. Le noeud DN, étant le premier CH à se déclarer, aura en plus la tâche de déclencher le processus de clusterisation de manière périodique afin de prendre en considération les éventuels déplacements des différents noeuds et l'état actuel du réseau. Relancer périodiquement le processus de clusterisation permet de maintenir par conséquence la structure générée par la clusterisation actualisée et toujours connexe. Pour cette raison, on doit toujours disposer d'un noeud DN ayant suffisamment de ressources. Une perte subite de ce noeud engendre une faille dans l'ensemble dominant connexe, formé par l'ensemble des clusterheads et gateways. Cette

déconnexion, due à l'extinction du noeud DN, pourrait nuire au fonctionnement des autres algorithmes qui reposent sur la connexité de cet ensemble devant couvrir la totalité du réseau. Par conséquence, trouver toujours un chemin à travers cette structure reliant entre toute paire de noeuds ne sera pas toujours valide. Par ailleurs, le départ imprévu du noeud DN empêche le re-lancement du processus de clusterisation périodiquement.

Pour cet effet, une procédure de délégation du rôle de DN a été mise en place afin de permettre à un noeud DN ayant épuisé ses ressources et ne pouvant plus assurer son rôle de passer la main à un successeur apte à assurer la relève. Ainsi, un tel DN doit choisir un noeud pour le remplacer parmi les noeuds GWs se trouvant à proximité, une fois le processus de déclaration de gateways achevé. Le choix du successeur peut reposer sur plusieurs critères, tels que l'énergie ou la mobilité ou même une combinaison de plusieurs critères.

Ainsi, le noeud DN devra attendre la finalisation de la phase de déclaration des gateways afin de recueillir les déclarations de ces derniers. Ces déclarations vont inclure certaines informations, à l'instar de la quantité d'énergie disponible au niveau du candidat, afin d'aider le DN à effectuer son choix. Le noeud DN aura à estimer le temps à attendre conformément à la formule (2.14) correspondant à la période **P_attente**. Une fois cette attente finie, le DN envoie un message de délégation au GW choisi. Il n'aura pas à calculer un délai aléatoire avant l'envoi de son message, étant donné l'existence d'un seul DN. Il aura uniquement à attendre une période d'écoute SIFS avant la transmission du message. A la réception d'un tel message, le gateway élu devra envoyer un accusé de réception au noeud DN après une période SIFS, s'il a accepté de prendre la relève. Le noeud DN doit aussi à son tour acquitter l'acceptation reçue de la part du GW. Cette délégation s'effectue grâce à trois échanges afin de s'assurer que les deux entités communicantes aient bien reçu les messages respectifs.

Ainsi, au début de la prochaine phase de clusterisation, le noeud DN renonce de manière effective à son rôle. Le noeud GW élu pourra initier ses nouvelles fonctionnalités et se voir attribuer le statut de DN. Dans le cas contraire, le noeud DN devra choisir un autre candidat.

2.3 Conclusion

Dans ce chapitre, nous avons proposé un nouveau mécanisme `TBCA` pour la génération d'un ensemble dominant connexe tout en tenant compte et exploitant les spécificités d'un environnement sans fil. L'organisation du processus de clusterisation en couches permet de réduire le nombre de candidats éventuels pour le rôle de clusterhead à un instant donné, réduisant par conséquence le risque de collision entre paquets de contrôle dédiés à cette phase. L'exploitation de telles situations de collision nous a permis d'achever rapidement le processus de clusterisation. Cette caractéristique nous autorise à suspendre momentanément le trafic de données durant la clusterisation du réseau, puis reprendre le traitement des paquets. Notre principal objective est

de réduire le trafic de contrôle afin d'améliorer le passage à l'échelle et de permettre des économies d'énergie. Opérant de la sorte, on ne pénalise pas les performances du réseau et on est apte à générer rapidement une dorsale pour supporter efficacement les protocoles de haut niveau. La quantifiaction des apports de notre mécanisme fera l'objet du chapitre suivant.

Chapitre 3

Evaluation de performances de notre algorithme de clusterisation

Ce chapitre présente les résultats des simulations entamées pour évaluer les performances de notre nouveau mécanisme. Pour valider notre algorithme et mettre en évidence ses apports, une implémentation sous le simulateur J-SIM [61, 72] s'est avérée primordiale à travers la modification de la couche MAC. Pour se comparer, le choix a été porté sur le protocole `SPAN` [23]. Ce dernier est intégré aussi au niveau MAC et opère dans un environnement synchrone, à l'instar de notre algorithme `TBCA`. Nous avons été aussi amenés à implémenter le protocole `SPAN` sous le même simulateur pour pouvoir comparer les deux protocoles.

On rappelle que le mécanisme `SPAN` repose sur l'élection d'un ensemble dominant connexe de manière distribuée à travers l'échange périodique de messages Hello. En se basant sur ces messages Hello reçus, chaque noeud est apte à vérifier son éligibilité à devenir coordinateur localement. Toutefois, ce trafic supplémentaire, nécessaire au maintien de l'ensemble dominant connexe, est proportionnel au nombre de noeuds du réseau. En outre, il va puiser des ressources partagées et entrer en concurrence avec le trafic de données pour accéder au canal. Cette concurrence entre ces deux types de trafic contribue à augmenter le risque de collisions engendrant des pertes d'informations (Hello) et les retransmissions inutiles pour les paquets de données. Ainsi, l'échange des messages de contrôle additionnels peut pénaliser le trafic de données, et peut nécessiter une longue période pour certains algorithmes pour la collecte des informations nécessaires à cette phase.

Notre algorithme de clusterisation a été conçu dans l'objectif d'offrir une structure de clusterisation rapidement pouvant servir comme appui à d'autres algorithmes de routage ou de conservation d'énergie. Dans ce contexte, notre algorithme essaie d'organiser le processus de clusterisation en couches afin d'accélérer la terminaison de ce dernier. Grâce à cette caractéristique, la suspension du trafic de données sera momentanée et courte afin d'éviter les éventuelles collisions entre les paquets de données et les paquets de contrôle dédiés à la clusterisation. Ainsi,

on minimise l'impact de ce trafic additionnel consommant des ressources en évitant les retransmissions inutiles des paquets de données et la perturbation du bon déroulement du processus de clusterisation. Pour mettre en valeur les apports de notre mécanisme, on va focaliser sur les critères suivants : le temps moyen de finalisation du processus de clusterisation, le nombre moyen de noeuds dominants, l'impact du trafic additionnel dédié au processus de construction et de maintenance de l'ensemble dominant et l'impact de la mobilité sur les performances du réseau.

Pour des raisons évidentes, seuls les réseaux connexes sont considérés lors des simulations. Durant toutes les expérimentations conduites, on assume que la portée maximale du signal est la même pour tous les noeuds du réseau, cette portée étant égale à 250m. Deux noeuds sont considérés comme voisins s'ils se trouvent respectivement dans la portée du signal l'un de l'autre. La bande passante est de 2 Mbps. Les résultats présentés tout au long de cette partie d'évaluation correspondent à la moyenne des résultats obtenus par simulation sur plusieurs topologies choisies de manière aléatoire. Chaque simulation réalisée dure 500 secondes.

3.1 Temps moyen de clusterisation

Les expérimentations conduites dans cette partie ont pour objectif de montrer le gain obtenu par notre protocole en terme de temps nécessaire pour finaliser le processus de clusterisation. Pour cet effet, pour vérifier les bornes établies, on a été amené à choisir plusieurs configurations de réseaux ayant des diamètres de 4, 6 et 8. On a fait varier les bornes D_{1max}, D_{1min} et D_2 utilisées lors du calcul du backoff durant les phases de déclaration des clusterheads et des gateways. L'objectif est d'évaluer l'impact du choix de ces bornes sur le temps nécessaire pour finaliser le processus de clusterisation et couvrir la totalité du réseau, étant données que ces bornes ont une influence sur le calcul des périodes T_i. Des topologies de 100, 200 et 300 noeuds ont été choisies pour conduire des simulations plusieurs fois. La périodicité pour relancer le processus de clusterisation a été fixée à 1 seconde. Le critère de la mobilité n'a pas été considéré durant ces expérimentations. Dans cette phase de simulation, nous avons mesuré :

- Le nombre moyen (Nbr) de CH et GW déclarés durant le processus de clusterisation, qui formeront l'infrastructure virtuelle ;
- Le temps moyen T_{clus} (Moy. Tclus) fait référence à la moyenne des temps, mesurés en secondes, requis pour la clusterisation du réseau durant toute cette phase de simulation ;
- Le temps maximal (Max. Tclus) correspond au temps maximal, requis pour la clusterisation du réseau, mesuré en secondes durant toute cette phase de simulation ;
- Le temps minimal (Min. Tclus), mesuré en secondes, représente le temps minimal mesuré en secondes durant toute cette phase de simulation pour finaliser la phase de clusterisation ;
- Le temps T_{clus} (Theo. Tclus) correspond au temps théorique, calculé conformément à la

Diamètre 4	(D_2, D_{1min})	Nbr	Moy. Tclus	Max. Tclus	Min. Tclus	Theo. Tclus
100 noeuds	(7, 7)	45	0.01456372	0.021673	0.010617	0.00908
	(15, 10)	12	0.00951413	0.011445	0.00307235	0.0115
	(31, 20)	10	0.01201683	0.014585914	0.003227	0.0152
200 noeuds	(7, 7)	79	0.02015157	0.02956906	0.01155264	0.00908
	(15, 10)	20	0.010632944	0.011884949	0.008779	0.0115
	(31, 20)	12	0.01338067	0.014578431	0.011565	0.0152
300 noeuds	(7, 7)	117	0.02678551	0.03071933	0.02096009	0.00908
	(15, 10)	28	0.011127514	0.012321526	0.008799	0.0115
	(31, 20)	16	0.013365299	0.01521203	0.011205	0.0152

Table 3.1 – Mesures réalisées de T_{clus} pour des topologies de taille variable ayant un diamètre de 4

Diamètre 6	(D_2, D_{1min})	Nbr	Moy. Tclus	Max. Tclus	Min. Tclus	Theo. Tclus
100 noeuds	(7, 7)	45	0.01217618	0.01557161	0.00876	0.009654
	(15, 10)	19	0.010543208	0.012179	0.008758	0.0122
	(31, 20)	17	0.01346586	0.016174	0.011205	0.0163
200 noeuds	(7, 7)	80	0.01630615	0.02132192	0.01234967	0.009654
	(15, 10)	23	0.010992441	0.011909	0.009627	0.0122
	(31, 20)	20	0.01319892	0.015895	0.010745	0.0163
300 noeuds	(7, 7)	124	0.023175604	0.026135305	0.018088973	0.009654
	(15, 10)	26	0.010818179	0.012108	0.009666	0.0122
	(31, 20)	22	0.013479852	0.015835	0.011577726	0.0163

Table 3.2 – Mesures réalisées de T_{clus} pour des topologies de taille variable ayant un diamètre de 6

formule (2.21), pour finaliser la phase de clusterisation du la totalité du réseau.

Le noeud DN a été choisi comme un noeud frontière durant toutes les simulations. Ainsi, on expérimente le pire des cas où la distance entre le noeud DN et le noeud le plus éloigné sera assimilée au diamètre. Les bornes D_{1max} et D_2 sont égales.

Les tables suivantes 3.1, 3.2 et 3.3 montrent les résultats mesurés pour des topologies de 100, 200 et 300 noeuds. Pour, le cas où les bornes maximales (D_{1max} et D_2) sont égales à 7, on remarque un dépassement de la borne théorique T_{clus} estimée pour la finalisation du processus de clusterisation. En effet, dans une telle situation, le risque d'occurrence de collision est élevé lors de la phase d'élection des clusterheads et celle de déclaration des gateways, en particulier pour les réseaux denses. Ainsi, plusieurs clusterheads vont participer à la phase de vérification. Le nombre moyen de clusterheads et gateways appuie ces résultats. On rappelle que la borne théorique T_{clus} ne prend en considération qu'une seule vague de vérification de la connexité par rapport au niveau précédent. Dans ce cas, une seule phase de vérification s'avère insuffisante pour rétablir la connexité entre les clusterheads des différents niveaux.

Ainsi, le choix des bornes utilisées est important afin de permettre aux candCHs et candGWs de se déclarer correctement tout en minimisant le risque d'occurrence de collision et le nombre

Diamètre 8	(D_2, D_{1min})	Nbr	Moy. Tclus	Max. Tclus	Min. Tclus	Theo. Tclus
100 noeuds	(7, 7)	41	0.010757799	0.012923664	0.008681	0.010228
	(15, 10)	29	0.01194258	0.01335155	0.010981	0.0129
	(31, 20)	26	0.0153158	0.01712556	0.01036119	0.0173
200 noeuds	(7, 7)	68	0.01248255	0.01637586	0.01005941	0.010228
	(15, 10)	32	0.01192476	0.01307195	0.010346	0.0129
	(31, 20)	30	0.01532508	0.01607781	0.012553	0.0173
300 noeuds	(7, 7)	91	0.01295194	0.0192561	0.0106346	0.010228
	(15, 10)	36	0.01211926	0.01306355	0.010981	0.0129
	(31, 20)	30	0.01503041	0.01667961	0.01144004	0.0173

Table 3.3 – Mesures réalisées de T_{clus} pour des topologies de taille variable ayant un diamètre de 8

de CH participant à la phase de vérification. Pour les valeurs 15 et 31 prises par les bornes maximales, on n'assiste pas à un dépassement de la borne T_{clus}. On fait remarquer aussi que cette borne T_{clus} reste la même pour des topologies ayant le même diamètre, indépendamment du nombre de noeuds.

3.2 Taille de l'ensemble dominant connexe

Le critère le plus répandu pour évaluer l'efficacité d'un algorithme de clusterisation est incontestablement la taille de l'ensemble dominant qui servira comme infrastructure virtuelle. Ce critère reflète le nombre de noeuds qui vont assurer des tâches additionnelles au profit des autres noeuds du réseau. Dans cette suite d'expérimentations, l'ensemble dominant connexe correspond aux coordinateurs élus pour le mécanisme `SPAN` et à l'ensemble formé par les clusterheads et les gateways pour notre mécanisme. Dans la suite, on a mesuré la taille moyenne des ensembles dominants connexes générés par les deux mécanismes, tout en variant la densité des noeuds et la taille de l'aire de simulations. La périodicité d'échange de messages hello est fixée à 10 secondes pour le mécanisme `SPAN`. La même périodicité est adoptée pour relancer notre algorithme de clusterisation.

Dans une première étape, on choisit des petites topologies de manière aléatoire, confinées dans une aire de simulation $500 \times 500m$. Les résultats exposés dans la figure suivante (3.1.a) montrent que le nombre de clusterheads et gateways comptabilisé pour notre algorithme est plus réduit lorsque la valeur des bornes maximales D_{1max} et D_2 est fixée à 31. Adopter une taille large pour ces bornes permet de réduire le risque de collisions pouvant survenir entre *cbeacons* et déclarations de gateways. En effet, opter pour une borne large réduit la probabilité de choisir le même délai aléatoire par les noeuds candidats durant les phases d'élection des clusterheads et des gateways.

On remarque que notre algorithme nous permet d'avoir un nombre moyen de coordinateurs

proche de celui de SPAN malgré qu'on opère sans une connaissance préalable du voisinage. En effet, l'organisation du processus de clusterisation en couches permet de limiter le nombre de candidats potentiels au rôle de clusterhead à un instant donné et d'avoir un certain ordonnancement géographique des clusterheads qui seront séparés de deux sauts dans la majorité des cas. Avoir une certaine redondance peut être parfois bénéfique pour l'acheminement des paquets de données. D'un autre côté, le recours à un calcul personnalisé des bornes D_1 et D_2 prenant en considération les critères d'énergie et le nombre de clusterheads voisins respectivement permet de favoriser certains noeuds plus aptes à assurer les rôles de clusterhead et gateway.

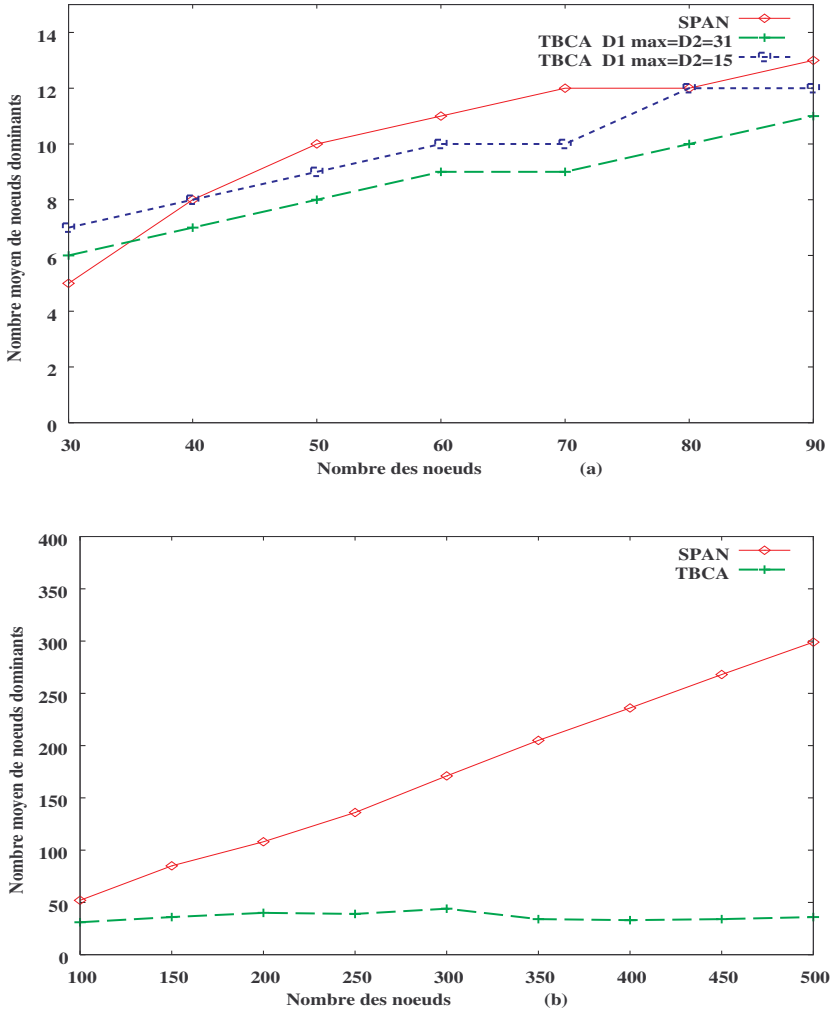


Figure 3.1 – Nombre moyen de noeuds dominants

Dans la deuxième étape, on considère de grandes topologies, de taille variable. Ces différentes topologies sont réparties dans une aire de simulation de $1000 \times 1000m$. Durant ces expérimentations, la taille des bornes maximales est fixée à 31, étant donné que le risque d'oc-

currence de collisions accroît avec la densité, pour les réseaux de grande envergure. Les résultats de la figure (3.1.b) consolident l'efficacité de notre mécanisme. Ce dernier maintient la taille de l'ensemble dominant connexe aussi réduite que possible sans une collecte préalable d'informations sur le voisinage. Cette réduction est importante, étant donné que l'ensemble dominant généré peut être en moyenne 7 fois plus petit que celui de `SPAN`, pour une topologie de 500 noeuds. Ainsi, on montre que la densité n'a aucun impact sur les performances de notre mécanisme. Alors que le nombre moyen de coordinateurs pour `SPAN` augmente considérablement avec le nombre de noeuds. En effet, pour le mécanisme `SPAN`, on assiste à la déclaration de plusieurs coordinateurs lors des premiers échanges de messages hello. Le risque de collisions entre ces messages Hello peut nuire au processus de réduction du nombre de coordinateurs. La probabilité croissante d'occurrence de collisions avec la densité des noeuds offre une vue incomplète sur le voisinage contribuant ainsi à fausser le jugement des noeuds sur le rôle à jouer.

Elire un nombre important de noeuds coordinateurs induit l'implication de plus de noeuds dans le processus routage, contribuant ainsi à épuiser plus rapidement l'énergie résiduelle des noeuds. Outre, l'envoi et la réception périodique des messages Hello, ces derniers doivent participer à l'acheminement des paquets de données.

3.3 Impact du trafic de contrôle sur le trafic de données

Dans cette section, on va s'intéresser à évaluer le coût de la clusterisation sur les performances du réseau. En effet, la majorité des approches existantes requiert un échange des messages de contrôle additionnels pour pouvoir réaliser la clusterisation et maintenir la structure générée actualisée suite aux changements topologiques pouvant survenir. Ce trafic additionnel peut pénaliser le trafic de données en particulier dans les réseaux de grande envergure. Pour pouvoir évaluer l'impact de ce trafic de contrôle additionnel introduit pour établir et maintenir la structure de clusterisation sur le trafic de données, le choix a été porté sur l'acheminement géographique pour le transfert des paquets de données. Afin d'anéantir le trafic induit par la découverte de la route, on assume la connaissance de l'emplacement géographique de tout noeud dans le réseau, cette information est soit fournie par un agent externe (telqu'un GPS), soit incluse dans les messages échangés entre voisins. Chaque noeud aura à connaître l'emplacement de la destination finale et de ses voisins dominants. Dans notre implémentation, ces informations ont été extraites du modèle de mobilité du simulateur JSIM afin d'éviter l'échange de messages complémentaires. On a adopté une approche *cross layer* où la couche MAC fait communiquer la liste des voisins dominants au processus de routage. Ainsi, la source envoie directement les paquets vers la destination, si cette dernière se trouve dans son voisinage immédiat. Dans le cas contraire, elle envoie le paquet vers le noeud dominant le plus proche de la destination. Les clusterheads et gateways élus par notre mécanisme forment l'ensemble dominant connexe, tandis

que les coordinateurs représentent les noeuds dominants pour le mécanisme SPAN. Chaque noeud dominant effectue un raisonnement similaire à celui réalisé par la source. Il achemine le paquet vers le noeud dominant voisin le plus proche de la destination, si cette dernière ne se trouve pas à proximité du noeud dominant en question. Ainsi, l'acheminement des paquets de données sera assuré uniquement par les noeuds dominants jusqu'à atteindre la destination.

Dans cette phase d'expérimentations, on va considérer des topologies fixes de 200 noeuds répartis dans une surface de $500 \times 500m$ offrant une bande passante de 2 Mbps. Lors des simulations, les noeuds formant le réseau sont choisis de manière aléatoire, exception faite pour les noeuds source et destination du trafic qui ont été placés aux frontières du réseau. L'intérêt de cette disposition est de garantir un routage multi sauts entre ces derniers. Le trafic injecté sur le réseau correspond à des flux CBR[16] provenant de 10 sources vers 10 destinations, avec une taille de paquet égale à 512 octets. Pour notre algorithme, les bornes maximales (D_{1max} et D_2) sont égales. La table suivante résume les paramètres adoptés durant cette phase de simulation (3.4) :

Paramètres de TBCA	
Périodicité de MaJ de la clusterisation	10 intervalles beacon
(D_{1max}, D_{1min})	(31, 25)
(D_{1pmin}, D_{1npmin})	(15, 31)
$NbrMax\_gw\_CH$	6
$NombreMoy\_gw/CH$	5
Nombre de périodes T_i successives en tant que dominant	5

Table 3.4 – Paramètres utilisés dans la simulation

On note que l'énergie initiale disponible au niveau de chaque noeud est de 1000 watts. Les taux de consommation d'énergie adoptés sont conformes aux valeurs spécifiées dans la table suivante (3.5) :

Transmission	Réception	Ecoute passive	Inactif
0.660	0.395	0.296	0.130

Table 3.5 – Taux de consommation de l'énergie (watts)

Lors de cette évaluation, on a focalisé sur la quantification du nombre de paquets délivrés avec succès pour toutes les destinations durant toute la simulation, reflétant l'aptitude du réseau à relayer le trafic de données en présence d'un trafic de contrôle additionnel. On a aussi mesuré le délai moyen de séjour, calculé selon la formule suivante :

$$D = \frac{\Sigma(r_i - g_i)}{Nombre\_paquets\_delivres\_avec\_succes} \tag{3.1}$$

Avec :

[16] CBR : Constant Bit Rate

r_i : Instant de réception du paquet i

g_i : Instant de génération du paquet i

On s'est intéressé aussi à la mesure de l'énergie résiduelle moyenne et l'énergie moyenne consommée par paquet, étant donnée que l'énergie est une ressource critique pour un réseau ad hoc. L'énergie résiduelle correspond à la moyenne de toutes les énergies résiduelles au niveau de tous les noeuds à la fin de chaque simulation. Le calcul de l'énergie moyenne consommée par paquet (EconsPkt) est défini comme suit :

$$EConsPkt = \frac{EConsG}{Nombre\_paquets\_delivres\_avec\_succes} \tag{3.2}$$

$$EConsG = Emax - Erest \tag{3.3}$$

Avec :

EConsG : Energie moyenne consommée pour tous les paquets

Erest : Energie moyenne résiduelle

Emax : Energie initiale maximale

3.3.1 Etape 1 : variation de la charge

La première étape consistait à faire varier la charge du réseau. Pour ces simulations, la périodicité de mise à jour pour les mécanismes de clusterisation a été fixée à 10 intervalles Beacon, la taille de l'intervalle beacon étant conforme au standard 802.11.

La figure suivante (3.2) illustre le nombre moyen de paquets délivrés avec succès. On remarque que les deux mécanismes offrent des résultats similaires à faible charge. Alors que, notre mécanisme `TBCA` permet de délivrer plus de paquets que le mécanisme `SPAN`, lorsque la charge du trafic augmente. En effet, notre algorithme requiert une période réduite pour finaliser la clusterisation de tout le réseau. Durant cette phase de clustering, le trafic de données est suspendu. Ainsi, les paquets de contrôle dédiés à la clusterisation ne vont pas entrer en concurrence avec le trafic de données laissant le canal libre le reste du temps aux paquets de données. De cette manière, on arrive à minimiser le risque de collisions et le canal va être mieux exploité pour l'acheminement des paquets de données. Alors que pour le mécanisme `SPAN`, les messages Hello vont entrer en compétition avec les paquets de données augmentant le risque d'occurrence de collisions. Ainsi, un paquet de contrôle Hello peut retarder l'envoi de paquets de données, si ce dernier les précède au niveau de la file d'envoi.

Ces résultats sont consolidés par les résultats correspondants au calcul du délai moyen de séjour, exposés par la figure (3.3). Comme le montre la figure, le délai moyen de séjour pour les deux mécanismes croît considérablement lorsque la charge du réseau augmente. Toutefois, les

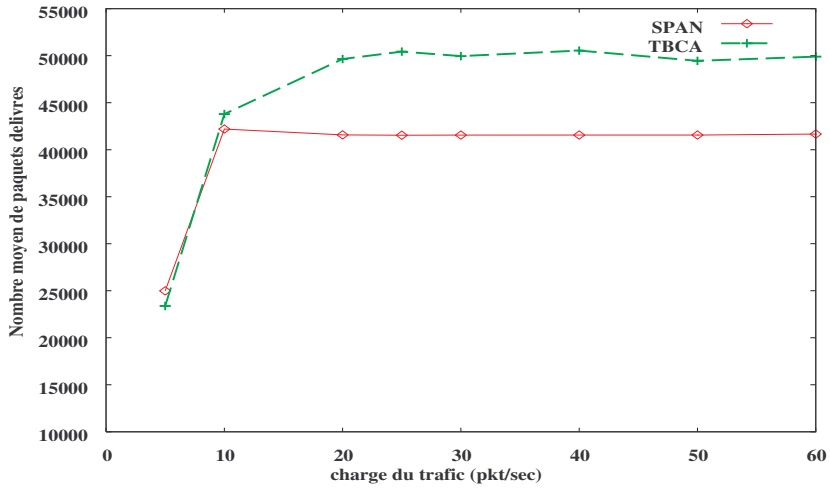


Figure 3.2 – Calcul du nombre moyen de paquets délivrés avec succès en fonction de la charge du trafic par flux

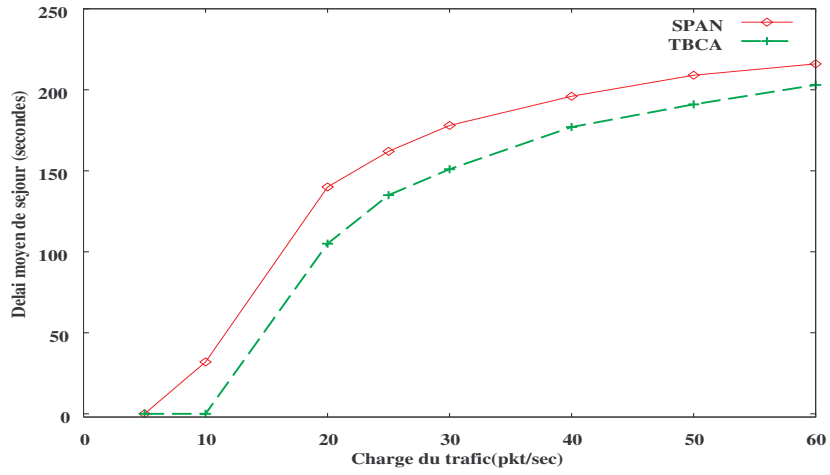


Figure 3.3 – Délai moyen de séjour en fonction de la charge du trafic par flux

délais moyens mesurés avec notre mécanisme sont toujours meilleurs que ceux obtenus par le mécanisme `SPAN`. En effet, les messages Hello périodiques, nécessaires au maintien de l'ensemble dominant généré par le mécanisme `SPAN`, contribuent à l'augmentation du délai moyen de séjour. Premièrement, ces messages vont concurrencer avec le trafic utile pour accéder au canal. Pour un environnement à charge élevée, un tel trafic encombre le canal et contribue à bloquer l'envoi des paquets de données. Comme on l'a déjà expliqué, les paquets de données au niveau d'un noeud donné vont rester en attente, si ce dernier a un paquet Hello à envoyer pour informer son voisinage de son statut, la liste de ses voisins et la liste de ses coordinateurs. Alors que pour notre mécanisme, une fois l'échange des paquets de données repris, aucun trafic de contrôle additionnel, dédié à la clusterisation, ne le perturbe. On rappelle que l'envoi des paquets de données est suspendu momentanément pour une période de temps négligeable. Tout noeud ayant participé à la phase de clusterisation ne va pas attendre la finalisation de ce processus pour pouvoir reprendre le traitement normal des paquets de données. Grâce à la formule (2.28 ou 2.29), il calcule la période d'attente au bout de laquelle il est autorisé à reprendre le traitement normal des paquets en attente, conformément au standard 802.11. Cette période d'attente est nécessaire afin de ne pas perturber le déroulement normal du processus de clusterisation au niveau des couches suivantes et éviter les éventuelles collisions entre les paquets de données et les paquets de contrôle (*cbeacon*, ACK_BT et les déclarations des gateways). Ces collisions peuvent fausser les décisions prises par les noeuds, concernés encore par le processus de clusterisation, et engendrer aussi des retransmissions inutiles pour les paquets de données.

Les résultats exposés dans les figures (3.4) et (3.5) rapportent les mesures faites pour calculer l'énergie résiduelle moyenne et l'énergie moyenne consommée par paquet. On remarque que malgré la différence remarquable au niveau du nombre moyen de paquets délivrés avec succès, l'énergie résiduelle moyenne mesurée pour notre mécanisme se rapproche étroitement de celle du mécanisme `SPAN`. De ce fait, le mécanisme `SPAN` est plus gourmand en terme d'énergie. Cette consommation est due à l'envoi des messages additionnels Hello, outre la transmission des paquets de données. Le nombre de paquets Hello périodique est proportionnel au nombre de noeuds, ainsi, pour un réseau de 200 noeuds, 200 messages Hello additionnels vont être injectés sur le réseau à chaque période de mise à jour. Par ailleurs, la probabilité d'occurrence de collision étant plus élevée avec la charge croissante, les retransmissions des paquets de données contribuent aussi à gaspiller l'énergie.

La figure (3.5) consolide l'efficacité de notre mécanisme à conserver l'énergie, en éliminant le trafic de contrôle nécessaire à la formation de l'ensemble dominant lors de l'acheminement des paquets de données. Ainsi, plus de ressources sont disponibles pour cette charge utile, permettant de faire écouler plus de paquets de données. Pour cette raison, l'énergie moyenne consommée par paquet est plus réduite pour notre mécanisme.

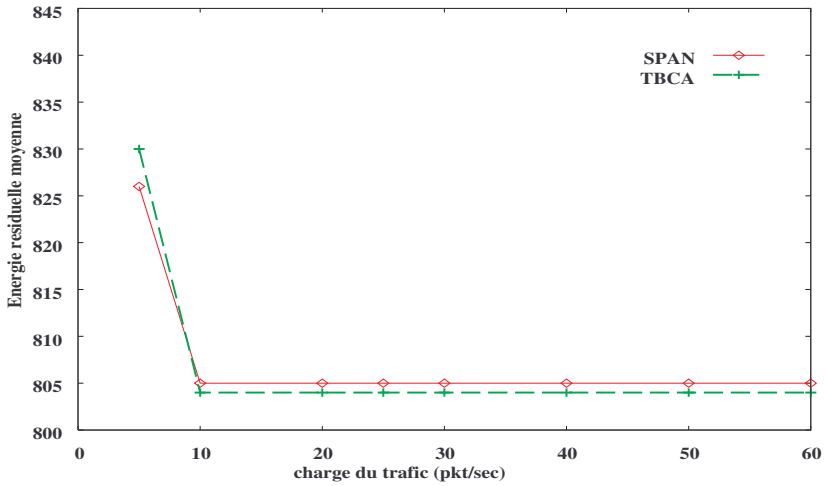


Figure 3.4 – Energie résiduelle moyenne en fonction de la charge du trafic par flux

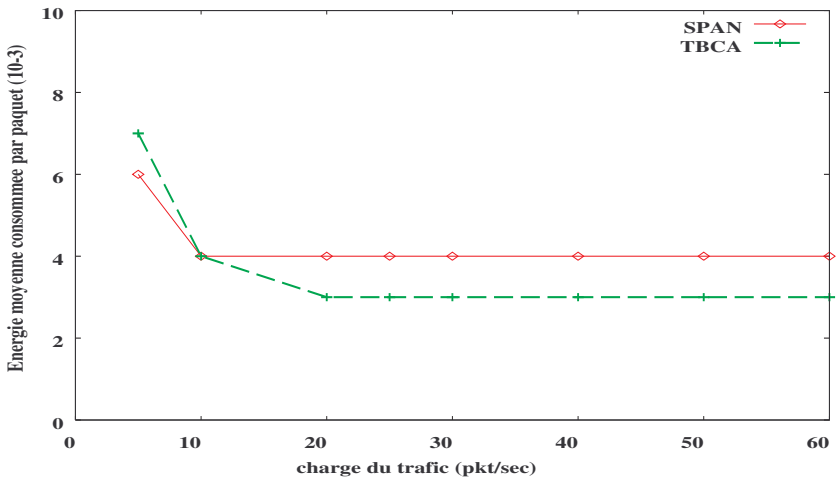


Figure 3.5 – Energie moyenne consommée par paquet en fonction de la charge du trafic par flux

3.3.2 Etape 2 : variation de la périodicité de mise à jour

La deuxième étape consistait à faire varier la périodicité de mise à jour et de reclusterisation respectivement pour `SPAN` et notre algorithme. L'objectif est de montrer et de mieux évaluer l'impact du trafic périodique introduit pour maintenir la structure de clusterisation actualisée. En effet, le mécanisme `SPAN` se base sur l'échange périodique de messages Hello afin de permettre à chaque noeud de collecter des informations sur son voisinage à deux sauts et évaluer son éligibilité au rôle de coordinateur. Alors que notre mécanisme déclenche périodiquement le processus de clusterisation pour déterminer un nouvel ensemble dominant connexe afin de prendre en considération les éventuels changements topologiques et permettre à tous les noeuds d'avoir une chance égalitaire pour assurer le rôle de CH ou GW.

Les mesures réalisées ont été faites dans un environnement à charge relativement élevée, où chaque source injecte 20 paquets/sec. Notre objectif est de simuler la charge pouvant transiter dans un réseau ad hoc de grande envergure.

Les mesures du nombre moyen de paquets correctement délivrés, exposées dans la figure (3.6), montrent que les résultats obtenus avec notre mécanisme sont nettement meilleurs. On note que le nombre moyen de paquets délivrés avec succès pour `TBCA` est relativement le même quelque soit la périodicité de mise à jour de la structure de clusterisation. La reclusterisation de la totalité du réseau, en ayant recours au mécanisme de stabilité défini, n'a pas d'impact négatif sur les autres algorithmes reposant sur l'ensemble dominant connexe et les performances du réseau. L'aptitude de `TBCA` à finaliser la phase de construction de l'ensemble dominant connexe en un temps très réduit et l'allègement du réseau des paquets de contrôle dédiés à cette étape constituent les apports majeurs ayant permis de préserver la capacité du réseau. Alors que le nombre moyen de paquets, pour le mécanisme `SPAN`, augmente petit à petit lorsque la périodicité d'envoi des messages Hello augmente. En espaçant les envois périodiques de ces messages de contrôle, le mécanisme libère plus de ressources au profit du trafic de données. Comme l'expose la figure (3.6), plus la périodicité de mise à jour du statut pour `SPAN` est petite, plus le nombre de paquets de données diminue vue que le trafic de contrôle dédié à la clusterisation est important.

Ce trafic de contrôle additionnel surcharge le réseau lorsque périodicité d'envoi des messages Hello est réduite. L'occupation du canal par ces messages Hello contribue à bloquer le trafic utile et à retarder la transmission des paquets de données. L'attente induite au niveau des files d'attente relatives aux noeuds impliqués dans l'acheminement contribue à faire croître le délai de séjour moyen pour le mécanisme `SPAN`, comme illustré dans la figure (3.7). En augmentant la périodicité de mise à jour, on remarque une diminution du délai de séjour moyen, étant donné que le trafic de contrôle dédié à la maintenance de l'ensemble dominant est plus espacé. Cette réduction au niveau du délai de séjour moyen est aussi sentie pour notre mécanisme lorsque la périodicité du lancement du processus de clusterisation croît.

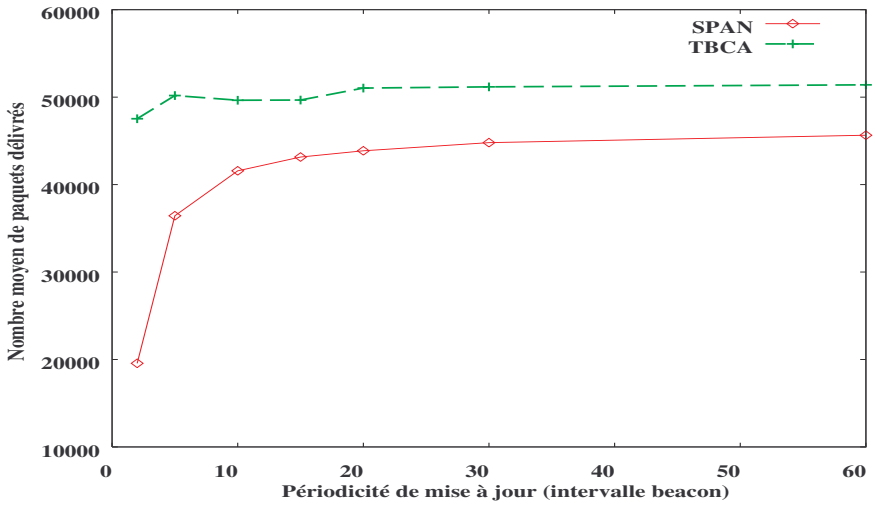


Figure 3.6 – Nombre moyen de paquets délivrés en fonction de la périodicité de mise à jour du mécanisme de clusterisation

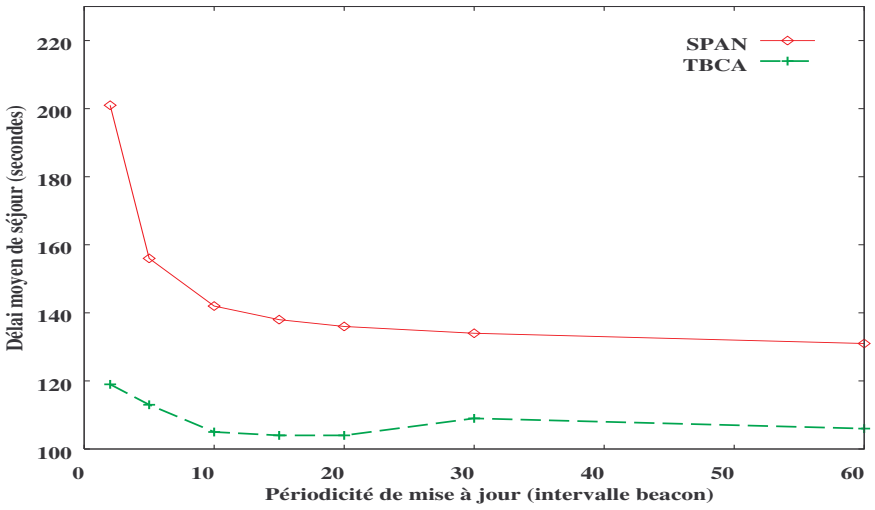


Figure 3.7 – Délai de séjour moyen en fonction de la périodicité de mise à jour du mécanisme de clusterisation

L'allègement du réseau des messages de contrôle à travers l'espacement des mises à jour contribue à faire écouler plus de trafic de données utiles. Par conséquence, les deux mécanismes consomment plus de ressources, étant donnée que l'opération de transmission est la plus coûteuse en terme d'énergie. La figure (3.8), exposant les mesures de l'énergie résiduelle moyenne, montre que le mécanisme `SPAN` conserve mieux l'énergie lorsqu'on opte pour des mises à jour fréquentes à petit intervalle. La cause d'une telle différence est la capacité de notre mécanisme `TBCA` à relayer plus de paquets de données. La différence remarquable au niveau du nombre de paquets délivrés avec succès, pour une mise à jour variant entre 2 à 20 intervalles beacon, explique cette légère consommation d'énergie supplémentaire par rapport à `SPAN`.

Par contre, le niveau de l'énergie résiduelle décroît pour le mécanisme `SPAN` lorsqu'on opte pour des mises à jour espacées dans le temps. Ce choix permet de réduire l'impact du trafic de contrôle sur le trafic de données. L'augmentation du nombre de paquets délivrés en témoigne et justifie cette réduction au niveau de l'énergie résiduelle moyenne. Pour notre mécanisme, le niveau d'énergie résiduelle se stabilise au fur et à mesure qu'on espace l'appel au processus de clusterisation. Ce gain en énergie est aussi du au fait qu'on fait intervenir moins de noeuds dominants dans l'opération d'acheminement des données. Par conséquence, cet avantage se traduit aussi par un gain au niveau de l'énergie consommée par paquet. L'énergie consommée par paquet est toujours plus réduite pour notre mécanisme, comme l'illustre la figure (3.9).

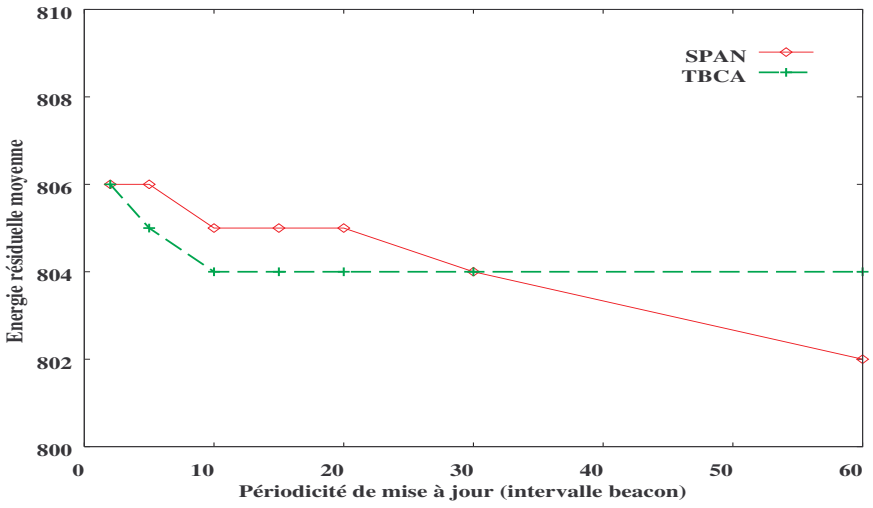


Figure 3.8 – Energie résiduelle moyenne en fonction de la périodicité de mise à jour du mécanisme de clusterisation

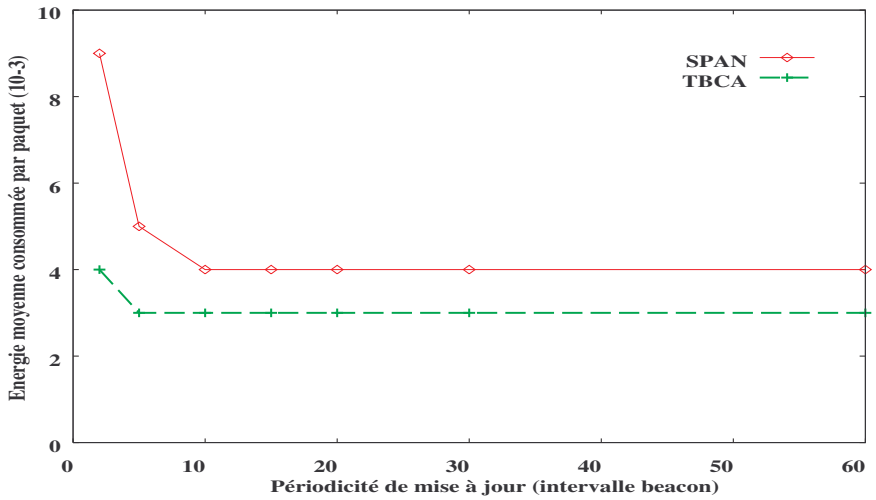


Figure 3.9 – Energie moyenne consommée par paquet en fonction de la périodicité de mise à jour du mécanisme de clusterisation

3.3.3 Etape 3 : variation de la densité

La dernière étape vise à évaluer l'impact de la densité sur les performances du réseau, en faisant varier la taille du réseau. Le choix a été porté sur différentes topologies de taille variable confinées dans une aire de simulation fixe à $500 \times 500m$. La périodicité de mise à jour est de 10 intervalles beacon. Chaque source transmet 20 paquets/sec afin de simuler le trafic pouvant transiter dans un réseau de grande envergure.

A travers les mesures relatives au nombre de paquets délivrés illustrées à travers la figure (3.10), on note une dégradation du débit pour le mécanisme `SPAN` lors de la croissance du nombre de noeuds. Cette diminution est le résultat de l'augmentation du trafic de contrôle proportionnel au nombre de noeuds. Etant donné qu'on a maintenu un flux constant de paquets, les messages Hello échangés de manière périodique constituent la charge additionnelle contribuant à surcharger le réseau et à affecter ses performances. Toutefois, le mécanisme `TBCA` semble insensible à la variation de la densité en préservant un débit constant indépendamment du nombre de noeuds. On rappelle que le temps de clusterisation dépend essentiellement du diamètre, par conséquence la variation du nombre de noeuds n'a pas un impact sur le temps requis pour achever cette phase de clusterisation.

L'influence de la densité sur les performances du mécanisme `SPAN` se confirme à travers les résultats exposés dans la figure (3.11) correspondant au délai de séjour moyen. Ce délai croît légèrement avec le nombre de noeuds, étant donné que le nombre croissant des messages additionnels Hello contribue à bloquer le trafic de données et à augmenter les délais d'attente au niveau des files d'attente sur le chemin emprunté. On remarque aussi que le délai mesuré pour le

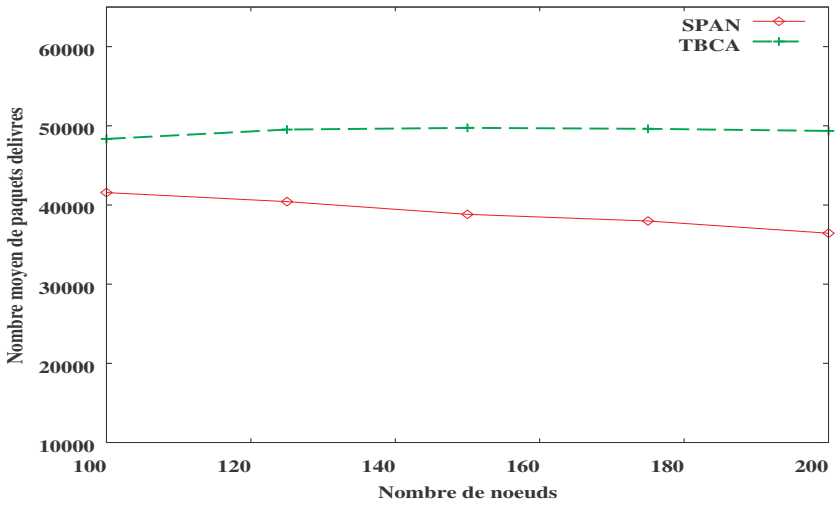


Figure 3.10 – Nombre moyen de paquets délivrés en fonction de la densité

mécanisme `TBCA` est inférieur au délai constaté pour le mécanisme `SPAN`, avec une légère élévation lorsque le réseau devient de plus en plus dense. Ce surcroît se justifie par la croissance de la taille de l'ensemble dominant avec la densité (se référer aux résultats exposés dans la section 3.2). Ainsi le nombre croissant de noeuds, impliqués dans les opérations d'acheminement et voulant accéder au canal, est un facteur principal dans l'augmentation du délai de séjour.

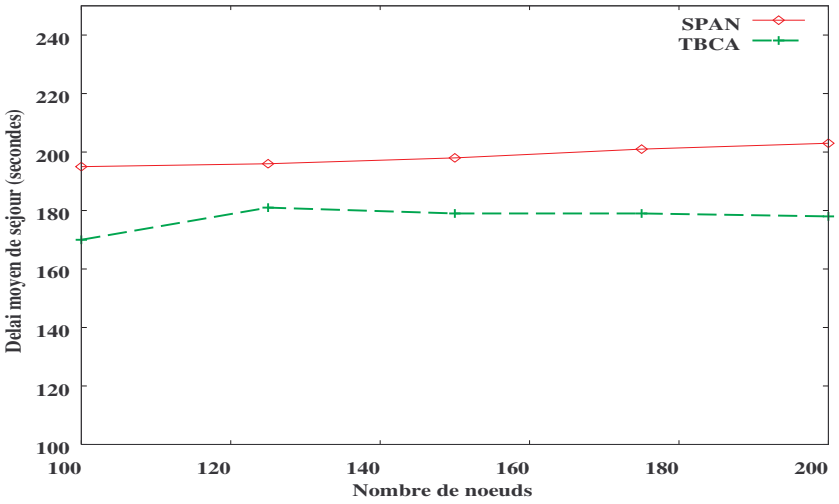


Figure 3.11 – Délai de séjour moyen en fonction de la densité

Le troisième critère adopté pour évaluer l'impact de la densité est l'énergie consommée par paquet, dont les mesures sont illustrées à travers la figure (3.12). Malgré la légère amélioration au niveau du débit pour le mécanisme `TBCA`, notée au niveau de la figure (3.10), on constate que

le coût en terme d'énergie décroît avec la densité pour notre mécanisme avant de se stabiliser. La taille de l'ensemble dominant connexe grandit avec le nombre de noeuds, ce qui permet d'offrir plus d'alternatives de routes et écouler plus de trafic. Cependant, cette taille grandissante du nombre de noeuds a un effet néfaste pour le mécanisme SPAN, étant donné qu'il gaspille son énergie pour les transmissions périodiques de paquets de contrôle. Ce trafic additionnel peut augmenter le risque d'occurrence de collision dans un environnement dense, produisant des retransmissions inutiles. Tous ces facteurs contribuent à augmenter le coût par paquet en terme d'énergie.

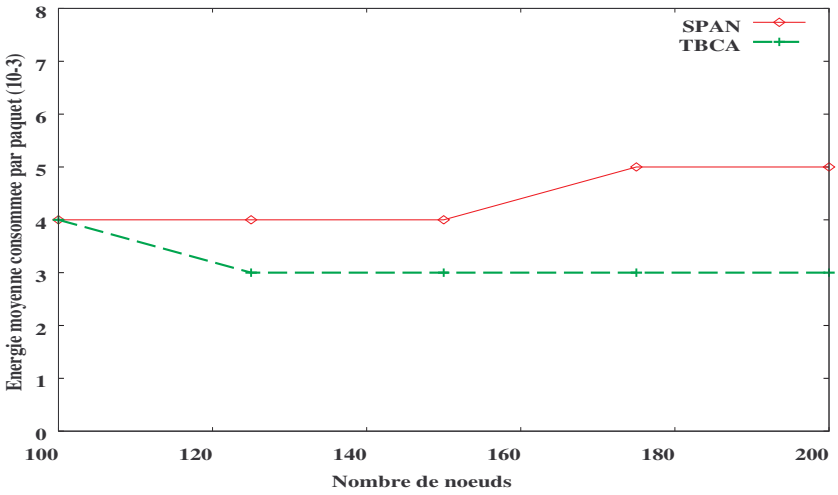


Figure 3.12 – Energie moyenne consommée par paquet en fonction de la densité

3.4 Impact de la mobilité

Un autre aspect auquel nous nous intéressons est la robustesse de la structure de clusterisation générée face aux changements topologiques dus à la libre mobilité accordée aux différents noeuds. Dans ce contexte, on s'est proposé d'évaluer l'impact de la vitesse et de la fréquence de mise à jour sur les performances du système.

Lors des simulations réalisées de 500 secondes, le choix a été porté sur différentes topologies de 100 noeuds en mouvement dans une aire de simulation de $500 \times 500m$, exception pour les noeuds sources et destinations qui conservent leur position initiale. Chaque source maintient un trafic CBR avec un taux d'envoi de 20 pkt/sec. Le modèle de mobilité adopté est le *random way point* [21]. Pour ce modèle, un noeud mobile commence par rester dans une position pendant une certaine période. Une fois cette période écoulée, le noeud mobile choisit une destination aléatoire dans l'aire de déplacement et une vitesse choisie uniformément entre [min vitesse, max vitesse]. Le noeud mobile se déplace vers cette nouvelle position avec la vitesse choisie. Une fois arrivé, il

effectuera une pause pendant une seconde avant de relancer le processus. Le critère d'évaluation adopté dans cette phase est le nombre de paquets délivrés avec succès vers toutes les destinations durant toute la simulation.

Lors de la première étape, on a fait varier la vitesse maximale des noeuds. La périodicité de mise à jour de la structure de clusterisation a été fixée à 10 intervalles beacon. D'après les résultats exposés dans la figure (3.13), on remarque la dégradation des performances des deux mécanismes avec l'accroissement de la vitesse. Toutefois, les impacts négatifs sont moins marquants pour le mécanisme TBCA. Les résultats de la figure (3.13) illustrent ces propos. Malgré la dégradation des performances de `TBCA`, ce dernier surpasse toujours le mécanisme `SPAN`. Le rafraichissement périodique nous permet de prendre en considération l'état actuel du réseau (connexité, énergie, ...) lors de la construction de l'ensemble dominant connexe. Ainsi, on conserve une dorsale actualisée apte à assurer les opérations d'acheminement.

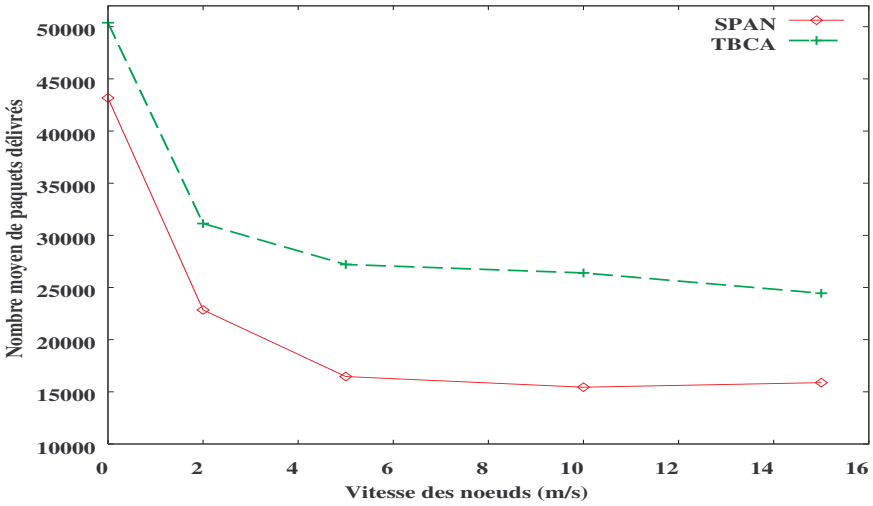


Figure 3.13 – Nombre moyen de paquets délivrés en fonction de la vitesse

Dans la deuxième étape, on a fixé la vitese maximale de tout noeud à 5 m/sec, et on a fait varier la périodicité de mise à jour adoptée par le mécanisme de clusterisation. La figure (3.14) montre la détérioration remarquable des performances du système pour le mécanisme `SPAN` lorsque la période de mise à jour est assez longue. Le recours à des informations collectées sur le voisinage pour la prise de décision sur le rôle à assurer par tout noeud justifie ce comportement. En effet, la collecte des informations à deux sauts requiert du temps, cette période dépend essentiellement de la périodicité de mise à jour. L'espacement des mises à jour ralentit la propagation des informations sur le voisinage à deux sauts. Dans un environnement ad hoc mobile caractérisé par l'imprévisibilité des changements affectant la topologie, ces informations peuvent devenir périmées lors de leur utilisation par le noeud. Pour cette raison, les performances

de SPAN se dégradent considérablement dans un environnement à mobilité élevée, en particulier lorsque la périodicité de mise à jour est peu fréquente. Dans une telle situation, les informations de routage peuvent être obsolètes lors de leur utilisation et les décisions de routage prises par les noeuds seront incorrectes.

Pour TBCA, le déclenchement périodique du processus de clusterisation nous permet d'avoir une structure plus robustesse face aux changements topologiques. Par ailleurs, le mécanisme de stabilité nous permet de garantir une certaine stabilité de l'infrastructure virtuelle. Ainsi, on essaie de limiter les impacts négatifs de la mobilité afin d'offrir une dorsale apte à supporter efficacement les protocoles de haut niveau. Le choix de la période de mise à jour est important du moment qu'elle peut affecter la stabilité de la structure de clusterisation.

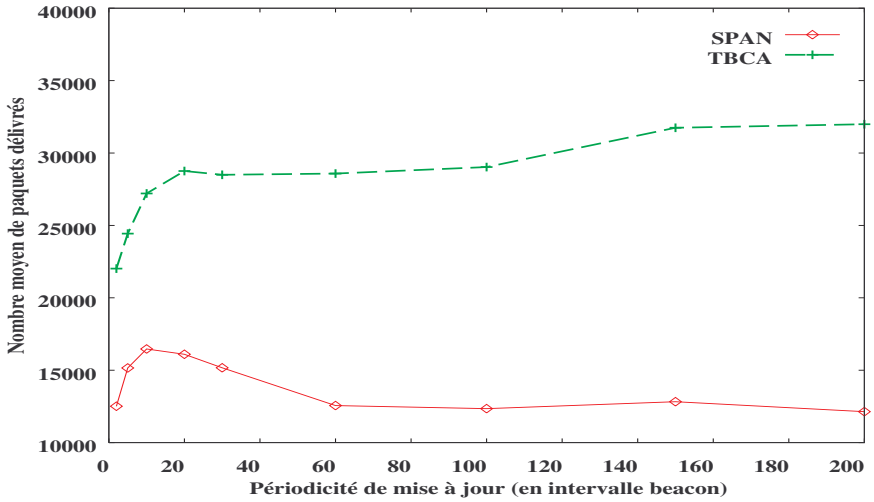


Figure 3.14 – Nombre moyen de paquets délivrés en fonction de la périodicité de mise à jour

3.5 Conclusion

Les performances réalisées par notre mécanisme ont été systématiquement évaluées et comparées à celles du mécanisme de référence choisi SPAN. SPAN a été choisi comme protocole de référence car il opère dans un environnement synchrone, à l'instar de notre mécanisme. Il est aussi mis en oeuvre au niveau de la couche MAC afin d'offrir une approche de conservation d'énergie. D'après les résultats des simulations, il est clair que TBCA surclasse SPAN.

Chapitre 4

Exploitation de la clusterisation pour la conservation d'énergie dans les réseaux Ad hoc

Malgré les progrès technologiques perçus, l'énergie demeure encore une ressource critique dans un environnement ad hoc, étant donnés que les hôtes mobiles sont munis de batteries autonomes. La consommation d'énergie dans les hôtes mobiles relève un défi pour la durée de vie du réseau, du moment que l'épuisement de l'énergie au niveau de certains noeuds peut entraîner le partitionnement du réseau et la perte de connexité par conséquence. Ainsi, pour assurer un déploiement efficace des réseaux ad hoc à grande échelle, il s'avère nécessaire de concevoir des mécanismes qui tiennent en compte la durée limitée des batteries des équipements mobiles. L'objectif principal est de minimiser le gaspillage de cette ressource afin de maintenir le réseau fonctionnel aussi longtemps que possible.

Par ailleurs, l'absence d'une administration centralisée et d'une infrastructure fixe rend obligatoire la collaboration de tous les noeuds du réseau. Pour assurer la connectivité dans un environnement ad hoc, chaque noeud est tenu alors à jouer le rôle d'hôte et de routeur afin d'assurer l'acheminement de trafic transitant à travers le réseau. Ainsi, tout noeud peut être appelé à participer dans les opérations de découverte de chemins, d'acheminement des paquets de données et aussi durant la phase de maintenance des routes établies suite à un changement de la topologie. La participation des différents noeuds à ces phases puise de leurs ressources limitées, étant donné que chaque noeud est appelé à rester à l'écoute du canal afin d'intercepter et de rediffuser les paquets de contrôle. D'un autre côté, la libre mobilité accordée aux noeuds impose des défis pour le routage dans ce type de réseaux. Les défaillances fréquentes des routes établies induisent un re-déclenchement de la phase de découverte des routes ou de la maintenance des routes.

Plusieurs travaux ont démontré que l'activité réseau est très coûteuse en énergie. Les opé-

rations d'émission et de réception des paquets de contrôle et de données, au profit des autres noeuds, consomment de l'énergie. Cependant, un noeud gaspille de l'énergie même étant inactif au niveau applicatif en restant à l'écoute du canal pour l'interception des paquets. En effet, le médium radio est très particulier. Une de ses principales propriétés est que le flux entre deux noeuds ne peut pas être isolé. Le médium est pervasif et lorsqu'un noeud émet, tous les voisins reçoivent le paquet. Cette énergie consommée pour la transmission ou réception de données, le traitement des requêtes ou l'acheminement des paquets au profit d'autres noeuds peut être considéré comme une consommation d'énergie utile. Alors que l'écoute du canal, l'interception de paquets destinés à un noeud voisin et la retransmission de paquets en cas de collision sont perçues comme un gaspillage de l'énergie, pouvant être évité.

Contrôler la consommation d'énergie individuelle ou totale, peut se faire de plusieurs manières et à plusieurs niveaux de la couche protocolaire [60]. Toutefois, nos travaux se focalisent sur les protocoles opérant au niveau MAC, ayant essayé de réduire la consommation d'énergie. Dans la littérature, les solutions proposées pour le problème de conservation d'énergie dans un environnement ad hoc peuvent être classées en trois catégories :

- Le contrôle de puissance de transmission : Il consiste à choisir le niveau de puissance de transmission approprié au niveau de chaque noeud dans un environnement ad hoc afin d'accroître la capacité du réseau tout en maintenant la connectivité de ce dernier [76, 90]. La sélection de la puissance de transmission optimale a été traitée dans [48, 86], sans toutefois s'intéresser à la connexité du réseau. Dans [76], on contrôle la topologie du réseau en ajustant la puissance de transmission des noeuds en fonction des changements topologiques pour maintenir la topologie connexe. Dans [91], on présente un algorithme distribué où chaque noeud échange des messages avec son voisinage à deux reprises afin de décider des liens à maintenir. L'approche présentée dans [18] coopère avec la couche routage. Un mécanisme estime la quantité d'énergie requise pour une communication fiable à travers n'importe quel lien. Cette estimation d'énergie permet de régler la puissance de transmission et le coût des liens. Certaines approches se sont basées sur la notion d'ensemble dominant connexe, à l'instar des travaux de [6, 57, 65]. Toutefois, l'utilité du contrôle de la puissance de transmission à réduire la consommation d'énergie dépend étroitement du modèle radio adopté et des spécifications du matériel.
- Le routage orienté conservation d'énergie pour lequel les décisions de routage prennent en compte l'optimisation de l'énergie. Dans [41], tout noeud, disposant d'énergie au dessous d'un certain seuil, n'est pas autorisé à participer à l'acheminement des paquets de données au profit des autres noeuds. Les auteurs de [79] proposent différentes métriques en se basant sur la consommation de l'énergie afin d'étendre la durée du vie du noeud et du réseau. La durée de vie d'un noeud a été aussi considérée dans [81]. Les auteurs de [9] optent pour la prise en compte du taux d'erreur du lien conjointement à l'énergie requise

pour la transmission d'un paquet lors du choix d'une route. Emprunter des liens ayant un taux d'erreur élevé peut augmenter considérablement l'énergie effective consommée pour l'envoi fiable d'un paquet, le gaspillage de l'énergie du aux éventuelles retransmissions de paquets justifie une telle augmentation.

- Le mode conservation d'énergie : Etant donnée que l'interface réseau constitue le majeur consommateur d'énergie, on doit identifier les principales sources de gaspillage d'énergie afin de mieux définir les compromis possibles une fois les origines de ces pertes cernées. L'une des principales causes contribuant à dissiper l'énergie est l'écoute passive du canal. Tout noeud doit scruter le canal de manière continue afin d'intercepter les paquets lui étant destinés. Toutefois, l'écoute de paquets dressés à des noeuds avoisinants adhère à réduire inutilement l'énergie résiduelle au niveau de chaque noeud. L'énergie consommée lors d'une opération de réception est légèrement inférieure à celle nécessaire pour une transmission de paquets, cependant elle dépasse légèrement l'énergie dissipée lors de l'écoute passive. L'occurrence de collisions et le risque d'interférence sont des caractéristiques inhérentes au médium partagé sans fil, les retransmissions de paquets engendrées consomment de l'énergie vainement. Dans ce contexte, le mode conservation d'énergie vise à contrôler la topologie du réseau en déterminant les noeuds aptes à rester actifs et à participer aux opérations additionnelles du réseau et ceux pouvant entrer en mode de veille.

Les deux premières catégories essaient de réduire la consommation d'énergie en choisissant les noeuds les plus appropriés pour les opérations de communication ou la puissance adéquate à utiliser. Toutefois, les sources de gaspillage d'énergie persistent. En effet, les noeuds doivent être toujours actifs même s'ils ne sont pas impliqués dans des opérations d'échange de données. Or, la consommation d'énergie, à cet état d'écoute, est légèrement inférieure à celle consommée lors d'une opération de réception ou de transmission. Ainsi, de tels noeuds vont perdre de leur énergie à leur insu, à travers l'écoute passive du canal et l'interception de paquets destinés à des voisins. La troisième approche permet aux noeuds mobiles d'entrer en mode veille pendant certaines périodes afin de réduire leur consommation d'énergie et de prolonger la durée de vie du noeud et du réseau par conséquence.

Parmi ces alternatives présentées, nous nous intéressons à la troisième approche adoptée pour la conservation d'énergie, du moment qu'elle peut être aisément utilisée conjointement avec une technique de clusterisation. Les approches se basant sur le principe du mode veille, proposées dans la littérature, peuvent être classées en fonction du mécanisme adopté pour assurer le réveil des différents noeuds [102]. Dans ce contexte, on distingue trois catégories : un réveil à la demande, un réveil ordonnancé et un réveil asynchrone pour les différents noeuds. Les mécanismes de réveil à la demande requièrent une signalisation hors bande pour activer les noeuds en mode veille.

Pour les mécanismes de réveil ordonnancé, les noeuds en mode conservation d'énergie se réveillent simultanément de manière périodique afin de pouvoir communiquer ensemble, à l'instar du mécanisme `PSM` [67] (Power Saving Mechanism) défini par le standard 802.11. Ce mode nécessite une certaine synchronisation horloge entre les noeuds afin de pouvoir planifier un réveil collectif de tous les noeuds au même instant. Le mécanisme `PSM` permet aux noeuds de rester en mode veille afin de réduire leur consommation en énergie. Ainsi, une station peut être soit active, fonctionnant de manière normale, soit en mode veille où elle est incapable de transmettre et de recevoir des paquets. Pour le mécanisme 802.11 `PSM`, le temps est divisé en intervalles beacon.

Figure 4.1 – Structure d'un intervalle beacon pour le mécanisme PSM

Chaque intervalle beacon est constitué de deux périodes distinctes, comme l'illustre la figure (4.1). La première période ATIM [17] permet aux noeuds d'annoncer le trafic en attente à travers l'échange de paquets ATIM et ACK entre source et destination. Chaque station devra émettre une trame ATIM vers toute station ayant un ou plusieurs paquets tamponnés en attente. Les trames ATIM unicast devraient être acquittées par les stations réceptrices. A la réception d'un acquittement à une trame ATIM envoyée, l'opération d'annonce est considérée réussie. Grâce à l'écoute de ces annonces, chaque noeud sera apte à décider de la nécessité de rester en mode actif ou de retourner au mode veille. Ainsi, les noeuds concernés par un échange de données restent actifs pendant la période suivant la fenêtre ATIM, en cas d'échange réussi d'annonces concernant des paquets unicast ou en diffusion. Alors que les noeuds n'ayant pas de données à recevoir ou à envoyer peuvent entrer en mode veille à l'expiration de la fenêtre ATIM afin de conserver de l'énergie.

Toutefois, le mécanisme 802.11 `PSM`, mode ad hoc, est conçu pour un réseau ad hoc à un saut (ou entièrement connexe). Pour l'application de ce mécanisme dans un réseau ad hoc multi sauts, on est confronté à certains problèmes. En premier, le 802.11 assume que tous les hôtes mobiles sont complètement connexes, la transmission d'une trame balise (beacon) peut servir pour synchroniser toutes les stations. Ainsi, toutes les stations peuvent être actives en même temps pendant la période ATIM. Alors que pour un environnement ad hoc multi sauts, la synchronisation est plus difficile à cause de l'imprévisibilité des délais des communications et des déplacements des hôtes, en particulier pour les réseaux de grande envergure. Deuxièmement, un noeud ne peut avoir connaissance de son voisinage que s'il intercepte des signaux de leur part

[17] ATIM : Announcement Traffic Indication Message

et qu'il émet un signal pour informer les autres hôtes de sa présence. Or, un noeud en mode conservation d'énergie se voit diminuer ses chances de transmission et d'interception des signaux provenant de la part de ses voisins. La transition à l'état inactif déconnecte le noeud du réseau et change par la suite la topologie de ce dernier. Ainsi, tout noeud peut avoir une information erronée de son voisinage, du moment qu'il peut ne pas détecter la présence des noeuds voisins opérant en mode conservation d'énergie. Cette situation pourrait générer des répercussions négatives sur les fonctionnalités de routage. Par la suite, le taux de gaspillage de la bande passante suite à l'occurrence de collisions est relativement important, en particulier lorsque la charge du réseau est élevée. En effet, lorsque le mécanisme `PSM` est activé, toutes les stations ne sont autorisées à émettre leurs paquets de données que durant la deuxième période de l'intervalle beacon, après avoir réussi à annoncer ces paquets durant la fenêtre ATIM. Des simulations réalisées dans [104] ont permis d'évaluer les performances du mécanisme `PSM` en fonction du nombre de stations actives. Les résultats obtenus montrent la dégradation des performances de ce mécanisme lorsque le nombre de stations s'accroît et l'augmentation de la probabilité de collisions entraînant un gaspillage de la bande passante.

En cas de charge faible, les noeuds impliqués dans un échange de paquets de données ou une opération de diffusion sont dans l'obligation de rester actif jusqu'à la fin de l'intervalle beacon courant, même après l'achèvement de cet échange. Le même traitement est infligé à toute station ayant émis un beacon, étant donnée que cette dernière doit rester active tout au long de l'intervalle beacon courant, même si elle n'est pas concernée par un échange de données éventuel.

Plusieurs améliorations du mécanisme `PSM` ont été proposées. Une amélioration du mécanisme `PSM` a été proposée dans [14] à travers le mécanisme `TA-PSM`[18] autorisant les stations actives durant la deuxième fenêtre à entrer en mode veille une fois les opérations d'échange la concernant achevées. D'autres approches ont joué sur la taille de la fenêtre ATIM, étant donné que le recours à la fenêtre ATIM gaspille une quantité considérable d'énergie pour un environnement ayant une charge faible [3, 4, 5, 15, 16, 17]. Dans cet axe, les auteurs de [70] proposent l'introduction d'une période d'écoute du canal T_{cs}, qui précède la fenêtre ATIM. Cette dernière permet à tout noeud d'informer ses voisins de son intention d'annoncer ou pas des paquets. Une deuxième période d'écoute CS_2 permet à un noeud de demander à ses voisins d'utiliser une fenêtre ATIM de taille statique. Une approche différente a été adoptée dans [62], où chaque noeud choisit de manière locale la taille de sa fenêtre ATIM en se basant sur l'observation de l'état et des conditions du réseau.

Le mécanisme, proposé par [2], opte pour la réduction du nombre d'annonces échangées afin de réduire la sous période relative à cette phase. Durant la fenêtre ATIM, les stations acquièrent une connaissance de l'état du voisinage (voisins actifs ou pas) grâce à l'interception des annonces ATIM et les acquittements associés. Les auteurs de [104] visent à réduire le nombre

[18] TA-PSM :Traffic Aware Power Saving Mode

de stations actives voulant accéder au médium durant un intervalle beacon afin de réduire le risque d'occurrence de collisions, contribuant ainsi à améliorer le débit et à assurer une meilleure conservation d'énergie. Les auteurs de [85] proposent de diviser le reste de l'intervalle beacon après la fenêtre ATIM en un certain nombre de slots de temps de longueur égale. Chaque noeud participant à la communication devient actif uniquement durant les slots de temps lui étant associés. Les autres noeuds peuvent rester en mode veille durant ces slots de temps qui ne leur sont pas attribués. L'allocation des slots est dynamique et les informations pour l'allocation des slots sont incluses dans les trames ATIM.

La troisième catégorie englobe les mécanismes de réveil asynchrones. Etant donné que chaque noeud peut adopter sa propre planification pour se réveiller, aucune synchronisation horloge n'est requise. Toutefois, un certain chevauchement entre les plannings des périodes d'activité correspondants à des noeuds voisins est nécessaire afin de permettre la communication. Les auteurs de [88] proposent trois protocoles asynchrones. L'idée de base est d'essayer de planifier les périodes d'activité des noeuds en mode conservation d'énergie (PS : Power saving) de manière à garantir que deux noeuds voisins soient capables de se détecter mutuellement en un temps fini. Par ailleurs, tout noeud en mode PS doit être capable de prédire la période d'activité d'un noeud en interceptant sa trame balise *beacon*. Le noeud doit être capable de dériver le schéma d'activité de l'émetteur du *beacon* en se basant sur la différence de leur horloge. Un des trois mécanismes a été étendu dans [58, 59]. Dans le même contexte, le protocole `PAMAS` [78] vise à remédier au problème lié à l'écoute de paquets de données destinés à un noeud voisin. Toute station désirant émettre des paquets de données doit effectuer un échange réussi de paquets RTS/CTS sur un canal de contrôle, alors que l'échange effectif des paquets de données est réalisé sur un deuxième canal de données. Grâce à l'écoute du canal de contrôle, tout noeud est apte à décider de la nécessité d'entrer en mode conservation d'énergie et à déterminer la durée de son inactivité.

Le mode conservation d'énergie a prouvé son utilité à minimiser la consommation d'énergie. Cependant, il affecte les performances du système. En effet, le passage à un état dormant réduit l'activité des noeuds, ce qui modifie la topologie du réseau. Ce changement est apte à influencer de manière négative la fonction de routage. Par ailleurs, pour pouvoir communiquer, tous les noeuds doivent se synchroniser pour assurer un réveil simultané ou planifier des périodes d'activité qui se chevauchent. La limitation de la période d'activité, durant laquelle tous les noeuds intéressés doivent être actifs, contribue à réduire le débit. Les critères de débit, de délai de séjour et de consommation d'énergie par paquet n'ont pas été pris en considération dans la majorité des approches présentées. La notion de clusterisation a été exploitée conjointement au mode veille par certaines approches pour ailler conservation d'énergie et acheminement des données. La présentation de ces approches fera l'objet de la section suivante.

4.1 Exploitation de la clusterisation pour la conservation d'énergie

D'autres approches ont eu recours à la technique de clusterisation afin d'assurer un acheminement convenable des paquets de données tout en assurant une certaine conservation d'énergie. Ces techniques de conservation d'énergie requièrent qu'un groupe de noeuds reste actif tout le temps afin d'assurer les fonctions de routage et de communication au profit des noeuds ordinaires pouvant adopter le mode veille. De cette manière, les noeuds ordinaires évitent les sources de gaspillage d'énergie et réduisent leur consommation d'énergie. Etant donnée l'activité continue des noeuds dominants, tout noeud n'a pas besoin d'annoncer le trafic destiné à ces derniers, réduisant ainsi le nombre de paquets à échanger. La majorité du trafic sera acheminé grâce aux noeuds dominants actifs, par conséquence on contribue à préserver la capacité du réseau.

Le mécanisme `GAF` [97] permet d'identifier les noeuds équivalents pouvant assurer les mêmes fonctionnalités de routage afin de permettre à certains noeuds d'entrer en mode inactif. L'objectif est de permettre aux noeuds d'arrêter momentanément leur interface radio afin d'éviter de dissiper de l'énergie lors de l'écoute passive du canal ou lors de l'interception de paquets destinés à des noeuds voisins, tout en assurant une communication ininterrompue entre les différents noeuds communicants. Le mécanisme `GAF` assume la disponibilité d'informations géographiques grâce à un système de positionnement, tel que GPS. Grâce à ses informations, on divise le réseau en des grilles carrées de taille fixe. La taille des grilles virtuelles est définie de manière à ce que les noeuds appartenant à deux grilles adjacentes peuvent communiquer. Chaque noeud est apte à déterminer sa grille d'attache grâce à son emplacement géographique et la taille de la grille. Au niveau de chaque grille, tous les noeuds sont équivalents d'un point de vue routage. Ces noeuds membres coopèrent afin de déterminer les noeuds pouvant entrer en mode inactif et la durée de leur repos. L'état de tous les noeuds alterne entre actif et découverte. Périodiquement, chaque noeud diffuse un message incluant son identifiant, l'identifiant de sa grille, son statut et le temps estimé pour rester actif (*enat*). Un protocole de découverte essaie d'assurer qu'un noeud reste actif au niveau de chaque grille afin d'assurer l'acheminement des paquets, tandis que les autres noeuds peuvent entrer en mode inactif. Ainsi, au niveau de chaque grille, tout noeud peut entrer en mode inactif lorsqu'il trouve un noeud équivalent actif apte à assurer la relève dans le transfert des paquets. Grâce à la valeur *enat* du noeud actif, les noeuds équivalents inactifs sont aptes à déterminer la durée de leur inactivité. A leur réveil, ces derniers retrouvent leur état de découverte. Dans un environnement mobile, le processus de découverte devient plus fréquent et la durée d'inactivité des noeuds devient plus courte avec la croissance de la vitesse. Dans un tel environnement, on assume que chaque noeud est capable d'estimer la durée au bout de laquelle il va quitter sa grille d'attache *engt*, cette information sera incluse dans son message de découverte. Ainsi, les noeuds inactifs vont adopter la valeur la plus petite entre *enat* et *engt* pour leur durée de sommeil.

Le mécanisme SPAN [23] opère différemment. Il se base sur l'élection de coordinateurs. Ces derniers doivent rester toujours actifs afin assurer l'acheminement des paquets et améliorer les performances du mécanisme de conservation d'énergie de 802.11 en terme de débit et de délai de livraison. Les autres noeuds opèrent en mode veille et vérifient périodiquement s'ils doivent se réveiller et devenir coordinateur. L'élection des coordinateurs se fait de manière distribuée en se basant sur informations locales collectées à travers un échange périodique de messages Hello. Tout noeud se déclare coordinateur s'il a deux voisins incapables de communiquer directement ou à travers un ou deux coordinateurs. Cette règle assure que la totalité du réseau est couverte par suffisamment de coordinateurs. La contribution apportée dans [23] consiste à introduire une nouvelle fenêtre *Advertised window*. Ainsi, l'intervalle beacon renferme trois fenêtres comme l'indique la figure (4.2) :

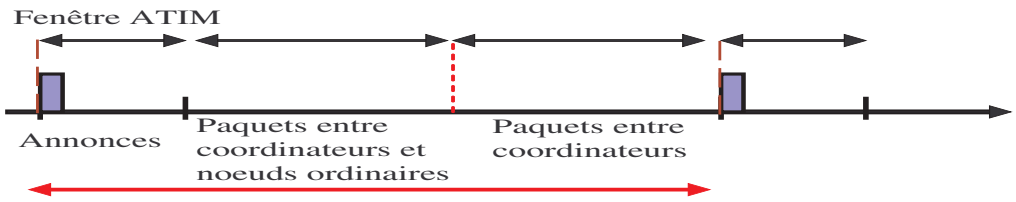


Figure 4.2 – Structure d'un intervalle beacon pour le mécanisme SPAN

La fenêtre ATIM est suivie d'une fenêtre durant laquelle uniquement les paquets annoncés et ceux destinés à un coordinateur sont envoyés. La dernière fenêtre est réservée pour l'échange des paquets entre coordinateurs uniquement. Pour cette approche, les mêmes règles adoptées pour le mécanisme PSM sont appliquées pour les transitions entre les états actif et inactif au niveau des noeuds ordinaires. Une légère modification consiste à autoriser les noeuds ordinaires à entrer en mode veille dès la fin de la 2^{me} fenêtre, étant donné que le trafic est majoritairement acheminé grâce aux services des noeuds coordinateurs actifs en permanence. Une autre amélioration a été apportée aux annonces ATIM. Le nombre de paquets en diffusion est indiqué dans l'annonce ATIM afin de permettre à tout noeud ordinaire d'entrer en mode conservation d'énergie suite à la réception de tous les paquets en diffusion, si aucune annonce unicast n'a été reçue. Par ailleurs, les paquets destinés aux noeuds coordinateurs ne sont pas annoncés au préalable.

AFECA [98] est un algorithme asynchrone distribué pour la construction d'une dorsale dédiée au routage où tout noeud décide de la nécessité de rester à l'écoute en se basant sur la taille de son voisinage. Tout noeud comptabilise le nombre de noeuds voisins, grâce à l'interception des transmissions dans son voisinage. Un noeud alterne entre l'inactivité et l'écoute du canal, en adoptant des périodes aléatoires d'inactivité proportionnelles au nombre de noeuds avoisinants. Indépendamment de la densité, le nombre de noeuds actifs est relativement constant. Ainsi, la conservation d'énergie s'améliore avec la croissance de la densité. Les paramètres de AFECA

sont choisis de manière à avoir une probabilité élevée assurant que les noeuds actifs forment un ensemble dominant connecté pour assurer l'acheminement des données.

Des approches similaires ont été élaborées pour adresser le problème d'énergie dans les réseaux de capteurs. Parmi ces propositions, le mécanisme `S-MAC` [99] repose sur la division équitable du temps en larges trames. Chaque trame possède deux parties : une partie active et une partie d'inactivité. Durant la période d'inactivité, tout noeud désactive son interface radio afin de réduire le gaspillage d'énergie due à l'écoute passive du canal. Durant la période active, il est apte à communiquer avec son voisinage et envoyer les paquets en instance. Pour cet effet, les noeuds voisins forment des clusters virtuels afin de s'auto synchroniser durant les périodes d'inactivité. Afin de pouvoir communiquer malgré l'existence de planifications différentes, les noeuds diffusent leurs planifications pour informer leurs voisins immédiats.

Le protocole `LEACH` [45] est aussi destiné pour les réseaux de capteurs. Il permet à son tour de sélectionner des clusterheads afin de collecter des informations locales et de les faire parvenir à une station de base dans un réseau de capteurs. L'idée de base est que les capteurs à proximité d'un clusterhead sont aptes à communiquer avec ce dernier à une puissance plus réduite. Les clusterheads auront la charge d'agréger les données collectées et de les transmettre à la station de base en ayant recours à une puissance de transmission plus élevée. Ce mécanisme assure la rotation du rôle de clusterhead entre les différents noeuds. En effet, le temps est divisé en périodes. Chaque noeud aura à assurer le rôle de clusterhead chaque $\frac{1}{p}$ périodes en moyenne, p étant un paramètre du système. Tout clusterhead, nouvellement élu, diffuse à tous les voisins membres de son cluster un ordonnancement TDMA pour les transmissions et un canal CDMA afin d'éviter les interférence avec les clusters adjacents.

Dans la section 2, nous mettons en évidence l'importance accordée au choix du mécanisme de clusterisation, à travers l'évaluation de l'impact du mécanisme de clusterisation sur les performances de l'approche de conservation d'énergie et les performances du réseau.

4.2 Impact de l'approche de clusterisation sur l'efficacité du mécanisme de conservation d'énergie

La section précédente nous a permis d'étaler brièvement les approches ayant essayé d'exploiter conjointement les apports du mode veille et de la clusterisation. En effet, maintenir une structure virtuelle constamment active semble une solution prometteuse pour réduire la consommation d'énergie tout en préservant la capacité du réseau. Elle permet aussi d'optimiser l'utilisation des ressources limitées en évitant l'écoute passive du canal et l'interception de paquets à délivrer aux voisins pour les noeuds ordinaires pouvant switcher au mode inactif. Ainsi, le choix des noeuds dominants revêt une grande importance. Pour cette raison, le choix du mécanisme de clusterisation s'avère critique, étant donnée que l'efficacité de ce dernier aura un impact direct sur

les performances du mécanisme de conservation d'énergie reposant sur l'infrastructure virtuelle générée. La phase de clusterisation requiert un certain temps pour la formation des clusters ou de l'ensemble dominant connexe. Elle introduit aussi un certain trafic de contrôle additionnel dédié à cette étape. Etant donnée que la plupart des approches de clusterisation sont distribuées, tout noeud doit collecter certaines informations sur son voisinage à h sauts (pour un h petit) afin de pouvoir décider de manière locale du rôle à assurer.

Par ailleurs, cette structure aura le fardeau des tâches additionnelles, telles que la recherche des routes et l'acheminement des données. Le maintien de cette structure actualisée nécessite un échange périodique de messages de contrôle, dans la majorité des approches existantes. L'introduction d'un tel trafic additionnel peut être néfaste pour le trafic de données dans un environnement opérant en mode conservation d'énergie. Ces messages dédiés à la maintenance de la structure vont puiser à leur tour des ressources limitées en terme de bande passante et d'énergie. En effet, chaque paquet de contrôle doit être annoncé au préalable durant la fenêtre ATIM afin d'être traité par tous les voisins immédiats de l'émetteur lors de sa diffusion. Ces annonces supplémentaires peuvent être pénalisantes dans un environnement dense où elles peuvent bloquées ou différées les annonces relatives à des paquets de données. Les messages de contrôle à leur tour vont concurrencer les paquets de données pour l'accès au canal durant la période active limitée. De ce fait, une certaine portion de la bande passante va être puisée par le trafic de contrôle introduit par la clusterisation et les annonces associées. Par conséquence, plus d'énergie va être gaspillée pour la transmission des paquets de contrôle et de leurs annonces, en particulier pour les réseaux denses et de grande envergure. D'autre part, à l'expiration de la fenêtre ATIM, l'attente de la diffusion des messages de contrôle peut obliger aussi certains noeuds à rester en mode actif sans avoir des paquets de données à émettre ni à recevoir.

Dans cette section, notre objective est de mettre en évidence l'impact du mécanisme de clusterisation adopté sur l'efficacité de l'approche de conservation d'énergie utilisée. Pour cet effet, on s'est proposé d'intégrer notre mécanisme de clusterisation dans une approche de conservation d'énergie existante se basant sur le concept d'ensemble dominant. Dans ce contexte, on va comparer les performances du mécanisme `SPAN` et notre mécanisme baptisé `SPAN+`. Le mécanisme `SPAN` se base sur l'élection d'un sous ensemble de noeuds pour assurer le rôle de coordinateurs. Ces derniers seront impliqués dans les tâches de routage et de communication. Pour cet effet, tous les noeuds coordinateurs doivent rester actif de manière continue, tandis que les autres noeuds ordinaires peuvent adopter le mode conservation d'énergie. L'apport de ce mécanisme se concrétise à travers l'introduction d'une nouvelle fenêtre, nommée *Advertised Window*, dédiée à l'échange de paquets entre noeuds ordinaires et coordinateurs. Le reste de l'intervalle *beacon* sert à l'échange des paquets entre noeuds coordinateurs. Durant cette période, les noeuds ordinaires entrent en mode veille. Etant donné que les noeuds coordinateurs sont toujours actifs, les paquets destinés à de tels noeuds ne sont pas annoncés au préalable, contrairement aux paquets destinés

à des noeuds ordinaires.

Dans ce qui suit, on va détailler le nouveau mécanisme hybride SPAN+, avant de présenter l'évaluation de performances des deux mécanismes SPAN et SPAN+ [56].

4.2.1 Mécanisme SPAN+

Le mécanisme SPAN+ fait référence à l'approche de conservation d'énergie introduite par le mécanisme SPAN, reposant sur la dorsale formée par l'ensemble dominant connexe généré grâce à notre mécanisme TBCA. Outre l'allégement du trafic de contrôle associé à l'établissement et la maintenance d'une infrastructure virtuelle, la rapidité de la finalisation du processus de clusterisation constitue l'un des apports essentiels de notre mécanisme TBCA, ce dernier achève la construction de l'ensemble dominant couvrant la totalité du réseau en un temps très réduit. Grâce à cette caractéristique, cette infrastructure virtuelle pourrait être déployée rapidement pour le support des algorithmes de haut niveau. Dans le contexte de la conservation d'énergie dans un environnement ad hoc, cette caractéristique peut être exploitée pour atténuer l'impact des paquets de contrôle requis pour le processus d'établissement et de maintien de la structure de clusterisation sur l'échange des annonces et des paquets de données. Ainsi, le recours à notre mécanisme TBCA nous accorde le privilège de réserver une sous période au début de la fenêtre ATIM pour finaliser le processus de la clusterisation, avant de reprendre l'envoi normal des annonces relatives aux paquets en attente. On rappelle que tout noeud est apte à déterminer l'instant de la reprise de son fonctionnement normal grâce aux formules (2.29 ou 2.30). Ainsi, la réservation d'une partie de la fenêtre ATIM sera périodique lors du déclenchement du mécanisme TBCA pour assurer la maintenance de l'infrastructure virtuelle, comme le montre la figure (4.3.a). La possibilité d'étendre la taille de la fenêtre ATIM durant l'intervalle beacon, où le processus de clusterisation est relancé, nous permet d'assurer un échange convenable des annonces entre noeuds voisins. Chaque noeud est apte à calculer cette extension, identique pour tous les noeuds, afin de garder la synchronisation entre eux. Le calcul de cette extension se fait conformément à la formule suivante :

$$extension\_atim = (3 + Nbr\_T_{iGW} + Nbr\_T_{iGW}\_sup + 4) * T_i \tag{4.1}$$

Cette extension inclut un nombre fixe de périodes T_i, à savoir le nombre de périodes T_i à attendre avant d'initier la phase de déclaration de gateways, le nombre de périodes T_i requis durant la phase de déclaration de gateways, le nombre de périodes T_i nécessaires durant la phase de vérification et le nombre de périodes T_i à attendre pour éviter la réalisation de la phase de vérification par deux clusterheads de niveaux adjacents. On ne peut pas étendre la fenêtre ATIM avec le nombre total de périodes T_i requis pour la clusterisation de tout le réseau afin de laisser plus temps pour l'échange des paquets de données.

Pour les autres intervalles beacon, la fenêtre ATIM sera totalement dédiée à l'échange d'annonces concernant les paquets en attente préservant ainsi sa taille originale, comme l'illustre la figure (4.3.b). Durant le processus de clusterisation, on autorise l'échange uniquement des pa-

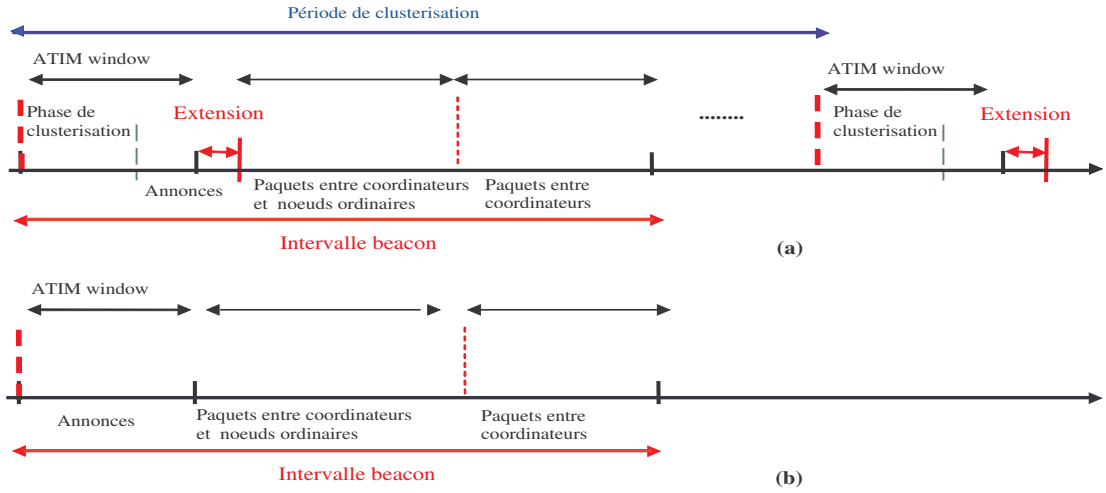


Figure 4.3 – (a) Structure de l'intervalle beacon du mécanisme `SPAN+` lorsque la clusterisation est déclenchée (b) Structure de l'intervalle beacon normal du mécanisme `SPAN+`

quets de contrôle associés à cette phase, le traitement des autres types de paquets est suspendu momentanément. Ainsi, on évite les collisions éventuelles entre les paquets dédiés à la clusterisation et les annonces ATIM à envoyer lors de l'activation du mode conservation d'énergie. Par ailleurs, l'achèvement rapide de la clusterisation du réseau permet de nous libérer des envois additionnels d'annonces relatives aux messages de contrôle dédiés à la clusterisation durant la fenêtre ATIM. Plus de ressources sont ainsi disponibles pour écouler les paquets de données en attente, étant donné qu'aucun paquet de contrôle dédié à la clusterisation ne va concurrencer avec les paquets de données annoncés avec succès durant la fenêtre *Advertised window*.

4.2.2 Evaluation de performances

A travers cette évaluation de performance, nous visons à vérifier l'influence du mécanisme de clusterisation sur l'efficacité de l'approche de conservation d'énergie et sur les performances du réseau. Afin de mettre en valeur l'impact du coût associé à la clusterisation en terme de temps d'exécution et de trafic de contrôle induit, on va annuler le coût induit par le routage. Pour cette fin, on opte pour l'acheminent géographique, décrit au niveau du chapitre 3. L'exploitation de la notion de dominance de graphe pour le routage fera l'objet du chapitre 6. Les deux mécanismes de conservation d'énergie, `SPAN` et `SPAN+`, ont été implémentés sous le simulateur JSIM. Pour comparer leurs performances, on a focalisé essentiellement sur les critères suivants :

- Le nombre moyen de paquets délivrés avec succès pour toutes les destinations durant la totalité de la simulation
- Le délai de séjour moyen correspond au temps mis par un paquet de données sur le réseau, depuis sa génération par le noeud source jusqu'à sa réception par le noeud destination finale
- L'énergie résiduelle moyenne fait référence à la moyenne de l'énergie restante au niveau de tous les noeuds à la fin de la simulation
- L'énergie moyenne consommée par paquet représente l'énergie moyenne consommée par tous les noeuds divisée par le nombre de paquets délivrés avec succès

On note que l'énergie initiale disponible au niveau de chaque noeud est de 1000 watts. Les taux de consommation d'énergie adoptés sont conformes aux valeurs spécifiées dans la table suivante (4.1) :

Transmission	Réception	Ecoute passive	Inactif
0.660	0.395	0.296	0.130

Table 4.1 – Taux de consommation de l'énergie (watts)

La table suivante résume les paramètres adoptés durant cette phase de simulation (4.2) :

Paramètres du réseau	
Portée	250m
Bande passante	2 Mbps
Durée de simulation	500 secondes
Paramètres de `TBCA`	
Périodicité de MaJ de la clusterisation	10 intervalles beacon
(D_{1max}, D_{1min})	(31, 25)
(D_{1pmin}, D_{1npmin})	(15, 31)
$NbrMax\_gw\_CH$	5
$NombreMoy\_gw/CH$	4
Nombre de périodes T_i successives en tant que dominant	5

Table 4.2 – Paramètres utilisés dans la simulation

Une suite de simulations a été conduite en considérant différentes topologies fixes afin de mettre en évidence l'impact du coût induit par l'utilisation d'un mécanisme de clusterisation pour générer et maintenir l'infrastructure virtuelle sur les performances de l'approche de conservation d'énergie et celles du système. Le facteur de mobilité n'a pas été considéré lors de cette phase d'évaluation, étant donné l'aptitude de notre mécanisme de clusterisation à s'adapter rapidement aux changements topologiques et à générer rapidement une dorsale, ce qui nous autorise à re déclencher le processus de clusterisation aussi fréquemment que possible.

Phase 1 : Impact du mécanisme de clusterisation sur l'approche de conservation d'énergie

La clusterisation est considérée comme une approche prometteuse pour remédier au problème de passage à l'échelle dans un environnement ad hoc, étant donnée que la structuration du réseau permet de rendre son exploitation plus aisée. Pour cette raison, on a simulé des réseaux denses à travers différentes topologies de 200 noeuds uniformément répartis dans une aire de simulation de $500 \times 500m$. Les simulations conduites durant cette phase visent à mettre en évidence l'impact du mécanisme de clusterisation sur les performances de l'approche de conservation d'énergie.

L'inconvénient des approches de clusterisation existantes est le coût induit pour l'établissement et la maintenance de l'infrastructure générée. Durant cette première étape, notre but est de valoriser les répercussions éventuelles sur les performances de l'approche de conservation d'énergie suite à l'introduction d'un tel trafic additionnel nécessaires pour maintenir actualisée l'infrastructure virtuelle. Pour cette fin, on fait varier la périodicité de mise à jour relative à l'échange périodique de messages hello pour le mécanisme `SPAN` et marquant le re-lancement du processus de clusterisation pour le mécanisme `SPAN+`. On maintient la même charge injectée sur le réseau, où chaque source émet à un taux de 20 pkt/sec. On vise à simuler un trafic pouvant transiter sur un réseau ad hoc de grande envergure.

Durant cette première phase, les valeurs définies dans le mécanisme `PSM` ont été adoptées pour la taille de l'intervalle beacon et de la fenêtre ATIM. Ainsi, on a fixé la taille de l'intervalle beacon à 0.100 secondes et la taille de la fenêtre ATIM à 0.008. Pour les mécanismes `SPAN` et `SPAN+`, deux fenêtres de taille égale ont été adoptées pour la fenêtre P_1, dédiée au trafic à échanger entre noeuds ordinaires et coordinateurs, et P_2, consacrée exclusivement aux échanges entre noeuds dominants. On a fixé leur taille à 0.046 secondes.

Les mesures du nombre de paquets délivrés avec succès en fonction de la périodicité de mise à jour, exposées dans la figure (4.4), confirment que le coût de la clusterisation peut affecter considérablement la capacité du réseau. On constate que les performances de `SPAN` ne s'améliorent que lorsque la fréquence de l'échange périodique des messages hello est réduite. En espaçant les mises à jour, on réduit le nombre d'annonces additionnelles requises avant toute diffusion de paquet hello et on restreint aussi les paquets de contrôle en compétition avec les paquets de données pour l'accès au canal durant la fenêtre *Advertised window*. Dans un réseau ad hoc dense ou à grande envergure, le trafic de contrôle puise des ressources limitées en terme de bande passante et d'énergie, il contribue aussi à surcharger le réseau. Le recours à ce trafic de contrôle peut anéantir les apports de la clusterisation. Le mécanisme `SPAN+` de son côté maintient un débit relativement constant malgré la variation de la périodicité de mise à jour. Cette stabilité s'explique par le changement du mécanisme de clusterisation. Le mécanisme `TBCA` nous a permis d'alléger le réseau des messages de contrôle dédiés à la phase de clusterisation. Ainsi, plus de

ressources sont disponibles pour le trafic de données. Par ailleurs, la variation de la périodicité de mise à jour n'affecte pas de manière notable le débit, étant donné que le temps requis pour finaliser la clusterisation est très insignifiant.

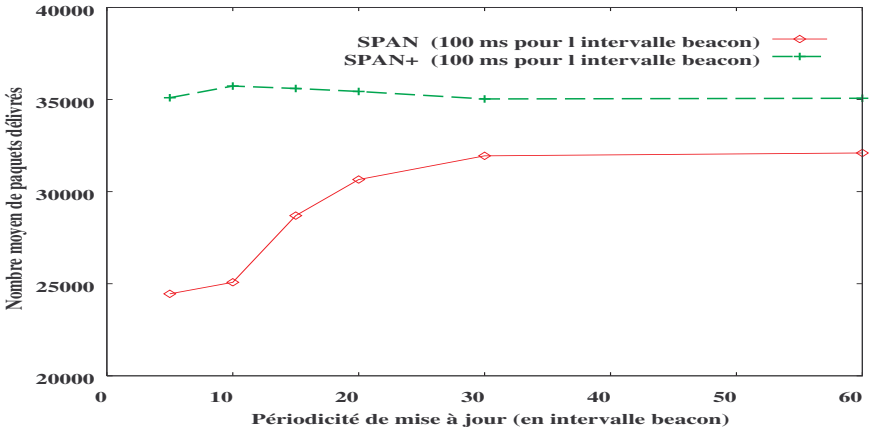


Figure 4.4 – Nombre moyen de paquets délivrés avec succès en fonction de la périodicité de mise à jour

Ce trafic additionnel peut aussi bloquer le trafic de données. On rappelle que tout paquet de contrôle doit être annoncé au préalable pour informer le voisinage, en particulier les noeuds ordinaires opérant en mode conservation d'énergie. Lorsque les mises à jour deviennent de plus en plus fréquentes, le nombre de paquets de contrôle croît considérablement, sans négliger les annonces associées à ces derniers. Ces annonces peuvent encombrer le canal et retarder la transmission des annonces relatives aux paquets de données. Ainsi, l'envoi des paquets de données peut être différé, ce qui induit des délais d'attente augmentant le délai de séjour, comme on peut le remarquer pour le mécanisme `SPAN` à travers la figure (4.5). Par ailleurs, le délai de séjour constaté pour le mécanisme `SPAN+` est relativement inférieur à celui de `SPAN`. Pour ce mécanisme, la réservation d'une période au début de l'intervalle beacon pour finaliser la phase de clusterisation nous permet d'éviter l'ajout d'annonces additionnelles , du moment qu'aucun paquet de contrôle utilisé par le mécanisme de clusterisation ne va concourir pour l'accès au canal durant la fenêtre P_1 conjointement avec les paquets de données.

Les mesures suivantes s'intéressent au critère le plus important pour une approche de conservation d'énergie, à savoir l'énergie. Les résultats de la figure (4.6) illustrent l'énergie résiduelle moyenne.

La différence remarquable entre les résultats fournis par les deux mécanismes s'explique toujours à travers le mécanisme de clusterisation adopté. En effet, en optant pour le rafraîchissement fréquent des informations concernant le voisinage à 2 sauts, le mécanisme `SPAN` contribue à surcharger le réseau par les paquets hello et les annonces associées. Les transmissions de ces paquets gaspillent de l'énergie. De son côté, le mécanisme `SPAN+` permet d'alléger le réseau d'un

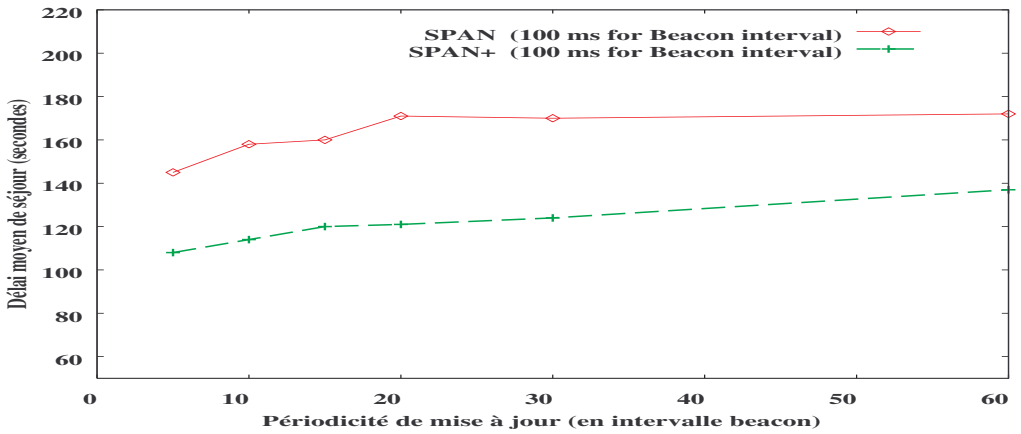


Figure 4.5 – Délai de séjour moyen en fonction de la périodicité de mise à jour

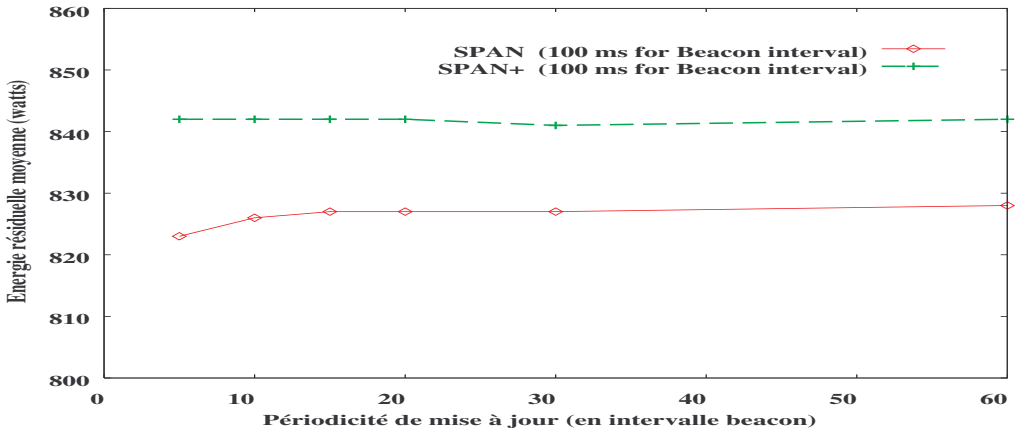


Figure 4.6 – Energie résiduelle moyenne en fonction de la périodicité de mise à jour

tel trafic de contrôle préservant par conséquence plus d'énergie. Ainsi, SPAN+ ne gaspille pas ses ressources en terme de bande passante et d'énergie pour la transmission de messages de contrôle et des annonces associées. Pour cette raison, l'énergie consommée par paquet mesurée pour SPAN+ est toujours réduite par rapport à celle notée pour le mécanisme SPAN, comme le montre la figure (4.7). Toutefois, une amélioration de ce critère est constatée pour le mécanisme SPAN lorsque la fréquence de mise à jour est réduite. Espacer les envois périodiques des messages hello permet de limiter l'impact négatif du trafic de contrôle. Ces résultats consolident l'aptitude de SPAN+ à mieux préserver l'énergie et à écouler plus de paquets grâce au mécanisme de clusterisation déployé.

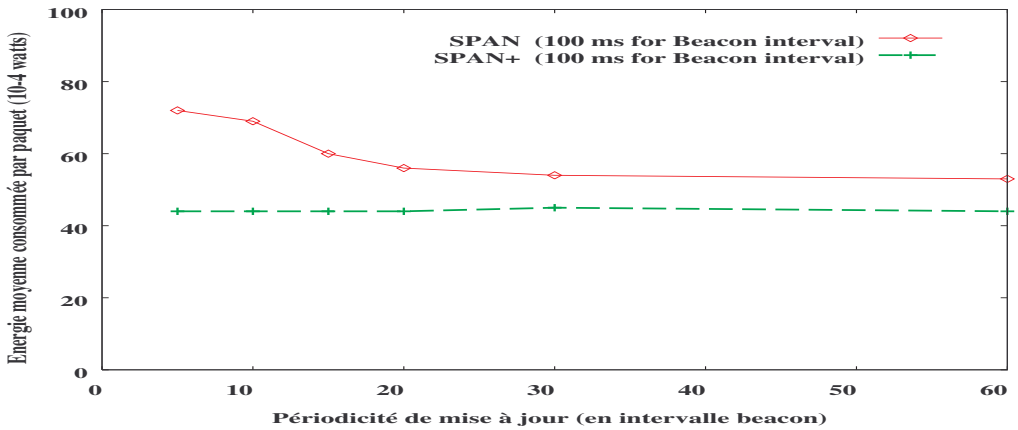


Figure 4.7 – Energie consommée par paquet en fonction de la périodicité de mise à jour

Vu que les tailles de l'intervalle beacon et de la fenêtre ATIM affectent considérablement les performances du système, on leur a attribué différentes tailles tout en variant la charge injectée sur le réseau. Durant cette deuxième étape, les tailles par défaut adoptées pour l'intervalle beacon et la fenêtre ATIM sont respectivement 100 ms et 8 ms. Les mêmes expérimentations ont été réalisées avec un intervalle beacon étendu de taille 300 ms et une fenêtre ATIM de 20 ms. Pour les fenêtres P_1 et P_2 spécifiques aux mécanismes SPAN et SPAN+, on a fixé leur taille à 46ms, puis à 140 ms. La périodicité de mise à jour est de 10 intervalles beacon.

Etendre la taille de la fenêtre ATIM et de l'intervalle beacon permet de laisser plus de temps pour l'annonce et le traitement des paquets de données. Ceci est concrétisé à travers l'amélioration perçue au niveau du nombre de paquets délivrés avec succès pour les deux mécanismes, comme on peut l'apercevoir à travers la figure (4.8). L'accroissement notable du débit, pour le mécanisme SPAN en particulier, est justifié par le fait qu'on a accordé plus de temps pour le traitement d'un nombre plus important de paquets. Ainsi, la présence et la compétition des messages hello et les annonces associées conjointement avec les autres ne perturbe pas énormément le fonctionnement du réseau. Cependant, on observe uniquement une légère augmentation du nombre de paquets

délivrés avec succès pour le mécanisme `SPAN+`. Ce dernier ne repose pas sur un échange périodique de messages de contrôle pour la maintenance de la dorsale, aptes à ralentir les paquets de données. Malgré cette légère amélioration, les performances de `SPAN+` surpasse toujours celles de `SPAN`.

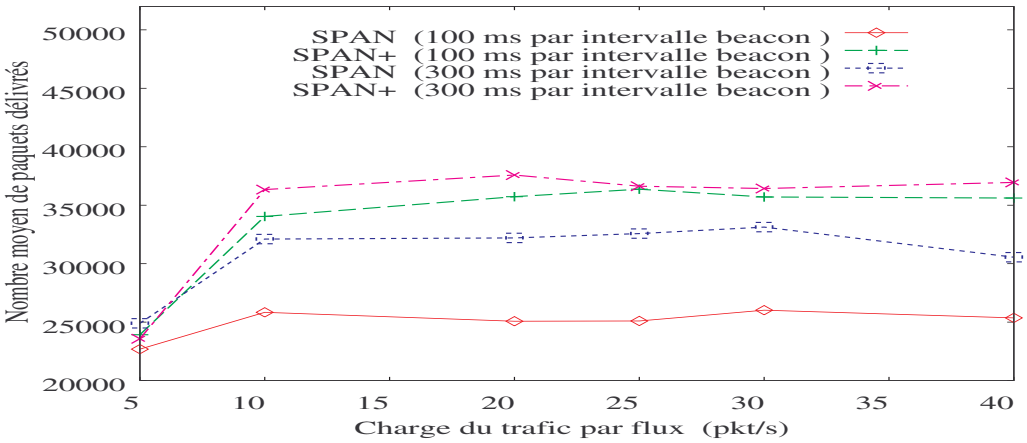


Figure 4.8 – Nombre de paquets délivrés avec succès en fonction de la charge du trafic par flux tout en variant la taille de l'intervalle beacon

L'augmentation de la taille des différentes fenêtres semble bénéfique en terme de débit. Toutefois, un paquet, généré après l'expiration de la fenêtre ATIM et destiné à une destination non avisée, doit être différé pour l'intervalle beacon suivant. La concurrence croissante peut obliger le noeud à repousser l'envoi de certains paquets pour l'intervalle suivant, même si ces derniers ont été préalablement annoncés avec succès. Vu que la taille de l'intervalle beacon a été agrandie, le temps d'attente au niveau des files d'attente et le délai de séjour s'accroissent par conséquence. La figure (4.9) expose les résultats consolidant ce raisonnement.

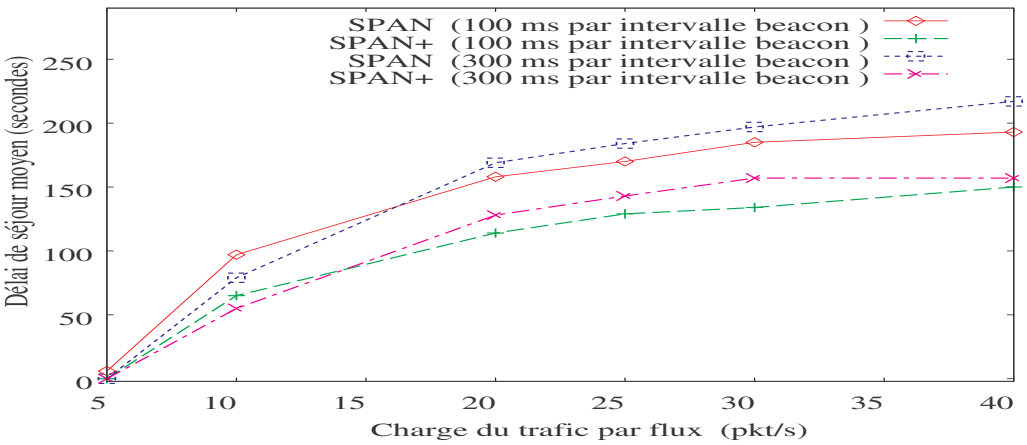


Figure 4.9 – Délai de séjour moyen en fonction de la charge du trafic par flux tout en variant la taille de l'intervalle beacon

Malgré l'amélioration perçue en terme de débit, on conserve approximativement les mêmes courbes relatives à l'énergie résiduelle pour les deux mécanismes, comme on peut le noter à travers la figure (4.10). Ainsi, les deux mécanismes permettent d'écouler plus de trafic tout en maintenant la même consommation d'énergie, grâce à l'agrandissement adopté pour la taille de la fenêtre ATIM et de l'intervalle beacon. Par conséquence, on arrive à annoncer plus de paquets de données et à acheminer plus de paquets de données suite à l'agrandissement des fenêtres P_1 et P_2. Ce comportement justifie la réduction du coût du paquet en terme d'énergie consommée, observée au niveau de la figure (4.10). Ce gain est mieux visible pour le mécanisme `SPAN`, toutefois, le gain est relativement réduit pour le mécanisme `SPAN+`. Ce dernier a acquis uniquement une légère croissance au niveau du débit suite aux changements ayant affectés les tailles des différentes fenêtres de l'intervalle beacon. Le gain s'accroît en fonction de la charge pour les deux mécanismes, étant données que les périodes d'activité sont mieux exploitées pour l'acheminement du trafic.

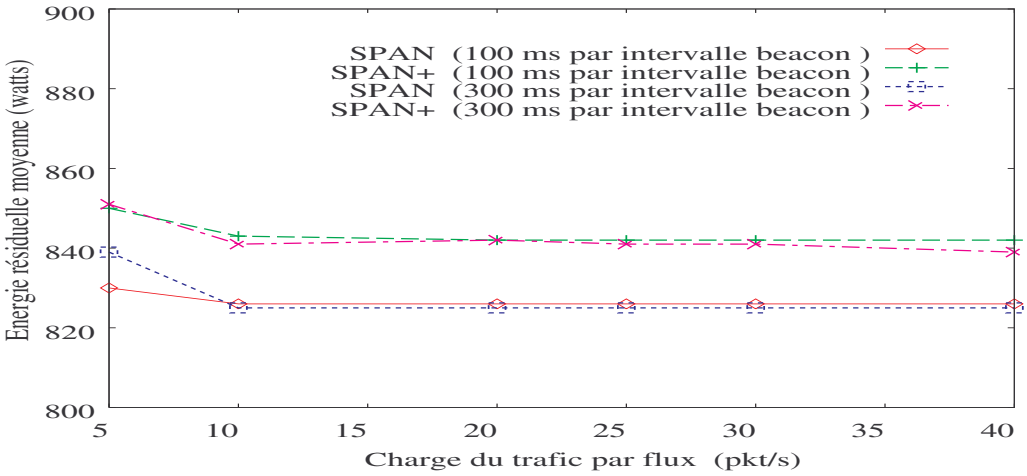


Figure 4.10 – Energie résiduelle moyenne en fonction de la charge du trafic par flux tout en variant la taille de l'intervalle beacon

Phase 2 : Impact du mécanisme de clusterisation sur les performances du système

En activant le mode conservation d'énergie, on contribue à réduire l'activité de certains noeuds même en ayant recours au service d'une dorsale, ce qui peut avoir des répercussions négatives sur les performances du réseau. Dans cette étape d'évaluation, on vise à mettre en évidence les répercussions éventuelles lors de l'activation du mode conservation d'énergie sur les performances globales du système. Notre but est de montrer la capacité du mécanisme de clusterisation déployé conjointement avec le mode veille à améliorer ces performances. Dans ce contexte, on compare les performances des mécanismes de clusterisation dans un environnement de charge variable, opérant avec et sans le mode conservation d'énergie. Le choix a été porté

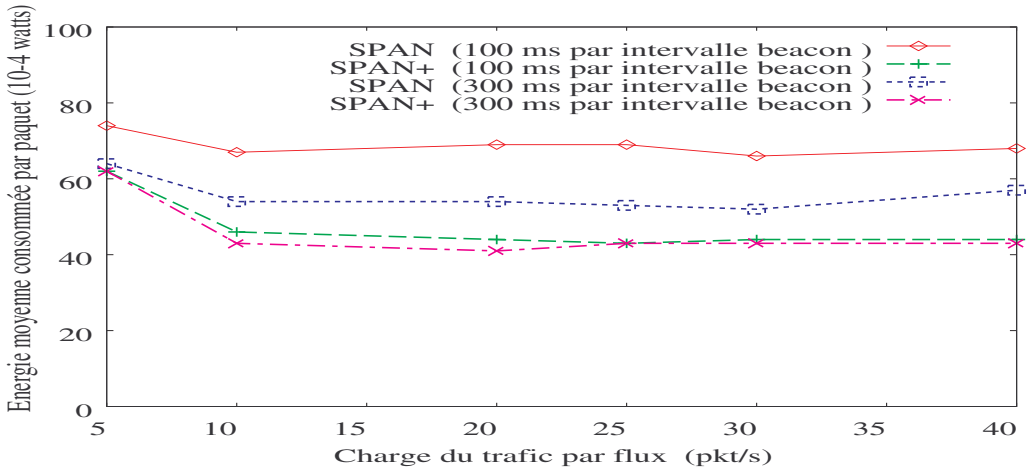


Figure 4.11 – Energie consommée par paquet en fonction de la charge du trafic par flux tout en variant la taille de l'intervalle beacon

sur des topologies de 200 noeuds uniformément distribués dans un espace de $500 \times 500m$. Les caractéristiques du trafic échangé sont identiques à celles adoptées durant les étapes précédentes. La périodicité de mise à jour des structures virtuelles a été fixée à 10 intervalles beacon, sachant que les tailles des différentes fenêtres sont conformes à celles fixées pour le cas d'un intervalle beacon de 300 ms durant la phase d'évaluation précédente.

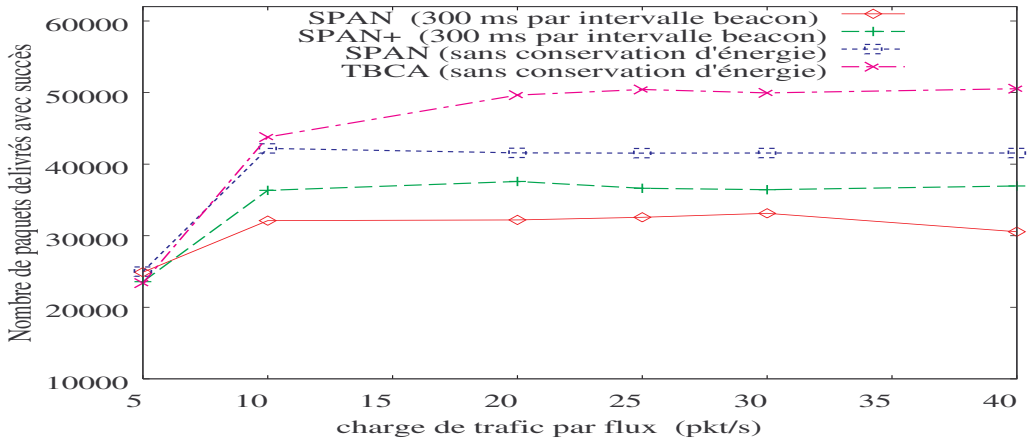


Figure 4.12 – Nombre moyen de paquets délivrés avec succès en fonction de la charge par flux dans un environnement avec et sans conservation d'énergie

Les mesures concernant le nombre moyen de paquets délivrés avec succès confirment qu'on doit tolérer une certaine baisse des performances du réseau afin de pouvoir conserver de l'énergie, comme le montre la figure (4.12). Cette baisse est clairement lisible à travers la diminution remarquable du débit lors de l'activation du mode conservation d'énergie pour les deux méca-

nismes `SPAN` et `SPAN+`. Cette détérioration se justifie par la réduction de la période active durant laquelle les noeuds sont autorisés à échanger les messages en attente. Les noeuds ordinaires ne peuvent transmettre les paquets bufférisés que durant la fenêtre P_1, alors que la fenêtre P_2 est dédiée exclusivement aux échanges entre noeuds coordinateurs. Malgré cette dégradation du débit, le mécanisme `SPAN+` permet de faire écouler plus de paquets que le mécanisme `SPAN`. La différence entre les deux mécanismes de conservation d'énergie `SPAN` et SPAN+ réside au niveau du mécanisme déployé pour assurer l'établissement et le maintien de l'ensemble dominant connexe opérant comme dorsale. Le recours à des échanges périodiques de messages hello pour le mécanisme `SPAN` contribue à étouffer le trafic de données. Outre les paquets de contrôle dédié aux tâches de routage, les messages Hello et les annonces ATIM relatives à ces derniers constituent un trafic additionnel pouvant pénaliser le trafic de données. Cette dégradation s'accentue dans le cas des réseaux denses ou de grande envergure, étant donné que le nombre des annonces et des messages Hello va s'accroître en fonction du nombre de noeuds formant le réseau. De son côté, le mécanisme `SPAN+` permet d'allouer plus de ressources au profit des paquets de données en déchargeant le réseau des messages relatifs à la phase de clusterisation. L'échange de ces derniers est limité à une petite période de temps, évitant ainsi la concurrence entre ces paquets et les paquets de données.

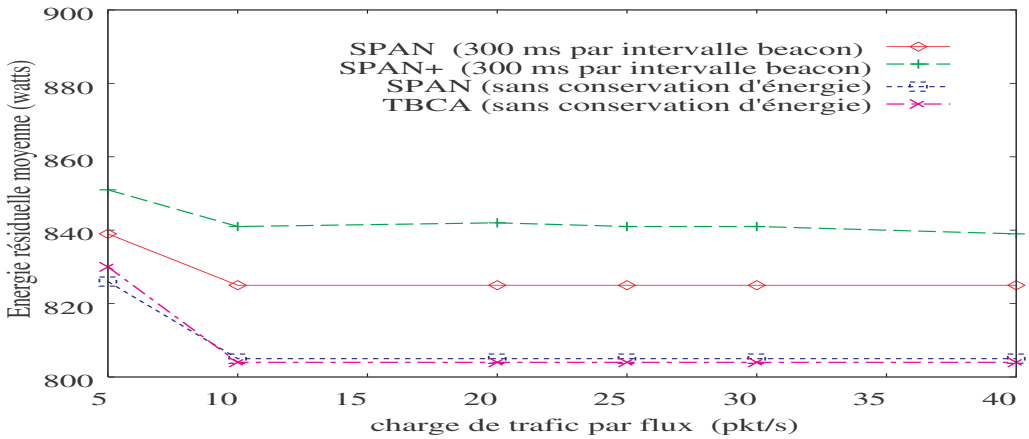


Figure 4.13 – Energie résiduelle moyenne en fonction de la charge par flux dans un environnement avec et sans conservation d'énergie

Malgré la détérioration du débit, les deux mécanismes parviennent à conserver de l'énergie, comme on peut le noter clairement au niveau de la figure (4.13). Le recours aux services d'une dorsale, formée par un sous ensemble de noeuds dominants, autorise les noeuds ordinaires à éviter le gaspillage de l'énergie due à l'écoute passive du canal et l'interception de paquets destinés à des noeuds voisins. Grâce à ce fonctionnement, les noeuds arrivent à réduire leur consommation d'énergie, étant donnée que la majorité du trafic sera acheminé par les noeuds coordinateurs. Pour éviter l'épuisement des ressources au niveau des noeuds dominants, les deux

mécanismes fournissent des règles assurant la rotation du rôle de dominant entre les différents noeuds de manière égalitaire, en privilégiant les noeuds ayant une abondance de ressources en terme d'énergie. Toutefois, le mécanisme `SPAN+` apporte un gain en énergie meilleur que celui de `SPAN` tout en assurant un meilleur taux de délivrance de trafic. Ainsi, le mécanisme `SPAN+` permet de préserver un coût par paquet en terme d'énergie consommée relativement identique à celui mesuré pour un environnement opérant sans mode de conservation d'énergie, lorsque la charge est relativement élevé. En présence d'une faible charge, le mécanisme `SPAN+` assure un coût meilleur à celui noté dans un environnement sans conservation d'énergie, comme on peut le remarquer à travers la figure (4.14). L'augmentation de l'énergie consommée par paquet pour le mécanisme `SPAN` est prévue, suite à la dégradation considérable du débit par rapport à un environnement sans conservation d'énergie.

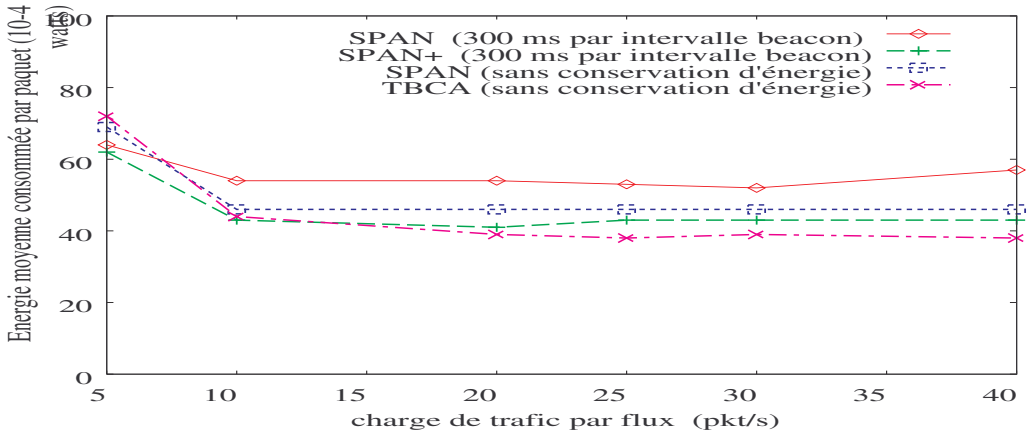


Figure 4.14 – Energie consommée par paquet en fonction de la charge par flux dans un environnement avec et sans conservation d'énergie

La figure (4.15) relative au délai de séjour moyen consolide le fait que les performances du réseau sont affectées lors du déploiement d'une approche de conservation, même si elle repose sur une dorsale constamment active pour assurer les tâches additionnelles.

Une augmentation du délai de séjour est constatée pour les deux mécanismes de conservation d'énergie `SPAN` et `SPAN+`. En effet, tout noeud doit annoncer certains paquets avant de pouvoir les transmettre, sans oublier que la période active où un noeud donné est autorisé à transmettre les paquets annoncés ou destinés à un coordinateur est limitée. Durant la fenêtre P_1, uniquement les échanges entre noeuds ordinaires et coordinateurs sont effectués en ayant recours à l'algorithme de backoff, tandis que la fenêtre P_2 est réservée exclusivement aux échanges entre noeuds coordinateurs. Ces contraintes peuvent contribuer à différer le traitement et l'envoi de certains paquets pour l'intervalle beacon suivant. Une telle situation peut survenir si le noeud n'arrive pas à annoncer le paquet en question, ou si le noeud est incapable à transmettre un

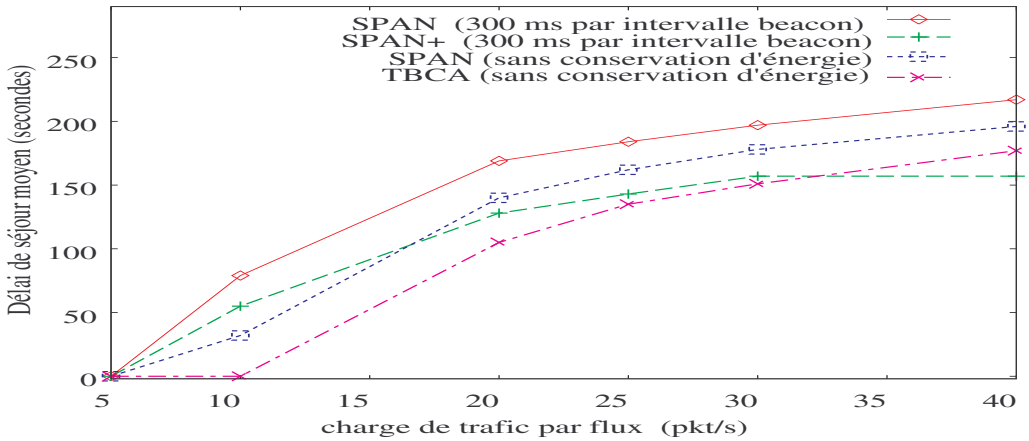


Figure 4.15 – Délai de séjour moyen en fonction de la charge par flux dans un environnement avec et sans conservation d'énergie

paquet, même annoncé, durant la période active à cause de l'occupation permanente du support.

On va essayer d'illustrer les situations possibles pouvant affecter ce critère à travers des exemples. On considère le premier exemple d'un noeud B ordinaire. La figure (4.16) illustre les différents instants possibles de génération d'un paquet. La génération d'un paquet dans cette explication fait référence à l'arrivée du paquet en question au niveau de la file d'attente pour être envoyé. La première situation concerne tout paquet, destiné à un noeud ordinaire et généré à l'instant t_1. Un tel paquet pourrait être envoyé à l'expiration de la fenêtre ATIM, si le noeud en question réussit à l'annoncer. Dans le cas où la génération du paquet survient à l'instant t_2, le paquet doit être différé pour l'intervalle beacon suivant afin d'être annoncé au préalable, dans le cas où la destination ordinaire n'a pas été avisée durant la fenêtre ATIM pour un autre transfert. Le même sort est infligé à tout paquet généré à l'instant t_3, indépendamment du statut de sa destination. Ces temps d'attente augmentés par la période ATIM peuvent pénaliser le trafic multimédia ou temps réel, en particulier dans un réseau à charge élevée.

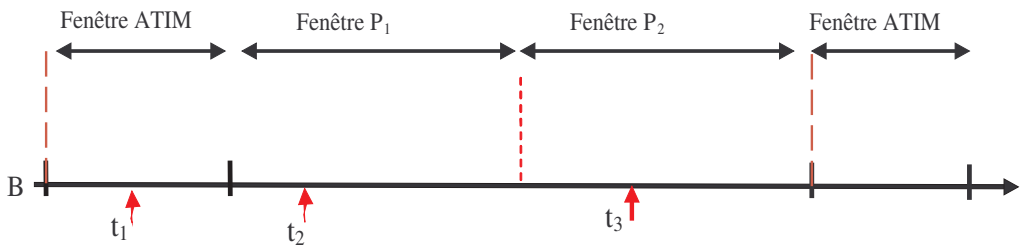


Figure 4.16 – Situations possibles pour la génération de paquets durant l'intervalle beacon

Le traitement diffère pour un paquet destiné à un voisin dominant, étant donné que ce dernier est continuellement actif. Pour cette raison, un paquet généré à l'instant t_1 va attendre uniquement l'expiration de la fenêtre ATIM pour être transmis et pourrait être écoulé durant la fenêtre P_1 en cours, si le noeud a pu accéder au canal. Par contre, le traitement accordé à un paquet généré à l'instant t_2 diffère selon l'état du noeud. Un noeud ordinaire actif ayant un tel paquet à l'instant t_2, va entrer en concurrence pour accéder au canal afin d'envoyer ce dernier. Alors qu'un noeud ordinaire déjà inactif va différer le traitement d'un tel paquet pour l'intervalle beacon suivant. Le même traitement sera accordé à un paquet arrivant à la file d'attente à l'instant t_3, durant la fenêtre dédiée aux échanges entre noeuds dominants.

On considère maintenant le cas d'un noeud dominant ayant un paquet à envoyé à noeud voisin ordinaire. Un paquet arrivant à la file d'attente à l'instants t_1 doit être annoncé durant la fenêtre ATIM afin de pouvoir être transmis durant la fenêtre P_1. L'avènement d'une telle situation à l'instant t_2 va prendre en considération le fait si le noeud récepteur a été déjà avisé durant la fenêtre ATIM ou pas. Si ce dernier a préalablement reçu une annonce ATIM de la part du noeud B durant la fenêtre ATIM, il va rester actif durant la fenêtre P_1 pour recevoir les paquets déjà annoncés lui étant destinés. Dans ce cas, le noeud B peut lui faire acheminer le paquet en question. Dans le cas contraire, le traitement de ce paquet sera différé pour l'intervalle beacon suivant. Le traitement est similaire pour tout paquet généré à l'instant t_3, qui doit être mis en instance jusqu'au début du prochain intervalle beacon.

Cependant, les paquets destinés à un noeud dominant voisin subissent moins d'attente aussi au niveau d'un noeud dominant, étant donné que de tels paquets n'ont pas besoin d'être annoncés au préalable. Des paquets arrivant à la file d'attente aux instants t_1 et t_2 devront être différés jusqu'à l'expiration de la fenêtre P_1 pour être traité. Tandis qu'un paquet arrivant à l'instant t_3 pourra être envoyé immédiatement, si le canal est libre après un délai aléatoire calculé conformément à l'algorithme de backoff. Ainsi, on constate que l'attente endurée par les paquets au niveau des files d'attente des noeuds, impliqués dans l'acheminement, contribue à augmenter le délai de séjour des paquets.

A travers cette phase d'évaluation de performances, on a démontré l'importance accordée au choix du mécanisme de clusterisation pour un support efficace des autres protocoles, à l'instar de l'approche de conservation d'énergie dans notre cas. Le mécanisme de clusterisation peut jouer un rôle important pour préserver les performances du système dans un environnement opérant en mode conservation d'énergie, dans la mesure qu'il est apte à atténuer les impacts négatifs suite à l'activation d'un tel mode pour réduire la consommation d'énergie. La majorité des approches, alliant le mode conservation d'énergie et une technique de clusterisation, se base sur l'élimination de l'activité réseau d'un sous ensemble de noeuds durant certaines périodes afin d'éviter le gaspillage de l'énergie. Toutefois, limiter l'échange des paquets à des périodes actives précises contribue à réduire le débit. On ne peut pas aussi négliger les délais d'attente

que subissent les paquets au niveau des files d'attente. Ainsi, la prise en compte de ces critères lors de la conception de l'approche de conservation d'énergie s'avère nécessaire.

4.3 Conclusion

Malgré les progrès technologiques perçus, l'énergie demeure encore une ressource critique, relevant encore des défis pour le déploiement à grande envergure des réseaux ad hoc et des réseaux de capteurs. Ce critère ayant un impact direct sur la durée de vie du noeud et du réseau par conséquence. L'importance accordée à ce paramètre a motivé le domaine de recherche. Dans cet axe, plusieurs approches ont été proposées afin de réduire les sources de gaspillage d'énergie, à l'instar de l'écoute passive et l'interception de paquets destinés à des noeuds voisins. Le mode conservation d'énergie, introduit dans le mécanisme `PSM` défini dans le standard 802.11, était le fondement de plusieurs travaux. Ce mode autorise les noeuds à devenir inactifs dès que possible. Il a été aussi exploité conjointement à des techniques de clusterisation afin de conserver de l'énergie tout en préservant la capacité du système. Ces propositions reposent sur le maintien d'un sous ensemble de noeuds actifs afin d'assurer les tâches de communication au profit des autres. Toutefois, le maintien de ces noeuds dominants est apte à réduire plus rapidement l'énergie disponible au niveau de ces noeuds. Durant les périodes de faible activité réseau, ces noeuds vont rester à l'écoute du médium inutilement. Par ailleurs, l'activation du mode conservation d'énergie peut pénaliser certains critères de performances importants pour certaines applications, à l'instar du délai de séjour. A la lumière de ces remarques, nous avons proposé une approche de conservation d'énergie basée sur l'existence d'une dorsale essayant de remédier à ces carences, qui fera l'objet du chapitre suivant.

Chapitre 5

Proposition de mécanismes pour la conservation d'énergie dans les réseaux Ad hoc

Le mode de conservation d'énergie vise à réduire la consommation d'énergie en autorisant les noeuds à entrer en mode inactif. De la sorte, on contribue aussi à étendre la durée de vie du réseau en ralentissant l'épuisement des ressources disponibles en terme d'énergie. Inclure des mécanismes de conservation d'énergie pour le déploiement réel des réseaux sans fil ad hoc ou de capteurs à grande envergure s'avère nécessaire. La prise en charge de tels dispositifs dès la phase de conception de protocoles destinés à ces environnements pourrait devenir une obligation. En effet, la majorité des applications cibles de telles infrastructures peuvent nécessiter un déploiement sur une longue durée dans des environnements hostiles, à l'instar des opérations militaires et de secours pouvant s'étendre sur plusieurs heures ou même plusieurs jours. Par ailleurs, malgré le progrès technologique, la production de batterie ayant une longévité plus grande s'avère insuffisante, étant donnée l'hétérogénéité native en terme de ressources perçue dans de tels réseaux.

Dans le chapitre précédent, on a présenté un recueil bref de certaines approches, ayant opté pour l'exploitation de la notion d'ensemble dominant connexe pour assurer la conservation d'énergie sans dégrader considérablement les performances du réseau. L'idée de base est de maintenir toujours un sous ensemble de noeuds actifs apte à assurer l'acheminement des données au profit des autres noeuds du réseau. Toutefois, l'impact du trafic additionnel dédié au processus de formation et de maintenance de l'ensemble dominant n'a pas été évalué et quantifié. D'un autre côté, garder les noeuds dominants actifs de manière continue contribue à épuiser leur énergie plus rapidement que les autres noeuds, sans négliger leur participation aux autres tâches additionnelles. L'épuisement de ressources pour certains noeuds pourrait entraîner le partitionnement du réseau.

Par ailleurs, malgré le recours à la notion d'ensemble dominant pour assurer l'achemine-

ment des données, on a négligé certaines exigences requises pour un certain type d'applications, telles que les applications multimédia et les applications temps réel. En effet, activer le mode conservation d'énergie impose l'annonce de tout paquet de données durant la fenêtre ATIM avant de pouvoir le transmettre, pour les variantes du mécanisme `PSM`. Grâce au paquet ATIM, le récepteur sera informé des paquets en attente qui lui sont destinés et reste en mode actif pour le reste de l'intervalle beacon. Tout paquet non annoncé sera différé pour l'intervalle beacon suivant, si la source n'a pas été notifiée au cours de la fenêtre ATIM. De telles situations sont fréquentes vu la limitation de la période d'annonce, durant laquelle tous les noeuds concernés par un échange essaient d'accéder au canal pour diffuser leurs annonces.

Les approches de conservation d'énergie, reposant sur une infrastructure virtuelle, essaient d'assurer un transfert continu des paquets de données en maintenant les noeuds dominants actifs. Ainsi, les paquets destinés à un noeud dominant ne requièrent aucune annonce au préalable. Uniquement les paquets destinés à un noeud ordinaire exigent une opération d'annonce réussie pour autoriser leur transmission durant la période d'activité relative à l'intervalle beacon courant. La transmission des paquets de données peut être différée pour l'intervalle beacon ultérieur, en particulier en présence de la concurrence d'un trafic de contrôle additionnel. Outre les messages relatifs au routage, les paquets de contrôle nécessaires à la formation et la maintenance de l'ensemble dominant connexe et les annonces associées peuvent avoir un impact négatif sur le trafic de données. Ce trafic additionnel va concourir avec les paquets de données pour accéder au canal. Cette concurrence peut ralentir ou bloquer le trafic de données, vu que la période active dédiée aux échanges effectifs de paquets est limitée. On rappelle que le trafic de contrôle croît avec le nombre de noeuds, contribuant à encombrer le canal. Ainsi, les paquets de données peuvent subir des délais d'attente considérables au niveau des files relatifs aux noeuds impliqués dans l'opération de transfert, en particulier dans un environnement à charge élevée. Le critère de délai de séjour peut être important ou insignifiant, ceci dépend essentiellement de l'application en cours d'exécution.

Dans ce contexte, on a proposé une approche de conservation d'énergie basée sur notre mécanisme de clusterisation `TBCA`. Ce travail essaie de profiter des apports du mode conservation d'énergie et des atouts de la clusterisation. L'idée fondatrice de cette approche est d'établir un classement des paquets selon leur priorité de traitement, la classification se base sur la consommation des ressources. Opérant de la sorte, on vise à garantir la satisfaction de certains critères, à l'instar du délai de séjour, débit, ..., importants pour certaines applications, dans un environnement opérant en mode conservation d'énergie. Par la suite, la présentation de notre approche de conservation d'énergie s'appuyant sur notre mécanisme de clusterisation `TBCA` fera l'objet de la section suivante.

5.1 Approche de prioritisation du trafic

Dans cette section, nous présentons une nouvelle approche de conservation d'énergie basée sur l'exploitation des apports de la clusterisation conjointement avec le mode veille, baptisée `CPPCM` 'Cluster based Prioritized Power Conservation mechanism'. Notre objectif principal est de réduire la consommation d'énergie tout en assurant l'acheminement des paquets de données sans endurer des temps d'attente importants au niveau des files d'attente des noeuds impliqués dans le transfert. A travers cette nouvelle approche, on vise à réduire la consommation globale d'énergie par paquet et à augmenter le nombre de paquets délivrés avec succès. Agissant de la sorte, on est apte à apporter une certaine garantie de qualité de service pour certains types de trafic, à l'instar des trafics multimédia et temps réel.

L'approche présentée se base essentiellement sur l'existence d'une infrastructure virtuelle formée par les clusterheads et gateways élus suite au déroulement de notre mécanisme `TBCA`. Le processus de clusterisation est déclenché périodiquement pour prendre en considération des informations actualisées lors de la reconstruction de l'ensemble dominant connexe. Le mécanisme permettant de ré élire les mêmes noeuds dominants vise à assurer une certaine stabilité à l'infrastructure virtuelle. A l'instar du mécanisme `SPAN+`, une période est réservée périodiquement pour finaliser le processus de formation de l'ensemble dominant connexe. Durant cette période, uniquement les messages dédiés au mécanisme `TBCA` sont échangés. Tout noeud est apte à déterminer l'instant de reprise de son traitement normal pour les annonces. Durant un intervalle beacon marqué par le relancment du processus de clusterisation, on étend la taille de la fenêtre ATIM pour annoncer convenablement les paquets en attente. Chaque noeud est apte à calculer cette extension conformément à la formule (4.1), cette dernière étant la même pour tous les noeuds afin de garder les mêmes repères. De cette manière, on évite l'envoi d'annonces additionnelles pour ces messages de contrôle. On contribue aussi à réduire le nombre de paquets de contrôle pouvant concurrencer les paquets de données ultérieurement lors de leur diffusion.

L'idée fondatrice de l'approche vise à maximiser le nombre de paquets délivrés en privilégiant le trafic ayant déjà consommé des ressources. Elle permet d'affecter des priorités différentes aux paquets de données selon la classification suivante qui différencie entre 3 types de paquets :

- La priorité 3 est attribuée au trafic de transit, ce dernier représente un trafic ayant déjà parcouru plusieurs noeuds et consommé par conséquence des ressources en terme d'énergie, de bande passante et de temps de traitement. Dans cette catégorie, on retrouve les paquets envoyés vers la destination finale (dernier saut) aussi bien par les coordinateurs que les noeuds ordinaires. Le deuxième cas de figure peut se présenter lorsqu'un noeud dominant perd son statut de CH ou GW pour devenir un noeud membre alors qu'il possède encore des paquets à délivrer vers un noeud voisin.
- La priorité 2 est octroyée au trafic échangé entre noeuds dominants, ce type de trafic

regroupe des paquets encore en cours d'acheminement, ayant consommé des ressources (essentiellement en terme d'énergie et de bande passante) et n'ayant pas encore atteint la machine cible.

- La priorité 1 est affectée au trafic externe, ce dernier correspond au trafic entrant n'ayant pas encore consommé des ressources. Dans ce type de trafic, on inclut les paquets envoyés par la machine source vers le coordinateur. Etant donné que l'acheminement du trafic est une tâche assignée aux noeuds de l'ensemble dominant, le prochain saut sera nécessairement un coordinateur ou la destination finale.

Le trafic ayant une priorité plus élevée sera privilégié par rapport aux autres types de trafic. Ainsi, on a intérêt à acheminer en priorité le trafic ayant déjà consommé des ressources (bande passante, énergie, ...) afin de minimiser la consommation d'énergie par paquet et les délais d'attente au niveau des files d'attente. Agissant de la sorte, on accorde aussi une certaine garantie de qualité de service pour certains types de trafic, même dans un environnement opérant en mode conservation d'énergie.

L'idée de notre approche a été inspirée par le mécanisme `SPAN` ayant introduit une fenêtre additionnelle pour différencier entre les paquets échangés entre coordinateurs et les paquets échangés entre coordinateurs et noeuds ordinaires. A l'instar de `SPAN`, on a introduit trois nouvelles fenêtres afin de pouvoir attribuer un traitement distingué à chaque type de trafic. Outre la fenêtre ATIM, chaque type de trafic aura une fenêtre qui lui est dédiée. Au niveau de chaque noeud, une file d'attente est attribuée à chaque type de trafic.

Dans ce qui suit nous allons exposer les deux variantes de cette approche. Dans la première variante, on va exposer le principe de base de l'approche de prioritisation dans laquelle on va opter pour l'utilisation de bornes fixes pour délimiter chaque fenêtre. Alors que la deuxième variante présente une optimisation permettant d'améliorer le débit tout en essayant de réduire le temps d'attente de paquets au niveau des files d'attente des noeuds impliqués dans l'acheminement. Dans cette deuxième variante, on a recours à des bornes dynamiques pour plus de flexibilité. Cette flexibilité se concrétise à travers la souplesse accordée lors de la détermination des bornes relatives aux différentes fenêtres de manière locale en fonction de la connaissance de l'état des files d'attente du noeud en question et de l'état du réseau dans le voisinage.

5.2 Variante 1 : bornes fixes

5.2.1 Principe

Notre principal objectif est d'essayer de minimiser les impacts des attentes subies dans les files d'attente aux niveaux des différents noeuds impliqués dans l'acheminement du trafic, lorsqu'un mécanisme de conservation d'énergie est déployé. En effet, même en présence d'une

charge faible, tous les paquets à émettre doivent être annoncés avant leur envoi. Cette annonce préalable des paquets permet de notifier le récepteur et de s'assurer que ce dernier sera actif au moment de la transmission effective du paquet de données. Dans un tel environnement, le recours à la notion d'ensemble dominant connexe vise à assurer l'acheminent des paquets de données grâce aux noeuds dominants toujours actifs. Les autres noeuds sont autorisés à entrer en mode de veille, si aucune réception ni transmission est prévue durant l'intervalle beacon courant.

Or, dans de telles approches, on pénalise les noeuds assurant le rôle de dominant, étant donné que ces derniers doivent toujours être actif, même s'ils ne sont pas impliqués dans une activité d'émission ou de réception. L'écoute passive du canal et l'interception des paquets destinés à un noeud voisin contribuent à épuiser plus rapidement l'énergie résiduelle du noeud dominant. Dans un tel contexte, cette consommation de l'énergie est considérée comme un gaspillage de l'énergie. Par ailleurs, tous les paquets, entrain d'être acheminés ou entrants dans le réseau, subissent un même traitement et possèdent des chances égalitaires pour accéder au médium. Un paquet ayant consommé des ressources, en terme d'énergie, de bande passante et de temps de calcul, doit être prioritaire par rapport aux paquets nouvellement générés par la source sans toutfois les étouffer. De cette manière, on peut réduire les temps d'attente subis par les paquets de transit et réduire par conséquence l'énergie consommée par paquet.

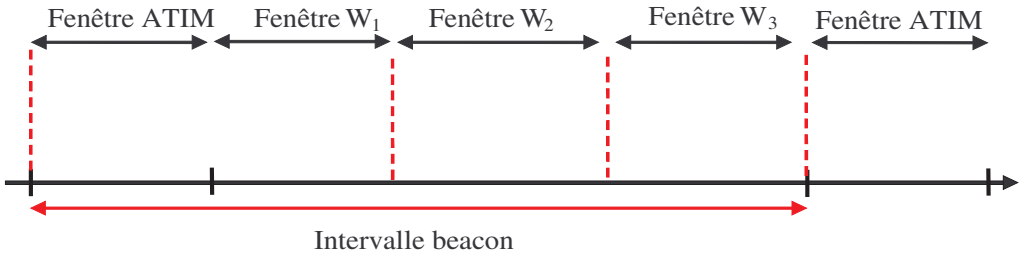


Figure 5.1 – Structure de l'intervalle beacon pour le mécanisme CPPCM

Dans notre approche, on a opéré de la même manière que le mécanisme `SPAN`. Toutefois, la structure de l'intervalle beacon adoptée par ce dernier a été modifiée afin de refléter les priorités accordées aux trois types de trafic. Pour pouvoir privilégier un type de trafic sur un autre, on a adopté la structure exposée dans la figure (5.1).

Conformément à son rôle défini dans le mécanisme `PSM`, la fenêtre ATIM est dédiée à l'annonce des paquets en attente dans les files d'attente des différents noeuds. On rappelle que l'ensemble des clusterheads et gateways constitue une infrastructure virtuelle chargée des tâches de routage et de l'acheminement des paquets de données.

Au début de chaque période de synchronisation, tous les noeuds ordinaires doivent être

actifs. En effet, tout noeud ordinaire doit être actif afin d'intercepter les annonces lui étant destinées provenant de ses voisins. Par ailleurs, durant cette période, tout noeud, ayant du trafic en attente destiné à un voisin ordinaire, aura la possibilité de notifier ce dernier à travers une annonce ATIM. A la réception d'un tel paquet, le noeud concerné sera apte de décider d'entrer en état inactif ou de rester actif. L'opération d'échange est considérée comme réussie, si l'annonce est correctement acquittée par le noeud destination. Toutefois, les annonces des paquets en multipoint ou en diffusion ne sont seront pas acquittées. On rappelle que les paquets à envoyer vers un voisin dominant n'ont pas besoin d'être annoncés, étant donnés que les noeuds dominants seront actifs durant les périodes dédiées pour l'acheminement de tels trafics. Pour cet effet, un noeud dominant, n'ayant aucun paquet à annoncer, peut entrer en mode conservation d'énergie durant la fenêtre ATIM. On réduit ainsi l'écoute passive pour les noeuds dominants.

Notre approche se base sur la distinction entre le trafic destiné vers la destination finale et le trafic nouvellement créé et voulant être injecté dans le réseau par la source. Ces deux types de trafic doivent être annoncés au préalable, si la destination est un noeud ordinaire. L'envoi effectif de tels paquets annoncés se déroule durant des fenêtres distinctes, en fonction du type du trafic. Afin de pouvoir établir une telle distinction, une file d'attente est réservée pour chaque type de trafic au niveau de chaque noeud. Une modification de la structure de l'annonce ATIM a été aussi nécessaire. Outre l'inclusion du statut du noeud, un champ *final_ destination* est ajouté pour informer le noeud récepteur s'il est la destination finale pour les paquets en attente. Ainsi, si ce bit est mis à 0, le noeud récepteur n'est qu'un noeud intermédiaire impliqué dans l'acheminement des paquets. Dans le cas contraire, le noeud est informé qu'il est la destination finale de ce paquet. Grâce à ses informations additionnelles, chaque noeud, ayant reçu une annonce ATIM, sera apte à déterminer durant quelle période il devrait rester actif. Les annonces relatives au trafic de transit doivent être prioritaires.

La première fenêtre, dite *Last Hop Window*, permet de faire écouler les paquets à envoyer vers leur destination finale. A l'expiration de la fenêtre ATIM, tous les noeuds dominants doivent être actifs. Tandis que les noeuds ordinaires doivent entrer en mode inactif, exception faite pour ceux ayant annoncé avec succès des paquets destinés à une destination finale ou ayant reçu une annonce ATIM avec le bit *final_ destination* marqué à 1.

La deuxième fenêtre, baptisée *Dominator window*, est dédiée uniquement à l'échange de paquets entre noeuds dominants, comme son nom l'indique. Pour cette raison, uniquement les noeuds dominants restent actifs pour assurer l'acheminement des paquets de données. Les noeuds ordinaires doivent retrouver leur état d'inactivité.

La dernière fenêtre, baptisée *Advertised window*, est associée au trafic nouvellement généré. A travers cette dernière fenêtre, on peut jouir d'un certain contrôle sur la charge à injecter dans le réseau. Le début de cette fenêtre exige le réveil des noeuds ordinaires ayant des paquets en instance à envoyer vers un noeud dominant ou ayant acquitté une annonce ATIM avec le

bit *final_ destination* marqué à 0. La réception d'une annonce ATIM en diffusion ou multipoint impose le réveil de tous les noeuds ordinaires durant cette fenêtre, du moment que tous les noeuds dominants doivent toujours rester actifs durant cette période pour accueillir les paquets entrants au réseau.

La distinction établie entre les différents types de paquets permet de limiter le nombre de noeuds actifs pendant une certaine fenêtre. Uniquement les noeuds concernés par ces échanges seront actifs, les autres noeuds, généralement ordinaires, peuvent retrouver leur état d'inactivité. Ainsi, on évite l'écoute passive du canal et l'interception des paquets destinés à des noeuds voisins.

5.2.2 Evaluation de performances

A l'instar des simulations conduites précédemment, le choix a été porté sur l'acheminement géographique pour le transfert des paquets de données entre les paires de source et destination. Anéantir le coût du routage nous permet de mieux évaluer les performances des approches de conservation d'énergie. La problématique du routage sera traitée ultérieurement. Durant cette phase d'évaluation, on a choisi différentes topologies de 200 noeuds uniformément distribués dans une aire de simulation de $500 \times 500m$. Les sources et destinations impliquées dans les échanges de données sont dispersées sur les frontières de l'aire de simulation afin de permettre une communication multi sauts.

Les scénarii de simulations visent à évaluer l'aptitude de `CPPCM` à préserver la capacité du réseau et à réduire les délais de bout en bout. Pour cet effet, on a comparé ses performances par rapport à celles de `SPAN` et `SPAN+`. La périodicité de mise à jour de la clusterisation est de 10 intervalles beacon. Les paramètres du mécanisme de clusterisation adoptés par `SPAN+` et `CPPCM` sont exposés dans la table suivante (5.1) :

Paramètres du réseau	
Portée	250m
Bande passante	2 Mbps
Durée de simulation	500s
Paramètres de `TBCA`	
Périodicité de MaJ de la clusterisation	10 intervalles beacon
(D_{1max}, D_{1min})	(31, 25)
(D_{1pmin}, D_{1npmin})	(15, 31)
$NbrMax\_gw\_CH$	6
$NombreMoy\_gw/CH$	5
Nombre de périodes de clusterisation successives en tant que dominant	5

Table 5.1 – Paramètres utilisés dans la simulation

Durant la première phase, on a fait varier la taille de la fenêtre W_1 afin de déterminer la taille appropriée permettant d'atteindre notre objectif, à savoir maximiser le débit tout en réduisant le délai moyen d'acheminement. La recherche de cette taille optimale a été effectuée pour des intervalles beacon de taille 100 ms et 300 ms. Les tailles des fenêtres ATIM associées sont respectivement 8 ms et 20 ms. Lors de ces simulations, on a choisi des tailles égalitaires des fenêtres W_2 et W_3. La charge injectée sur le réseau est de 10 flux de 20 pkt/sec chacun, transitant entre 20 sources et destinations.

Les figures suivantes (5.2, 5.3) montrent que la taille de la fenêtre W_1 a une grande influence sur les performances du système. Cette fenêtre revêt une importance particulière du moment qu'elle sert le trafic le plus prioritaire selon notre classification, à savoir les paquets en dernière phase d'acheminement. Toutefois, accorder une taille importante à cette fenêtre détériore les performances du système, en terme de débit et de délai d'acheminement. En effet, dans une telle situation, on va pénaliser le trafic de transit et le trafic nouvellement générés. Une taille réduite pour W_2 et W_3 n'autorise pas le traitement de tous les paquets en attente, le traitement des paquets restants est différé à chaque fois pour l'intervalle beacon suivant. Ce comportement fonctionnel justifie la réduction du nombre de paquets délivrés et l'augmentation du délai d'acheminement en agrandissant la taille de W_1.

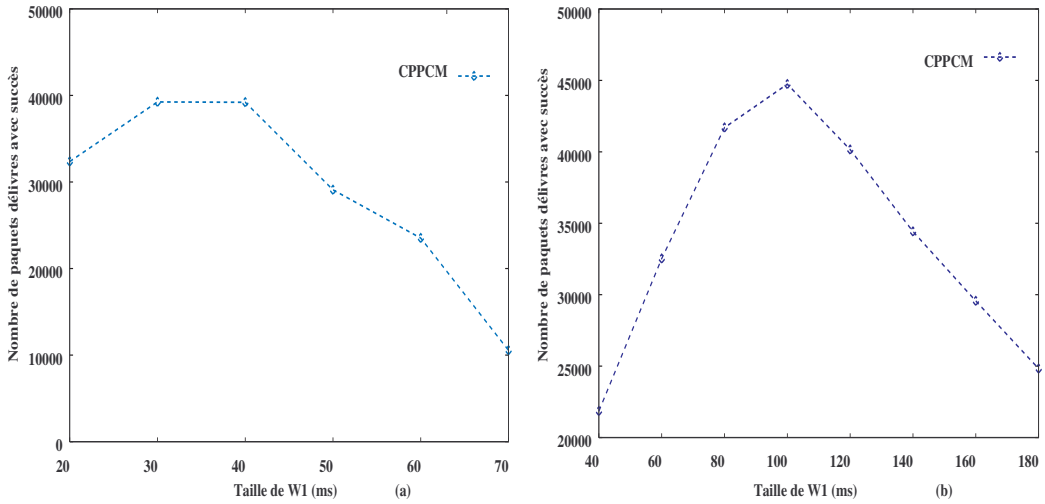


Figure 5.2 – (a)Nombre moyen de paquets délivrés avec succès en fonction de la la taille de la fenêtre W_1 pour un intervalle beacon 100 ms (b)Nombre moyen de paquets délivrés avec succès en fonction de la la taille de la fenêtre W_1 pour un intervalle beacon 300 ms

A travers les résultats présentés, on note que la taille de la fenêtre W_1 qui maximise le débit et réduit les délais d'acheminement pour un intervalle beacon de 100 ms est de 30 ms , alors que celle relative à un intervalle beacon de 300 ms est de 100 ms. Ces valeurs optimales seront adoptées durant le reste des éxpérimentations.

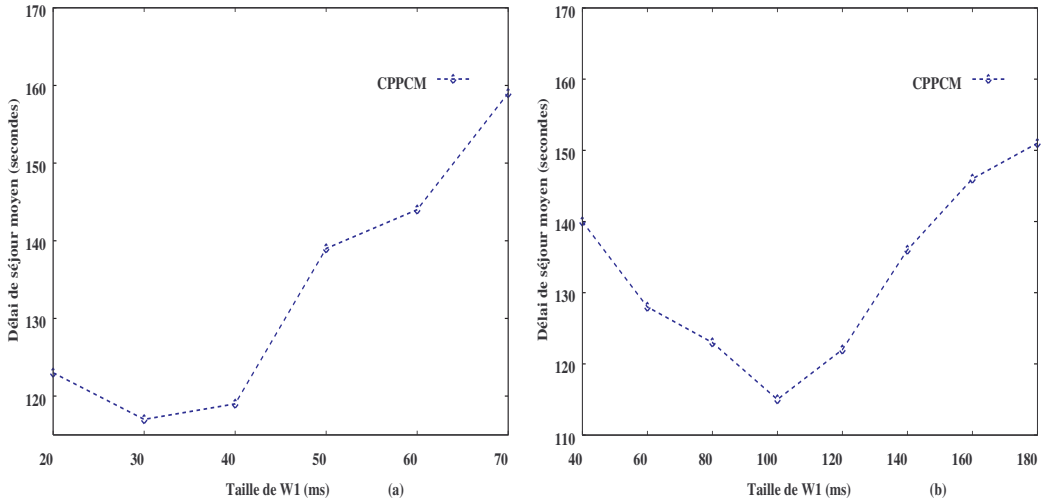


Figure 5.3 – (a)Délai moyen de séjour en fonction de la la taille de la fenêtre W_1 pour un intervalle beacon 100 ms (b)Délai moyen de séjour en fonction de la la taille de la fenêtre W_1 pour un intervalle beacon 300 ms

Dans une deuxième étape, notre objectif est d'évaluer l'impact de la variation des tailles des différentes fenêtres et du trafic sur les performances de l'approche proposée. Pour les valeurs par défaut de la fenêtre ATIM et de l'intervalle beacon, on a fixé les fenêtres P_1 et P_2, relatives aux mécanismes `SPAN` et SPAN+, à 46 ms chacune. Pour le mécanisme `CPPCM`, on a accordé 30 ms à la W_1, tandis qu'on a octroyé 31 ms pour chacune des deux fenêtres W_2 et W_3. Pour un intervalle beacon de 300 ms, la fenêtre ATIM correspondante a été fixée à 20 ms. Les fenêtres P_1 et P_2 associées aux mécanismes `SPAN` et `SPAN+` sont égales à 140 ms. Pour le mécanisme CPPCM, on a affecté 90 ms pour chacune des deux fenêtres W_2 et W_3, alors que la fenêtre W_1 s'est vue attribuée une période de 100 ms. Durant cette étape d'évaluation, on va adopter les critères de comparaison utilisés précédemment dans le chapitre 4.

Les mesures relatives au nombre de paquets délivrés avec succès confirment que le choix des tailles des différentes fenêtres revêt une grande importance, étant donné que ce choix peut influencer les performances des approches de conservation d'énergie proposées. Allouer plus de temps aux différentes fenêtres dédiées aux échanges des paquets est bénéfique pour les 3 approches, vu l'amélioration du débit constatée en comparent les résultats exposés dans les figures (5.4.a) et (5.4.b). Toutefois, l'approche `CPPCM` surpasse `SPAN` et `SPAN+` dans les deux cas. La différence est plus remarquable lorsqu'on opte pour un intervalle beacon de 300ms, grâce à l'extension des différentes fenêtres ayant permis d'annoncer et d'écouler plus de paquets. D'un autre côté, le mécanisme de clusterisation `TBCA` permet d'alléger le réseau des paquets de contrôle associés au processus de clusterisation, laissant ainsi plus de ressources au profit des paquets de données. Les deux approches `SPAN+` et `CPPCM` reposent sur ce mécanisme de clusterisation, ce qui justi-

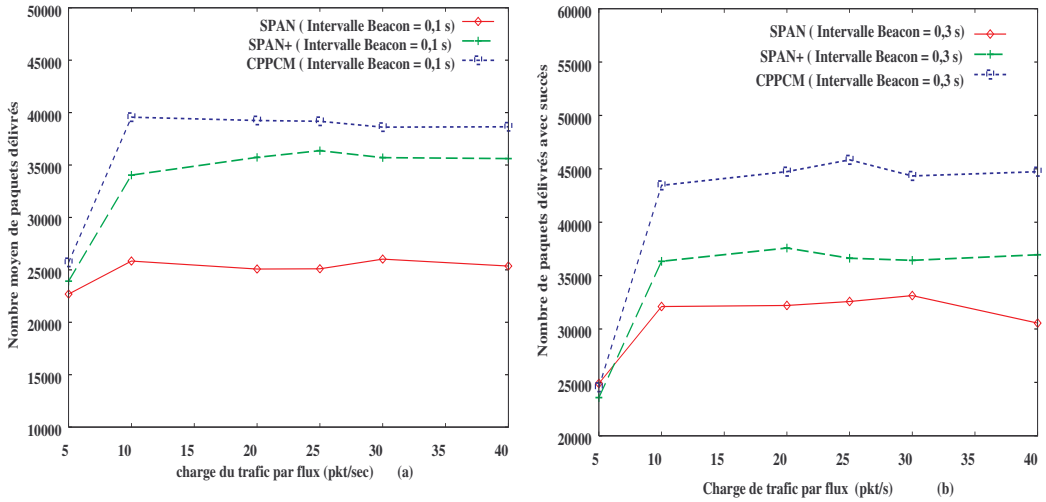


Figure 5.4 – Nombre moyen de paquets délivrés avec succès en fonction de la charge du trafic par flux tout en variant la taille des différentes fenêtres

fie l'accroissement du débit noté par rapport à `SPAN`. Cependant, le recours au mécanisme de clusterisation `TBCA` n'est pas le seul élément ayant avantagé `CPPCM`, la classification des paquets adoptée permet de privilégier les paquets en cours d'acheminement par rapport à ceux nouvellement générés, augmentant ainsi le taux de délivrance de ce dernier. La distinction entre ces 3 types de trafic permet de réduire le nombre de noeuds candidats potentiels en compétition pour accéder au canal durant les 3 fenêtres distinctes.

Ainsi, en octroyant plus de priorité aux paquets ayant déjà consommé des ressources, on permet de les acheminer plus rapidement à travers le réseau. Les paquets, ayant atteint leur dernier saut, sont prioritaires par rapport à ceux encore en cours d'acheminement. Agissant de la sorte, on minimise les délais d'attente subis au niveau des différents noeuds impliqués dans l'opération de transfert. Pour cette raison, le délai de séjour moyen constaté pour `CPPCM` est plus réduit que ceux de `SPAN` et `SPAN+` pour un intervalle beacon de 300 ms. Pour ces deux derniers, tous les paquets sont traités de manière égalitaire. Aucun traitement particulier n'est adopté lors du traitement des paquets de données. Les résultats sont moins bons pour un intervalle beacon de 100 ms, vu que la période active a été segmentée en trois fenêtres distinctes. Ainsi, la durée accordée à chaque fenêtre sera insuffisante pour écouler les paquets en attente en particulier lorsque la charge croît.

A travers la figure (5.6), on expose les mesures relatives à l'énergie résiduelle moyenne pour les trois approches de conservation d'énergie. L'amélioration du débit notée pour le mécanisme `CPPCM` justifie la légère consommation additionnelle en terme d'énergie par rapport à `SPAN+`. Malgré cela, le mécanisme `SPAN` demeure le mécanisme le plus gourmand en terme d'énergie, vu qu'il repose sur un échange périodique de messages de contrôles supplémentaires. Pour cette

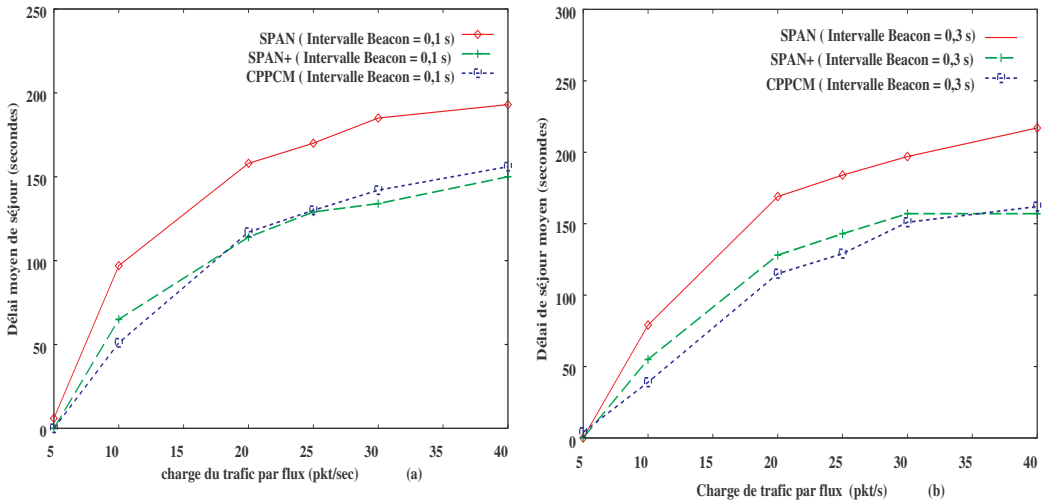


Figure 5.5 – Délai de séjour moyen en fonction de la charge du trafic par flux tout en variant la taille des différentes fenêtres

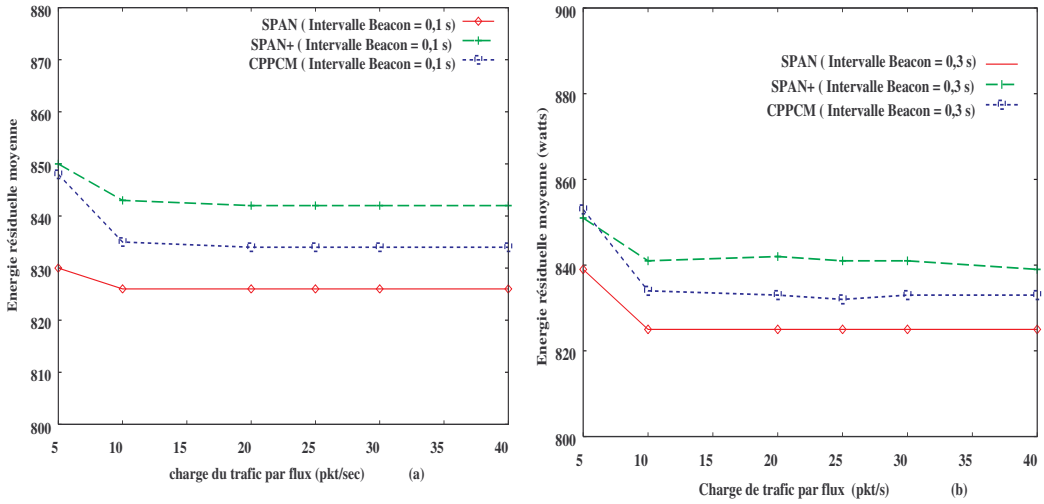


Figure 5.6 – Energie résiduelle moyenne en fonction de la charge du trafic par flux tout en variant la taille des différentes fenêtres

raison, il détient le coût du paquet en énergie consommée le plus élevé. L'aptitude de CPPCM à écouler plus de trafic, en privilégiant les paquets ayant consommé plus de ressources, lui permet de réaliser le coût le plus bas parmi les 3 protocoles comparés. Ce coût s'améliore pour les trois approches en augmentant la taille des différentes fenêtres, comme on le remarque à partir des figures (5.7.a) et (5.7.b).

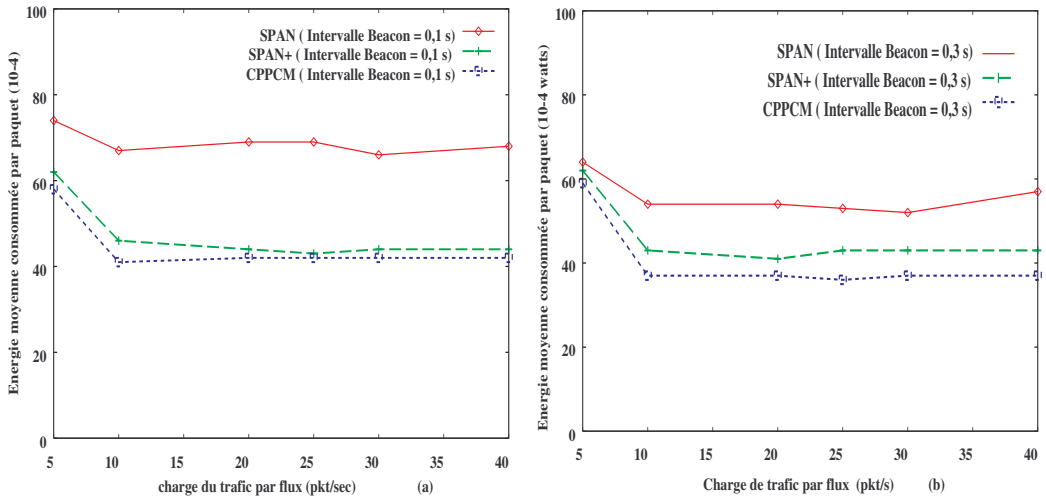


Figure 5.7 – Energie consommée par paquet en fonction de la charge du trafic par flux tout en variant la taille des différentes fenêtres

5.3 Variante 2 : bornes dynamiques

5.3.1 Principe

La première variante de notre approche de conservation d'énergie permet d'améliorer certaines caractéristiques à l'instar du débit et du délai de séjour lors du déploiement du mode veille. L'exploitation de la classification proposée en se basant sur la consommation de ressources nous a permis d'accorder un certain privilège aux paquets déjà injectés sur le réseau afin de réduire les délais d'attente au niveau des files d'attente des noeuds intermédiaires et l'énergie consommée par conséquence. Toutefois, le recours à des bornes rigides fixées au préalable peut contribuer à sous exploiter les ressources disponibles. En effet, en fonction de l'état des files d'attente relatives aux différentes catégories de paquets, tout noeud doit être apte à modifier la taille de ses différentes fenêtres pour s'adapter à ses besoins. Ainsi, un noeud, ayant beaucoup de paquets à envoyer vers leur destination finale, a intérêt à étendre la taille de la fenêtre W_1 pour écouler les paquets en attente. Par contre, un noeud dominant, ayant uniquement des paquets à acheminer vers ses confrères dominants, doit raccourcir la fenêtre W_1 au profit de la fenêtre W_2, afin de disposer de plus de temps pour traiter tous les paquets. De cette manière, un tel

noeud dominant peut profiter du canal libre pour acheminer les paquets en attente au niveau de sa file d'attente, du moment qu'aucun autre voisin n'ait l'intention de délivrer des paquets vers une destination finale. Grâce à l'écoute des annonces des paquets en attente et l'état de ses propres files d'attente, tout noeud essaie d'estimer le degré d'occupation des différentes fenêtres. Toutefois, on rappelle qu'aucune annonce n'est requise pour les paquets destinés à un noeud dominant, qu'il soit destination finale ou un noeud intermédiaire sur le chemin emprunté. Ainsi, l'estimation de la charge relative aux fenêtres W_1 et W_3 ne peut qu'être approximative.

Pour cet effet, la version, baptisée `DCPPCM` 'Dynamic Cluster based Prioritized Power Conservation mechanism', autorise les noeuds dominants et certains noeuds ordinaires à ajuster de manière dynamique les bornes des différentes fenêtres afin de profiter des instants de repos durant ces dernières et allouer plus de temps si nécessaire aux paquets les plus prioritaires. Le pouvoir d'ajuster de manière locale et dynamique les bornes des différentes fenêtres a été accordé aux noeuds dominants. En effet, les tâches additionnelles sont attribuées à l'ensemble des noeuds dominants, à l'instar de la recherche des routes et le transfert des paquets de données. Par conséquence, la majorité du trafic va être acheminé grâce au service de cette infrastructure virtuelle fournie par le mécanisme de clusterisation. L'aptitude à contrôler les tailles des trois fenêtres définies leur autorise une meilleure exploitation des ressources disponibles. Pour certains cas précis, les noeuds ordinaires peuvent profiter de cette flexibilité. Les bornes bénéficiant de cette souplesse sont celles marquant la fin de la fenêtre W_1 et la fin de la période W_2. Tout au long de cette section, on va les référencer respectivement par les variables b_1 et b_2.

Au début de chaque intervalle beacon, tous les noeuds doivent être actifs. Durant la fenêtre ATIM, les noeuds membres sont dans l'obligation de rester actifs afin d'intercepter les annonces qui leur sont destinées. Ainsi, ils peuvent déterminer durant quelles périodes ils doivent être actifs pour recevoir les paquets annoncés avec succès. Les noeuds dominants doivent aussi rester actifs durant toute la fenêtre ATIM, même s'ils ne possèdent pas de paquets à annoncer. Ces derniers doivent rester à l'écoute du canal pour intercepter les annonces échangées dans le voisinage. Ainsi, tout noeud dominant peut estimer la charge du trafic à prévoir durant les fenêtres W_1 et W_3. Pour faciliter cette tâche, tout noeud doit indiquer la quantité du trafic (Qte_trafic) en attente vers le récepteur de l'annonce ATIM, tout en spécifiant le nombre de paquets (Nbr_pkt).

Grâce à l'écoute des annonces ATIM, les noeuds dominants calculent les barres b_1 et b_2. En comptabilisant les annonces ATIM avec le bit *final_ destination* à 1, tout noeud dominant aura une idée sur l'occupation du canal durant la fenêtre W_1 relative aux envois des paquets destinés à un récepteur final. Le reste des annonces reflète la charge approximative à prévoir durant la fenêtre W_3. Ainsi, tout noeud dominant alloue deux variables *Qte_ traficW1* et *Qte_ traficW3*, pour estimer la quantité de trafic à échanger durant les fenêtres W_1 et W_3 respectivement, et deux variables *pkt_ W1* et *pkt_ W3* pour approximer le nombre de paquets à transmettre durant ces deux fenêtres aussi. Ces différentes variables seront incrémentées à l'interception de toute

annonce ATIM dans le voisinage. Les variables *Qte_ traficW1* et *pkt_ W1* seront incrémentées à l'interception de toute annonce ATIM avec le bit *final_ destination* à 1, alors que les deux autres ne sont actualisées que lors de l'interception des annonces relatives au trafic à échanger durant la fenêtre W_3. Les paquets en diffusion sont aussi annoncés afin de pouvoir être transmis durant la fenêtre W_3. Pour cette raison, tout noeud doit aussi maintenir une variable *pktDif_ W3* à actualiser conjointement à la variable *Qte_ traficW3*, lors de la réception de toute annonce ATIM pour des paquets en diffusion.

De leur côté, les noeuds ordinaires, concernés par un échange de données durant W_1, doivent approximer la durée de leur activité en se basant sur les annonces envoyées ou reçues avec le bit *final_ destination* à 1. A l'instar des noeuds dominants, tout noeud ordinaire actualise ses variables *pkt_ W1* et *Qte_ traficT1* à la réception de toute annonce ATIM lui étant destinée et ayant le bit *final_ destination* à 1. Le cas de noeuds ordinaires ayant des paquets en attente vers des destinations finales se présente lorsqu'un noeud dominant perd son rôle pour devenir ordinaire et qu'il possède encore des paquets à délivrer à une destination finale.

Ainsi, à travers la collecte de certaines informations à partir des annonces échangées dans le voisinage et l'état des files d'attente, chaque noeud aura une vue locale approximative sur l'occupation du canal durant les fenêtres W_1 et W_3. Ainsi, l'estimation des tailles des différentes fenêtres sera entièrement personnalisée pour tout noeud afin de s'adapter à ses propres besoins.

A l'expiration de la fenêtre ATIM, un noeud dominant doit aussi prendre en considération ses paquets annoncés avec succès à envoyer durant W_1, lors de son estimation de la barre b_1. Pour cela, il doit compter le nombre de paquets destinés à un récepteur final annoncés avec succès et les ajouter à la variable *pkt_ W1*. Il doit aussi comptabiliser la quantité de trafic associée à ses propres paquets et les ajouter aussi à la variable *Qte_ traficW1*. L'estimation de la taille de la fenêtre W_1 sera conformément à la formule suivante :

$$\begin{aligned} occupation \;=\; & (pkt\_W1 * (DIFS + CWsize + RTS\_Time + CTS\_Time + ACK\_Time \\ & + (3 * SIFS))) + Tx(Qte\_traficW1) \end{aligned} \tag{5.1}$$

Avec :

Tx(Qte_traficW1) : Temps nécessaire pour transmettre *Qte_ traficW1*

RTS_Time : Temps nécessaire pour transmettre un RTS

CTS_Time : Temps nécessaire pour transmettre un CTS

ACK_Time : Temps nécessaire pour transmettre un ACK

Ainsi, un noeud dominant doit estimer le temps nécessaire à l'envoi de *pkt_ W1* paquets de données avec l'échange préalable RTS/CTS, en considérant un délai aléatoire maximal pour tout paquet avec la barre minimale de la fenêtre de contention. Toutefois, afin de ne pas étrangler le

trafic entrant, l'extension de la fenêtre W_1 est bornée par le seuil maximal égal à *Taille* (W_1+W_2). Si la valeur de *occupation* dépasse cette valeur, la taille de W_1 sera fixée à ce seuil. La fenêtre W_1 peut être raccourcie au profit de la fenêtre W_2, dans le cas où aucun paquet ne sera adressé à un noeud ordinaire dans le voisinage. Une telle situation n'aura aucun impact négatif sur les échanges entre le noeud dominant concerné par la modification et ses confrères. Ce dernier sera toujours actif et peut aisément recevoir les paquets lui étant destiné en tant que noeud intermédiaire ou en tant que destination finale.

Algorithme 1 calcul de la taille de la fenêtre W_1 par un noeud dominant

```
1: occupation ⟵ 0
2: //calcul de occupation
3: occupation ⟵ (pkt_W1*(DIFS+CWsize+RTS_Time+CTS_Time+ACK_Time+
   (3*SIFS))) + Tx(Qte_traficW1)
4: if (occupation > W1_Window) then
5:    seuil ⟵ (W1_Window+W2_Window) // calcul du seuil maximal
6:    if (occupation <= seuil) then
7:       Taille(W1) ⟵ occupation
8:    else
9:       Taille(W1) ⟵ seuil
10:   end if
11: else
12:   if (occupation == 0) then
13:      Taille(W1) ⟵ 0
14:      Etendre_la_fenetre_W2
15:   else
16:      Taille(W1) ⟵ occupation
17:      Etendre_la_fenetre_W2
18:   end if
19: end if
```

A l'expiration de la fenêtre ATIM, les noeuds, n'ayant reçu ou émis aucune annonce ATIM avec le bit *final_ destination* à 1, peuvent entrer en mode conservation d'énergie. Dans le cas contraire, tout noeud ordinaire reprend les mêmes actions entamées par un noeud dominant. Il doit prendre en considération ses propres paquets de données annoncées avec succès à écouler durant la fenêtre W_1. Une fois cette mise à jour des variables *pkt_ W1* et *Qte_ traficW1* achevée, il estime la taille de la fenêtre W_1 conformément à la formule (4.1). Pour un noeud ordinaire, la fenêtre W_1 gardera sa taille initiale si l'estimation de la charge est inférieure à cette dernière. Cette précaution a été prise comme mesure de sécurité dans le cas où la transmission de tous les paquets nécessite plus que le temps estimé. En effet, chaque noeud ne peut avoir qu'une vue locale approximative sur l'occupation du canal durant la fenêtre W_1, étant donnés que les paquets destinés à un noeud dominant ne sont pas annoncés au préalable.

Afin de permettre aux noeuds ordinaires actifs durant W_1 de retrouver leur état d'inactivité

Algorithme 2 calcul de la taille de la fenêtre W_1 par un noeud ordinaire

```
1: occupation ⟵ 0
2: //calcul de occupation
3: occupation ⟵ (pkt_W1*(DIFS+CWsize+RTS_Time+CTS_Time+ACK_Time+
   (3*SIFS))) + Tx(Qte_traficW1)
4: if (occupation > W1_Window) then
5:    seuil ⟵ (W1_Window+W2_Window) // calcul du seuil maximal
6:    if (occupation <= seuil) then
7:       Taille(W1) ⟵ occupation
8:    else
9:       Taille(W1) ⟵ seuil
10:   end if
11: end if
```

dès que possible, tout émetteur doit signaler l'achèvement du transfert de données avec une destination donnée. L'ajout d'un bit, baptisé *last_frame*, permet à toute source d'indiquer si elle a d'autres paquets annoncés avec succès et destinés au même récepteur. L'activation de ce bit dans le dernier paquet de données informe le récepteur de la finalisation de l'étape de transfert des paquets de données avec cet émetteur. Ainsi, tout noeud ordinaire peut entrer en mode conservation d'énergie, s'il n'est pas impliqué dans d'autres opérations de transfert.

A l'expiration de la fenêtre W_1, tous les noeuds ordinaires retrouvent leur état d'inactivité. De leur côté, les noeuds dominants doivent estimer s'il est nécessaire d'étendre la taille de leur fenêtre W_2 en se basant sur l'état de la file d'attente relative aux paquets à échanger avec les noeuds dominants. Toutefois, tout noeud dominant doit au préalable calculer l'extension maximale autorisée. En effet, on va emprunter du temps de la fenêtre W_3. Occuper constamment le canal avec le trafic entre coordinateurs peut empêcher le traitement des paquets à acheminer normalement durant la fenêtre W_3. Une telle situation est apte à augmenter considérablement l'attente des paquets nouvellement générés et par conséquence le délai de séjour.

Pour cette raison, tout noeud dominant doit estimer le temps nécessaire pour écouler le trafic entrant dans son voisinage immédiat. L'objectif est d'éviter la saturation des files d'attente relatives au trafic entre noeuds ordinaires et/ou noeuds dominants et d'égorger le trafic entrant. Grâce aux annonces ATIM avec le bit *final_ destination* à 0, tout noeud dominant peut estimer la quantité du trafic à écouler par des noeuds ordinaires ou dominants vers des noeuds ordinaires. Lors de l'estimation de l'occupation de W_3, tout noeud dominant doit aussi comptabiliser ses propres paquets annoncés avec succès (unicast et en diffusion) à travers la mise à jour des variables *pkt_ W3*, *pktDif_ W3* et *Qte_ traficW3*.

On rappelle que la fenêtre W_3 est destinée pour le trafic nouvellement généré. Etant donné que l'acheminement des paquets de données s'effectue à travers la dorsale, la majorité de ce trafic sera destinée à des noeuds dominants. Vu qu'un tel trafic n'est pas annoncé durant la fenêtre

ATIM, tout noeud dominant essaie de l'approximer en se basant sur un historique des transmissions réussies vers des noeuds dominants durant la fenêtre W_3 antérieure. Ainsi, tout noeud dominant réserve une variable *Nbr_sccuess* qui sera incrémentée lors de l'interception de tout paquet de données destiné à un noeud dominant, le noeud lui-même ou un voisin dominant. Cette approximation sera comptabilisée lors du calcul de l'occupation de la fenêtre W_3. L'estimation de la charge de la fenêtre W_3 sera conformément à la formule suivante où on comptabilise le trafic annoncé avec succès durant la fenêtre ATIM (unicast et en diffusion) et le trafic approximatif destiné vers les noeuds dominants :

$$\begin{aligned} estimation\_traficUnicast &= (pkt\_W3 * (DIFS + CWsize + RTS\_Time + CTS\_Time \\ &\quad + ACK\_Time + (3 * SIFS))) + Tx(Qte\_traficW3) \quad (5.2) \\ estimation\_traficDiffusion &= (pktDif\_W3 * (DIFS + CWsize)) \quad (5.3) \\ estimation\_traficDominant &= (Nbr\_sccuess * (DIFS + CWsize + RTS\_Time \\ &\quad + CTS\_Time + ACK\_Time + (3 * SIFS) \\ &\quad + Tx(TailleMax\_pkt))) \quad (5.4) \\ occupation3 &= estimation\_traficUnicast + estimation\_traficDiffusion \\ &\quad + estimation\_traficDominant \quad (5.5) \end{aligned}$$

On note que la variable *Qte_traficW3* représente la somme des paquets unicast et en diffusion annoncés avec succès et associés à la fenêtre W_3. Pour approximer le trafic destiné aux noeuds dominants, on considère la taille maximale d'un paquet. Ainsi, tout noeud dominant sera apte à calculer combien de temps il est autorisé à emprunter de W_3 grâce à la formule suivante :

$$Extension2 = W3\_Window - occupation3 \quad (5.6)$$

Afin de déterminer si la fenêtre W_2 doit être étendue, tout noeud dominant doit estimer l'occupation de cette dernière en comptabilisant ses propres les paquets en attente destinés à des voisins dominants. Cette estimation est effectuée conformément à la formule (4.9) :

$$\begin{aligned} occupation2 &= (pkt\_W2 * (DIFS + CWsize + RTS\_Time + CTS\_Time + ACK\_Time \\ &\quad + (3 * SIFS))) + Tx(Qte\_traficW2) \quad (5.7) \end{aligned}$$

Avec :

pkt_W2 : Variable pour comptabiliser le nombre de paquets en attente

Qte_traficW2 : Variable pour estimer la quantité de trafic associée à la fenêtre W_2

Lors du calcul personnalisé de la taille de la fenêtre W_2, tout noeud dominant doit tenir en compte l'éventuelle extension de la fenêtre W_1.

Algorithme 3 calcul de la taille de la fenêtre W_2 par un noeud dominant

1: // calcul de occupation relative à W3
2: $occupation3 \longleftarrow (pkt\_W3*(DIFS+CWsize+RTS\_Time+CTS\_Time+ACK\_Time+(3*SIFS)))+(pktDif\_W3*(DIFS+CWsize))+Tx(Qte\_traficW3)+(Nbr\_sccuess*(DIFS+CWsize+RTS\_Time+CTS\_Time+ACK\_Time+(3*SIFS)+Tx(TailleMax\_pkt)))$
3: //Calcul de l'extension relative à la fenêtre W2
4: **if** $(occupation < W3\_Window)$ **then**
5: $Extension2 \longleftarrow W3\_Window - occupation3$
6: **else**
7: Extension2 $\longleftarrow 0$
8: **end if**
9: //Calcul du seuil pour la fenêtre W2
10: $seuil \longleftarrow W2\_Window + extension2$
11: //Calcul de occupation relative à W2
12: $occupation2 \longleftarrow (pkt\_W2*(DIFS+CWsize+RTS\_Time+CTS\_Time+ACK\_Time+(3*SIFS)))+Tx(Qte\_traficW2)$
13: // Pas d'extension pour W1
14: **if** $(W1\_Window == taille\_par\_defaut)$ **then**
15: **if** $((occupation2 <= seuil)And(occupation2 > W2\_Window))$ **then**
16: $Taille(W2) \longleftarrow occupation$
17: $Taille(W3) \longleftarrow W3\_Window - (occupation2 - W2\_Window)$
18: **else**
19: **if** $(occupation2 > seuil)$ **then**
20: $Taille(W2) \longleftarrow seuil$
21: $Taille(W3) \longleftarrow W3\_Window - extension2$
22: **end if**
23: **end if**
24: **else**
25: // la fenêtre W1 a été étendue
26: // Calcul de l'extension de la fenêtre W1
27: $extension1 \longleftarrow W1\_Window - taille\_par\_defaut$
28: **if** $(((occupation2 + extension1) <= seuil)And((occupation2 + extension1) > W2\_Window))$ **then**
29: $Taille(W2) \longleftarrow occupation$
30: $Taille(W3) \longleftarrow W3\_Window - ((occupation2 + extension1) - W2\_Window)$
31: **else**
32: **if** $((occupation2 + Extension1) > seuil)$ **then**
33: $Taille(W2) \longleftarrow seuil$
34: $Taille(W3) \longleftarrow W3_Window - extension2$
35: **else**
36: $Taille(W2) \longleftarrow taille\_par\_defaut - extension1$
37: **end if**
38: **end if**
39: **end if**

5.3.2 Evaluation de performances

Dans cette évaluation de performance, on a repris le scénario relatif à la variation de la charge par flux pour mettre en évidence l'utilité des barres flexibles. Les résultats exposés sont relatifs au cas d'un intervalle beacon de $300ms$, étant donnée que cette augmentation de la taille a été bénéfique pour les différentes approches de conservation d'énergie. Dix flux parcourent le réseau formé par 200 noeuds éparpillés dans une aire de simulation de $500 \times 500m$. La table suivante résume les paramètres adoptés durant cette phase de simulation (5.2) :

Paramètres du réseau	
Portée	250m
Bande passante	2 Mbps
Durée de simulation	500s
Paramètres de TBCA	
Périodicité de MaJ de la clusterisation	10 intervalles beacon
(D_{1max}, D_{1min})	(31, 25)
(D_{1pmin}, D_{1npmin})	(15, 31)
$NbrMax\_gw\_CH$	5
$NombreMoy\_gw/CH$	4
Nombre de périodes de clusterisation successives en tant que dominant	5
Paramètres des mécanismes de conservation d'énergie	
Intervalle beacon	300ms
Fenêtre ATIM	20 ms
(W_1, W_2, W_3) pour CPPCM et DCPPCM	(100, 90, 90)
(P_1, P_2) pour SPAN et SPAN+	(140, 140)

Table 5.2 – Paramètres utilisés dans la simulation

On assume que l'énergie initiale disponible au niveau de tout noeud est de 1000 watts. Les taux de consommation d'énergie adoptés sont conformes aux valeurs spécifiées dans la table suivante (5.3) :

Transmission	Réception	Ecoute passive	Inactif
0.660	0.395	0.296	0

Table 5.3 – Taux de consommation de l'énergie (watts)

La première figure expose le nombre de paquets délivrés avec succès durant toute la durée de la simulation vers toutes les destinations. A travers la figure (5.8), on remarque une légère amélioration du débit avec le mécanisme DCPPCM, en particulier pour les charges faibles et moyennes. En effet, dans de tels cas, DCPPCM profite de la flexibilité qui lui a été accordée en exploitant les périodes d'inactivité qui s'offrent aux noeuds dominants. Dans un environnement à charge élevé,

les différentes fenêtres seront trop encombrées, ainsi, les extensions possibles seront trop réduites ce qui explique le débit mesuré.

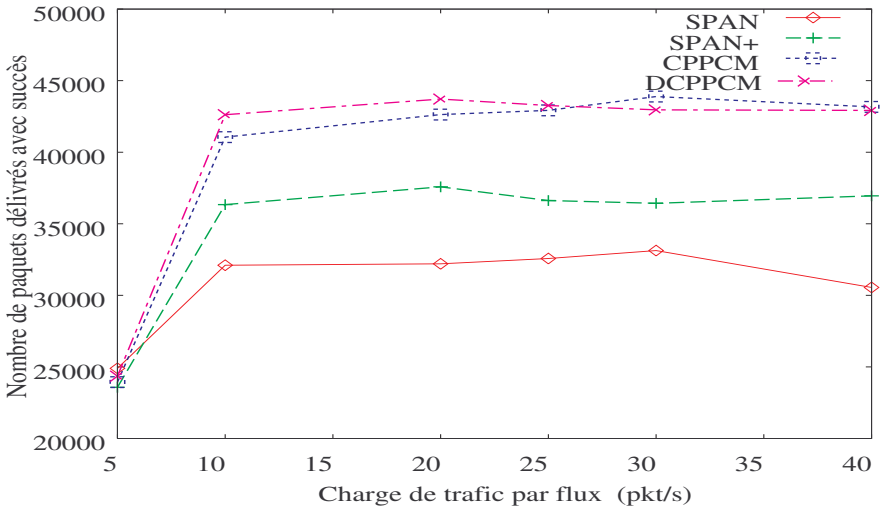


Figure 5.8 – Nombre moyen de paquets délivrés avec succès en fonction de la charge du trafic par flux

Cette souplesse a permis aussi à `DCPPCM` d'octroyer plus de privilège aux paquets les plus prioritaires, en se basant sur les ressources consommées comme critère de classification. Ainsi, les paquets destinés vers un récepteur final, annoncés avec succès, sont écoulés en premier. Les noeuds émetteurs, indépendamment de leur statut, essaient d'étendre la taille de la fenêtre W_1 afin de pouvoir traiter tous les paquets en attente. Les paquets entre noeuds dominants détiennent la deuxième position. Ainsi, tout noeud dominant essaie de prolonger la fenêtre W_2, sans toutefois pénaliser le trafic entrant. Grâce à ce fonctionnement, on réduit les attentes au niveau des files d'attente des différents noeuds. Les mesures relatives au délai de séjour moyen confirment l'aptitude de `DCPPCM` à s'adapter aux besoins de chaque noeud pour améliorer les performances globales du système. Le délai de séjour de `DCPPCM` est inférieur par rapport aux autres délais, comme l'illustre la figure (5.9).

Cette version dynamique a montré aussi sa capacité à réduire l'énergie consommée globale et par paquet, étant donnés que les paquets ayant consommé plus de ressources ont été plus favorisés lors de l'opération d'acheminement. Le gain en énergie résiduelle de `DCPPCM`, lisible sur la figure (5.10), et l'amélioration en débit perçue expliquent la réduction du coût par paquet en terme d'énergie consommée.

Le gain apporté par la version dynamique lors de ces expérimentations est réduit. Lors de cette première phase, le nombre moyen de sauts entre une paire de noeuds communicants est de l'ordre de 3. Dans une telle situation, un noeud dominant sera généralement à proximité d'un noeud source ou un noeud destination. Ainsi, les extensions éventuelles seront très limitées,

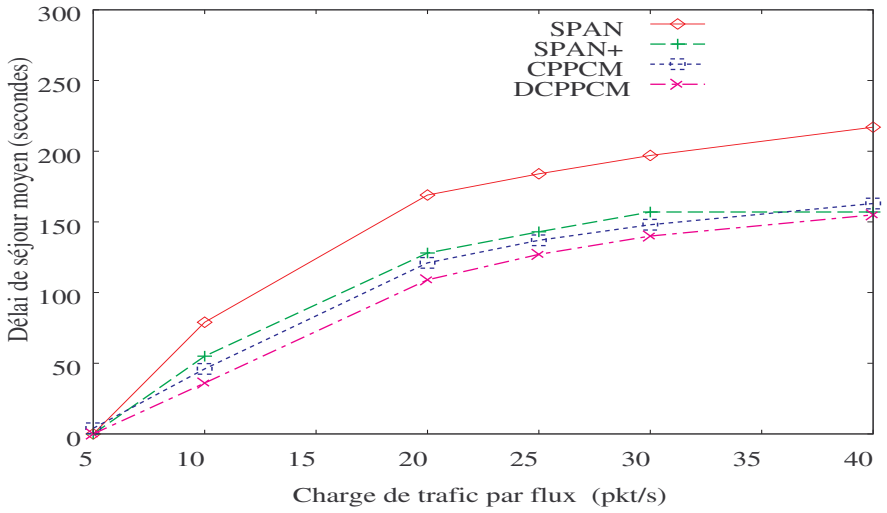


Figure 5.9 – Délai de séjour moyen en fonction de la charge du trafic par flux

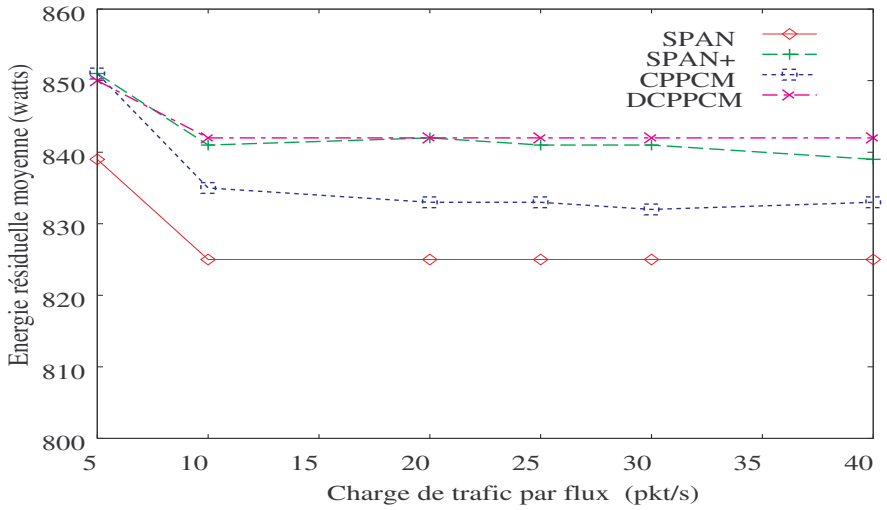


Figure 5.10 – Energie résiduelle moyenne en fonction de la charge du trafic par flux

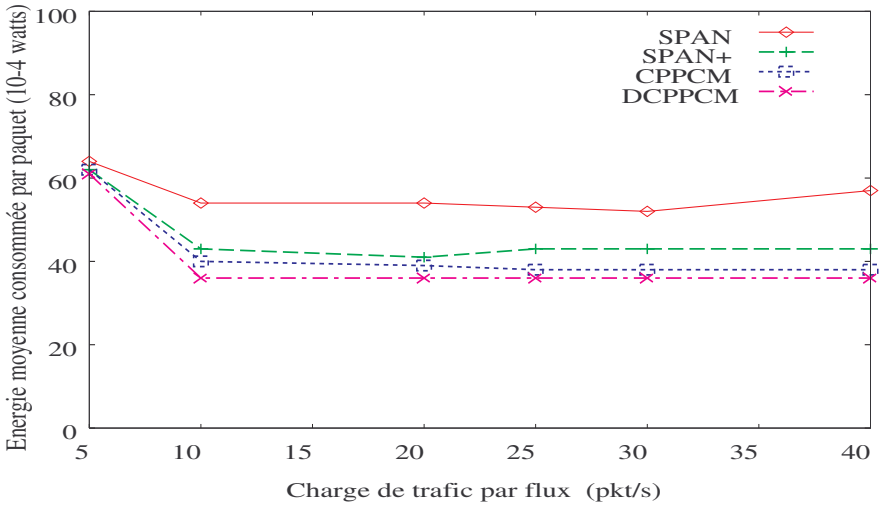


Figure 5.11 – Energie consommée par paquet en fonction de la charge du trafic par flux

du moment que les trois fenêtres seront entièrement exploitées par les différents noeuds pour acheminer les trois types de paquets identifiés.

Pour mieux mettre en valeur les avantages de la variante dynamique de notre approche de conservation d'énergie, on a choisi des topologies de 200 noeuds confinées dans une aire de simulation de 800 × 800 m durant cette deuxième étape. Les sources et destinations sont éparpillées sur les frontières de l'aire de simulation afin d'autoriser une communication multi sauts de l'ordre de 5 sauts en moyenne entre toute paire de source et destination. Notre objectif est de mettre en évidence l'utilité des bornes dynamiques pour s'adapter aux besoins spécifiques de tout noeud. Pour cette raison, on a focalisé sur la comparaison des mécanismes `DCPPCM` et `CPPCM` uniquement. Les paramètres de simulation adoptés précédemment sont utilisés dans ce scénario.

L'utilité de l'utilisation des bornes dynamiques est nettement observée dans la figure (5.12), où on expose le nombre de paquets délivrés avec succès. On voit clairement que la variante dynamique de notre mécanisme de conservation d'énergie surpasse la variante fixe. La flexibilité accordée essentiellement aux noeuds dominants leur permet de bien exploiter les ressources disponibles en terme de bande passante et d'énergie. La vue personnalisée cultivée par chaque noeud lui permet de profiter de la liberté du canal durant les périodes d'activité de ses voisins, grâce à son pouvoir d'étendre ou de raccourcir une fenêtre au profit d'une autre selon ses besoins. Cette caractéristique autorise les noeuds dominants à acheminer plus de paquets.

Par ailleurs, se conformer aux bornes rigides prédéfinies des différentes fenêtres peut être pénalisant, du moment qu'on limite ainsi la période active pour chaque type de paquets. Cette taille prédéfinie peut être parfois insuffisante pour écouler tous les paquets en attente. Le traite-

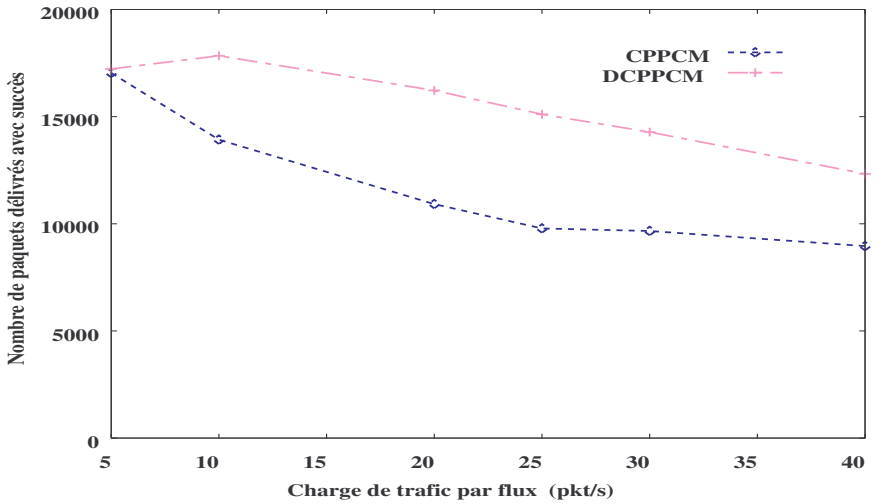


Figure 5.12 – Nombre moyen de paquets délivrés avec succès en fonction de la charge du trafic par flux

ment des paquets restant sera différé pour l'intervalle beacon suivant. D'autre part, tout noeud doit attendre le début de la fenêtre appropriée afin de pouvoir traiter les paquets en attente associés à cette dernière. Ainsi, avec la variante fixe, les paquets peuvent passer plus de temps dans les files d'attente des différents noeuds. Ceci justifie l'accroissement du délai de séjour moyen relatif au mécanisme `CPPCM` par rapport au délai mesuré pour `DCPPCM`, comme l'expose la figure (5.13). La souplesse dans la définition des bornes des différentes fenêtres avec `DCPPCM` nous permet d'accorder plus de temps aux paquets en cours d'acheminement, selon la classification spécifiée précédemment. De cette manière, on réduit les temps d'attente des paquets déjà injectés dans le réseau.

De ce fait, la version dynamique exploite plus efficacement les ressources disponibles, en particulier en terme d'énergie. Etant donnés que les noeuds dominants profitent des périodes d'inactivité du canal pour écouler plus de paquets, l'énergie de ces derniers ne sera pas gaspillée dans l'écoute passive du canal. Cette version dynamique permet d'écouler plus de paquets et d'exploiter convenablement l'énergie disponible. Ce fonctionnement justifie le gain en énergie consommée par paquet remarquée dans la figure (5.14). le coût en énergie consommée s'accroît avec la charge du trafic, vu que les tailles fixes n'autorisent pas le traitement de tous les paquets en attente. Dans une telle situation, l'attente de ces derniers sera prolongée afin de permettre leur traitement durant l'intervalle beacon suivant.

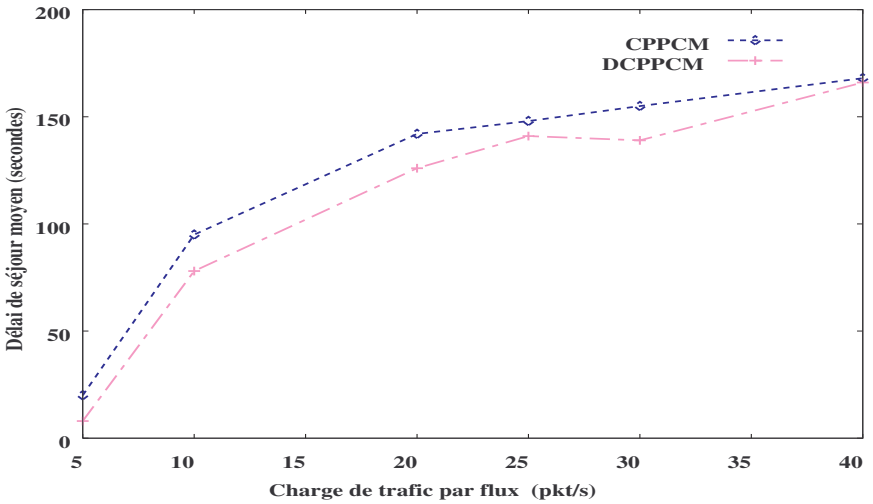


Figure 5.13 – Délai de séjour moyen en fonction de la charge du trafic par flux

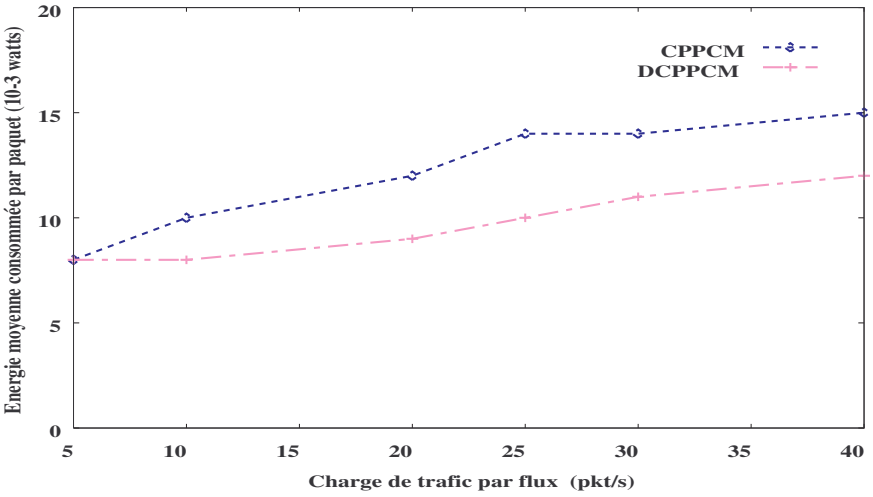


Figure 5.14 – Energie consommée par paquet en fonction de la charge du trafic par flux

5.4 Conclusion

L'énergie demeure encore une ressource critique dans un environnement ad hoc. Les efforts perçus dans la majorité des travaux s'orientent vers la réduction de l'énergie gaspillée. Dans cet axe, le mode conservation d'énergie a prouvé son aptitude à réduire l'écoute passive du canal et l'interception de paquets destinés à des noeuds voisins. Toutefois, de tels mécanismes ne prennent pas en considération l'impact du mode veille adopté sur les performances du système. La réduction du débit et l'augmentation du délai d'acheminement et du coût du paquet en terme d'énergie consommée peuvent être des répercussions négatives directes. Le recours à la notion d'ensemble dominant connexe pour allier conservation d'énergie et acheminement des paquets semble une solution prometteuse. Notre approche `CPPCM`, avec ses deux variantes, a exploité cette combinaison, tout en essayant de garantir une certaine qualité de service. Une classification adoptant les ressources consommées comme critère permet de privilégier les paquets déjà en cours d'acheminement afin de minimiser les temps d'attente au niveau des files d'attente des noeuds intermédiaire et l'énergie consommée par paquet. Les simulations réalisées confirment l'apport fourni par notre approche.

Chapitre 6

Proposition d'un algorithme de routage basée sur la présence d'une dorsale

Le passage à l'échelle demeure un frein majeur pour l'exploitation à grande échelle des réseaux ad hoc. L'absence d'infrastructure fixe impose la coopération de tous les noeuds pour assurer les fonctionnalités couramment fournies par cette infrastructure réseau. Maintenir la connexité du réseau et assurer un acheminement des données continu entre toute paire d'entités communicantes sont parmi les tâches critiques réparties sur l'ensemble du réseau. Le recours à un algorithme de routage est essentiel lors de l'établissement des routes et lors de leur maintenance. Toutefois, la majorité des algorithmes de routages souffrent du problème de passage à l'échelle, principalement à cause de la technique de diffusion déployée. Cette dernière surcharge le réseau et pénalise le trafic de données. Par ailleurs, le routage plat adopté par ces derniers impose la participation de tous les noeuds de manière égalitaire à ces tâches additionnelles, malgré l'hétérogeinité perçue en terme de ressources dans un environnement ad hoc.

Minimiser le trafic de contrôle additionnel relatif aux opérations de routage est un facteur clé pour assurer le passage à l'échelle. Cet objectif est d'autant difficile à réaliser en présence d'un grand nombre de noeuds relativement mobiles. Dans cet axe, le recours à une infrastructure virtuelle peut être bénéfique pour optimiser l'utilisation des ressources dans un environnement ad hoc. Ainsi, un sous ensemble de noeuds uniquement sera impliqué dans les tâches additionnelles afin de simplifier le processus de routage.

Parmi les protocoles de routage hiérarchique se basant sur une structure de clusterisation, le protocole `CGSR` (Clusterhead-Gateway Switch Routing) [29] est un exemple typique. En effet, les paquets de données entre une paire d'entités communicantes ne sont acheminés qu'à travers des chemins de la forme suivante "clusterhead_gateway_clusterhead_gateway_.. ". Tout noeud à proximité de plusieurs clusterheads joue le rôle de gateway. L'organisation du réseau en groupes est obtenue grâce à l'algorithme de clusterisation `LCC` (Least Clusterhead Change). Pour cet algorithme, un changement de chefs de groupes survient, seulement dans le cas où il y aurait une

fusion de deux clusters, ou dans le cas où un noeud sortirait complètement de la portée de tous les clusterheads du réseau. Tout noeud maintient deux tables : une table des membres du groupe et une table de routage à vecteur de distance. La première table permet de spécifier le chef de groupe pour chaque noeud. La diffusion périodique de cette table permet à tout noeud d'actualiser sa table. La deuxième table de routage réserve une entrée pour chaque cluster en spécifiant le chemin vers son clusterhead, indépendamment du nombre de noeuds membres attachés à ce dernier. En comparaison aux protocoles de routage à vecteur de distance, cette caractéristique permet de réduire la table de routage au niveau de chaque noeud et par conséquence la taille des paquets relatifs aux informations topologiques à diffuser. Pour acheminer un paquet de données, tout noeud consulte la table des noeuds membres pour extraire le clusterhead du noeud destination. Le parcours de la table de routage permet au noeud de retrouver le prochain saut vers le clusterhead du noeud destination. Une fois le paquet atteint le clusterhead de la destination, ce dernier le délivre au noeud concerné, du moment que c'est un membre de son cluster. Toutefois, le maintien de cette structure de clusterisation nécessite l'introduction d'un trafic additionnel pour la formation et la maintenance des clusters. Par ailleurs, la mobilité potentielle des différents noeuds rend difficile la tâche de maintenance de la table des membres actualisée.

De son côté, le protocole `HSR` (Hierarchical State Routing) [49, 73] se base sur une clusterisation multi niveaux. Les noeuds d'un même niveau logique sont groupés en clusters. Les clusterheads élus deviennent les membres du niveau suivant et s'organisent à leur tour en clusters. Les clusterheads issus de cette deuxième étape forment les membres du niveau suivant et ainsi de suite. L'objectif derrière cette organisation multi niveaux est de réduire le contrôle des données de routage (traitement, stockage et transmission) à chaque niveau hiérarchique. HSR adopte un routage à état des liens. Les noeuds du premier niveau gèrent l'état des liens relatifs à leurs voisins et les diffusent à l'intérieur de leur cluster. Le clusterhead résumé les informations collectées dans son cluster et les propage aux clusterheads voisins, via les noeuds gateways. La connaissance de la connexité entre clusterheads voisins autorise la formation du deuxième niveau hiérarchique de clusters. Le recours au processus de clusterisation récursif, mentionné précédemment, permet d'élire des clusterheads pour le deuxième niveau et ainsi de suite. Une fois les informations d'état de liens collectées à un niveau donné, chaque noeud virtuel la propage aux noeuds des clusters du niveau inférieur. De cette manière, tout noeud physique acquière une information topologique hiérarchique et dispose d'une adresse hiérarchique. Pour `HSR`, l'identifiant hiérarchique d'un noeud (HID) est une concaténation des adresses des différents noeuds faisant partie du chemin à partir du niveau hiérarchique le plus élevé jusqu'au noeud lui-même. Ainsi, le HID du noeud 5 égale à $HID(5) =< 1, 1, 5 >$ révèle que le clusterhead auquel le noeud 5 est directement rattaché est le noeud 1, et que le clusterhead du niveau hiérarchique suivant est aussi le noeud 1. Tout noeud est apte à actualiser localement son HID lors de la réception des mises à jour de routage, provenant des noeuds du niveau supérieur. Cet adressage hiérarchique est suffisant pour délivrer des paquets à un noeud destination quelconque, grâce à la table `HSR`.

Cependant, les changements fréquents affectant la topologie rendent difficile la localisation des noeuds et la maintenance de clusterisation multi niveaux et des identifiants hiérarchiques. La quantification du trafic de contrôle pour cet algorithme de routage hiérarchique a été abordée dans [84].

D'autres approches ont proposé des algorithmes pour la formation d'un ensemble dominant connexe [33, 80, 94, 96], ces noeuds dominants forment la dorsale pour assurer les fonctionnalités de routage. Cependant, ces approches ont focalisé sur la description des mécanismes de formation d'un ensemble dominant connexe, sans trop détailler l'approche de routage déployée lors de l'établissement. Par ailleurs, en présence de la mobilité, le maintien de cette dorsale actualisée impose l'introduction d'un trafic de contrôle additionnel. Ces opérations de maintenance et de réparations sont nécessaires afin que l'infrastructure virtuelle puisse offrir un support efficace aux protocoles de haut niveau (routage, conservation d'énergie, sécurité, ...). La quantification de ce trafic additionnel et la maintenance des chemins établis en cas de défaillance de liens n'ont pas été aussi abordées.

Dans ce contexte, nous proposons notre algorithme de routage "Layred Cluster based Routing" (`LCR`). Notre idée de base est d'exploiter la présence d'une dorsale pour simplifier le processus de routage.

6.1 Principe de LCR

Une approche de routage proactive permet d'offrir des routes prêtes à l'emploi en cas de besoin, à travers la maintenance des tables de routage des différents noeuds actualisées. Une telle approche de routage repose sur un échange périodique des informations topologiques, puisant ainsi des ressources limitées du réseau. Même en ayant recours à une infrastructure virtuelle pour limiter ce gaspillage, on aura à maintenir des routes, qui peuvent ne jamais être utilisées. Par ailleurs, dans un environnement dynamique, de telles routes peuvent devenir rapidement invalides. Ainsi, l'exploitation brève des routes établies ne justifie pas les ressources consommées en terme de bande passante et d'énergie pour leur établissement et leur maintenance. Pour cet effet, nous nous plaçons dans le cadre d'un routage réactif afin d'éviter la surcharge du réseau de messages de contrôle additionnels.

A travers l'assignation des tâches de routage à l'ensemble dominant généré par notre algorithme de clusterisation `TBCA`, on réduit le nombre de noeuds participants au processus de routage et le nombre de paquets de contrôle échangés par ce dernier. Du moment que notre algorithme de clusterisation opère au niveau de la couche MAC et que toutes les informations nécessaires pour l'acheminement des paquets de données sont disponibles au niveau de cette couche, la migration du processus de routage au niveau de la couche MAC s'avère nécessaire. En effet, au niveau de cette couche, tout noeud disposerait de la liste de ses voisins dominants, une fois la phase de

clusterisation finalisée. Il est apte à collecter des informations sur le voisinage à travers l'écoute passive des trames échangées. Notre objectif est de trouver un chemin entre les entités désirant communiquer en un temps très réduit. Pour cet effet, on va privilégier les paquets de contrôle lors de l'accès au canal.

6.1.1 Etablissement des chemins

Une station, S désirant communiquer avec une destination donnée D, vérifie en premier l'existence d'une route valide vers cette destination dans sa table de routage. Si aucune route n'est disponible, elle initie une phase de découverte de routes à travers l'envoi d'une requête RREQ. Pour accélérer le processus de formation de routes et réduire par conséquence les temps d'attente subies par les paquets de données à acheminer, on va privilégier le noeud S lors de l'accès au canal. Pour cette fin, toute station source S doit attendre une période d'écoute égale à SIFS avant de calculer un délai aléatoire d_1 pour l'envoi de sa requête. Ce délai aléatoire permet d'éviter les collisions éventuelles lors d'un accès simultané de plusieurs machines au canal. La décrémentation de ce temporisateur s'effectue conformément à l'algorithme standard de backoff. Lorsque le canal est occupé, on suspend la décrémentation du délai d_1 pour ne pas compromettre les transmissions en cours. A l'expiration de ce dernier, la station est autorisée à diffuser sa requête. La station source déclenche un temporisateur **Temps_requete**. Ce temporisateur estime la période au bout de laquelle la source doit relancer une nouvelle phase de recherche, si la première n'a pas abouti. Ainsi, à l'expiration de ce temporisateur, la station source doit diffuser une nouvelle requête, si elle n'a pas reçu de réponse pour la première requête diffusée. L'incrémentation du numéro de séquence de la requête est nécessaire.

Etant donnée que la tâche de routage est assignée aux noeuds dominants, uniquement ces derniers sont appelés à traiter et à rediffuser les requêtes, exception faite pour la station destination ayant le statut d'un noeud ordinaire. Une station ordinaire n'a pas à participer à la phase de recherche de routes. A la réception d'une requête, une station ordinaire peut mettre à jour la liste de ses voisins immédiats et la liste des noeuds dominants voisins, étant donné que chaque noeud inclut son statut avant de diffuser la requête.

Lors de l'interception de la requête par un noeud dominant, on va différencier le traitement réalisé au niveau d'un noeud clusterhead et d'un noeud gateway. Notre objectif est d'éviter les collisions éventuelles dues à un envoi simultané des requêtes par des noeuds voisins. Si tous les noeuds dominants essaient d'accéder au canal de manière égalitaire, le risque d'occurrence de collisions sera élevé. Dans une telle situation, on réduit les chances de retrouver un chemin valide, étant donnée que les collisions peuvent nuire à la bonne réception des requêtes diffusées par les noeuds avoisinants et en particulier par la destination. Pour cette fin, tout noeud clusterhead devra attendre une période d'écoute égale à PIFS avant de calculer un délai aléatoire d_1. Ce délai d_1 est un nombre aléatoire de slots compris entre 0 et une borne B_1. A l'expiration de

ce backoff, le noeud clusterhead est autorisé à diffuser sa requête. Un noeud gateway aura à attendre une période d'écoute égale à DIFS avant de calculer son backoff. Afin d'éviter les collisions pouvant survenir lors de l'envoi des requêtes par des clusterheads et des gateways voisins, on doit s'assurer que ces envois vont se faire en différé. Pour cela, le délai d_1 choisi par un noeud gateway est un nombre aléatoire compris entre les bornes B_1 et B_2, avec $B_2 > B_1$. Du moment que la décrémentation des délais aléatoires se fait conformément à l'algorithme de backoff, on garantit que la diffusion des requêtes par les noeuds clusterheads devancera la diffusion des requêtes par les noeuds gateways voisins. Par ailleurs, tout noeud dominant garde trace des requêtes diffusées afin d'éviter la retransmission de requêtes dupliquées et surcharger par conséquence le réseau. Ainsi, le noeud dominant maintient une table, dite **Table_requête**, au niveau de laquelle il stocke les informations relatives à toute requête, à savoir l'adresse de la source initiale, l'adresse de la destination finale, l'adresse de l'émetteur de la requête, le numéro de séquence de la requête et son éloignement par rapport à la source ayant initié cette phase de découverte de route en nombre de sauts. Avant la diffusion de toute requête, le noeud dominant parcourt cette table **Table_requête** afin de s'assurer que la requête en cours de traitement n'a pas été traitée auparavant. If vérifie si une entrée, ayant la même source, la même destination et le même numéro de séquence, existe déjà ou pas.

Pour réduire le nombre de paquets de contrôle relatifs au processus de routage, on a introduit une nouvelle notion, dite la direction. L'un des apports de notre mécanisme de clusterisation `TBCA` est d'offrir une organisation des noeuds par rapport au noeud référence, le noeud DN. Notre algorithme de clusterisation progresse en couches. Une fois le processus de clusterisation finalisé, tout noeud connaît le niveau de sa couche par rapport au noeud DN. On considère la topologie exposée dans la figure suivante. Le noeud DN et ses voisins immédiats forment le premier niveau. Les clusterheads à deux sauts et leurs noeuds membres forment le deuxième niveau, à l'instar des noeuds i, j, k et h.

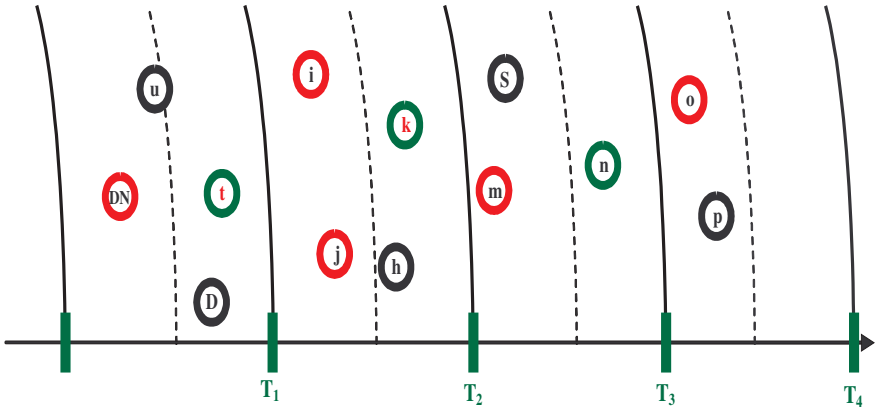


Figure 6.1 – Organisation du réseau en couches grâce au mécanisme de clusterisation `TBCA`

Dans notre algorithme de routage, on va exploiter cette organisation pour localiser l'emplacement de la destination et des noeuds intermédiaires par rapport à la source. Ainsi, le noeud source indique son niveau dans la requête diffusée et met la valeur du champ direction à zéro. En recevant cette requête, les noeuds dominants vont se baser sur ces deux informations pour décider de traiter la requête ou pas. Les voisins immédiats, du même niveau que la source, doivent rediffuser la requête, en préservant la valeur zéro du champ direction. Si le niveau du noeud voisin est inférieur à celui du noeud source, on assigne la valeur 1 au champ direction, tandis qu'on lui attribue la valeur 2 si le niveau du noeud voisin est supérieur au niveau du noeud source. Ainsi, la valeur 1 du champ direction oriente la diffusion de la requête vers les couches de niveau inférieur par rapport à la source, alors que la valeur 2 dirige la diffusion vers les couches de niveau supérieur par rapport à la source. Les autres noeuds dominants vont traiter à leur tour toute requête reçue, si le niveau de la source est supérieur au niveau du noeud concerné et la direction est fixée à 1 ou si le niveau de la source est inférieur au niveau du noeud en question et la direction est mise à 2.

Dans l'exemple de la figure (6.1), le noeud S désire établir une communication avec le noeud destination D. En diffusant sa requête, la source S spécifie son niveau (3) et met à zéro le champ direction. En interceptant cette requête, le noeud voisin k met la valeur du champ direction à 1, du moment que son niveau est inférieur à celui de la source. Cette indication permet d'orienter la diffusion de la requête vers les couches de niveau inférieur. Ainsi, les noeuds dominants appartenant aux niveaux inférieurs traitent la requête diffusée par le noeud k, alors que les noeuds appartenant aux niveaux supérieurs n'ont pas à traité une telle requête. Si, on considère le noeud m, ce dernier va ignorer la requête diffusée par le noeud k sans effectuer aucune vérification. Il n'a pas à parcourir sa table **Table_requête** pour vérifier s'il a déjà traité cette requête ou pas.

Placer le noeud DN, responsable du déclenchement du processus de clusterisation, aux frontières du réseau, autorise un tel dénouement et offre une telle organisation. Du moment que le noeud DN et ses voisins forment toujours le premier niveau, le reste des noeuds formeront par la suite les niveaux suivants, donc auront toujours un niveau supérieur à 1. Or, un noeud DN localisé à l'intérieur du réseau pose un problème, du moment qu'on aurait une symétrie de niveaux autour du noeud DN. Pour mieux illustrer la situation, on considère la topologie exposée dans la figure (6.2). Dans ce cas, le noeud DN est situé au centre du réseau. Les voisins clusterheads à deux sauts du DN et leurs membres (noeuds i, j, t et h), localisés à gauche de ce dernier, forment le niveau 2. Or, le clusterhead o et ses noeuds membres D et p forment aussi le niveau 2. En effet, les voisins immédiats du noeud DN envoient un ACK_BT. L'interception de ce dernier par les voisins à deux sauts leur permet d'initier la phase de clusterisation au niveau de leur couche.

Pour mieux illustrer la situation, on reprend le scénario précédent concernant la machine S

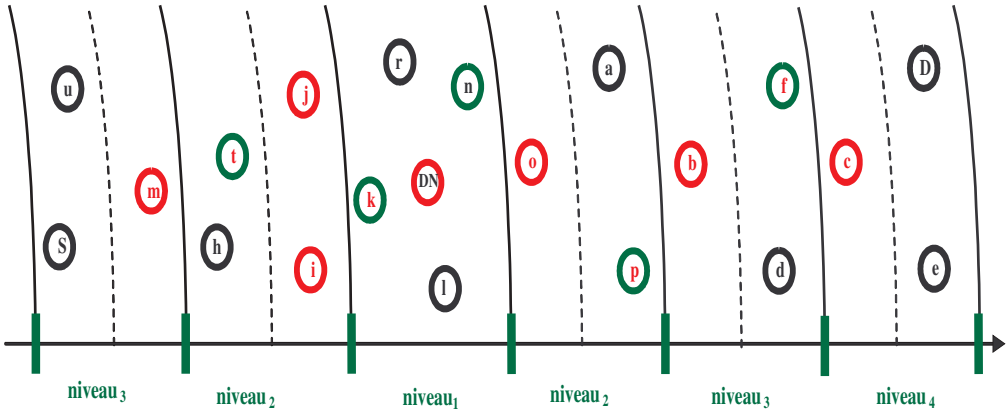


Figure 6.2 – Exemple

voulant communiquer avec la machine *D*. En initiant la phase de recherche de routes, la machine S diffuse une requête RREQ, en mettant le champ direction à 0. Cette valeur particulière du champ direction oblige les voisins dominants immédiats de la source à traiter la requête. Dans notre exemple, cette requête sera traitée par le noeud m, appartenant au même niveau. Ce dernier conserve la même valeur du champ direction lors de la diffusion de cette requête. Le noeud t, du niveau suivant, va modifier la valeur du champ direction et lui assigner la valeur 1, du moment que le niveau de la source est supérieur au niveau du noeud en question. Ainsi, on oriente la diffusion de la requête vers les noeuds des couches de niveau inférieur par rapport à la source. La modification du champ direction ne sera pas nécessaire, une fois il se voit attribuer une valeur différente de 0. Toutefois, en préservant un tel raisonnement, la requête de la source va être propagée à travers les niveaux 2 et 1. Les noeuds dominants du niveau 2 (à savoir les noeuds h et f), situé du côté opposé du noeud DN, vont à leur tour diffuser la requête. En effet, le niveau de la source est supérieur au niveau des noeuds de cette couche et la direction est mise à 1. Toutefois, la diffusion de la requête va s'arrêter au niveau de la couche 3. Le noeud h va ignorer le traitement de cette requête. En effet, les noeuds dominants vérifient en premier la valeur du champ direction et leur niveau par rapport à la source. Vu que le champ direction est mis à 1 et le niveau de la source n'est pas supérieur au niveau du noeud en question, la requête va être ignorée par le noeud. Ainsi, la diffusion de la requête va être stoppée au niveau de la couche 3 dans notre exemple et elle ne va pas parvenir à atteindre le noeud destination.

Pour remédier à ce problème, une légère modification s'impose. Les noeuds dominants, du niveau 1, doivent modifier les champs niveau de la source et direction. Le champ direction doit être réinitialisé à zéro et le niveau de la source se voit attribuer la valeur 1. Du moment que tout noeud dominant devra traiter toute requête ayant le champ direction à 0, les noeuds, des niveaux

adjacents, devront prendre en considération une telle requête et vérifier ultérieurement s'ils ont déjà traité une copie de la requête reçue. En recevant une telle requête, les noeuds i et j vont la supprimer après avoir vérifié qu'elle a été déjà traitée. De son côté, le noeud o va modifier le champ direction à 2 pour orienter la diffusion de la requête vers les couches de niveau supérieur à 1. De cette manière, la requête va être propagée jusqu'à atteindre le noeud destination.

Ainsi, la requête diffusée doit inclure des informations additionnelles spécifiques à notre algorithme de routage. Outre le numéro de séquence de la requête et les adresses des noeuds émetteur, source et destination, le noeud source doit spécifier son niveau et la direction pour orienter la diffusion de la requête. L'utilité de cette notion se concrétise mieux lors des opérations de réparation des routes suite aux défaillances pouvant survenir à cause de la mobilité potentielle des noeuds.

Lorsque la station destination intercepte la requête, elle ajoute une entrée à sa table de routage et génère le paquet réponse, baptisé RREP. Afin de pouvoir localiser la position géographique de la destination par rapport à la source et par rapport aux noeuds intermédiaires, la destination spécifie son niveau dans la réponse renvoyée. Lors de l'envoi de cette réponse, la destination doit être la plus prioritaire lors de l'accès au canal. L'objectif est de faire parvenir plus rapidement la réponse à la machine source concernée. L'établissement rapide des chemins permet de réduire les délais d'attente des paquets nouvellement générés au niveau du noeud source. De ce fait, la destination doit attendre une période d'écoute égale à SIFS avant d'envoyer un paquet RTS vers la station source, ou la station ayant émis la requête. A travers cet échange de RTS/CTS, on réserve le canal dans les voisinages respectifs des deux entités communicantes. Suite à un échange réussi des paquets RTS/CTS, la destination attend une période SIFS avant d'envoyer le paquet réponse. Le récepteur doit acquitter suite à la réception de ce RREP. On rappelle que pour le mécanisme CSMA/CA, une fois le canal libre, toute station désirant émettre doit attendre une période d'écoute égale à DIFS avant de calculer un délai aléatoire. En attendant uniquement une période SIFS avant de répondre, la destination est privilégiée par rapport aux autres stations en concurrence pour l'accès au canal. La réservation du canal à travers un échange RTS/CTS permet d'éviter le problème de stations cachées et les éventuelles retransmissions en cas de collisions.

Le même traitement sera accordé au paquet RREP lors de son acheminement vers la station source. Tout noeud dominat intermédiaire impliqué sur le chemin de retour vers la station source, ajoute une entrée dans sa table de routage. Chaque entrée comprend l'adresse de la station source, l'adresse de la destination, le niveau de la source, le niveau de la destination, la longueur du chemin en nombre de sauts et l'éloignement du noeud intermédiaire par rapport au noeud source. A travers la liste des requêtes traitées, le noeud peut retrouver le niveau de la source, sa position par rapport à cette dernière et le noeud qui le précède vers lequel la réponse doit être envoyée. A l'instar du noeud destination, le noeud va privilégier le traitement de la réponse reçue

par rapport aux autres paquets de données. La même démarche que le noeud destination va être adoptée lors de l'envoi du paquet RREP tout au long de la route de retour vers le noeud source.

6.1.2 Maintenance des routes

La migration du processus du routage au niveau de la couche MAC nous libère de l'utilisation des messages additionnels hello pour vérifier l'existence des noeuds voisins. Au niveau de la couche MAC, toute transmission de paquets de données doit être acquittée. Le départ d'un noeud voisin impliqué dans une session de communication peut être déduit si on atteint le nombre maximal de retransmissions et aucune réponse ne nous parvient. Lors de la détection de la défaillance d'un lien, la phase de maintenance peut s'effectuer à travers une réparation locale ou en reportant une notification à la source pour initier une nouvelle phase de recherche de route. L'emplacement du noeud ayant détecté la rupture est important lors de la prise de cette décision sur la manière à effectuer l'opération de maintenance.

Si la distance séparant le noeud en question de la source est supérieure à la moitié du chemin menant vers la destination finale, on a intérêt à opter pour une réparation locale. Ce noeud est plus proche de la destination et le chemin jusqu'à ce point semble inaffecté du moment que le noeud reçoit encore les paquets de données en provenance de la source. Ainsi, le noeud va envoyer une requête pour chercher une route vers la destination finale. La connaissance de l'ancien niveau de la destination permet de localiser la position géographique de cette dernière. Cette information, extraite à partir de la table de routage, permet d'orienter la diffusion de la requête. Etant donnés que les mouvements des noeuds ne sont pas drastiques et radicaux, l'éloignement des noeuds sera progressif. Un noeud mobile sera toujours dans les alentours de la même zone géographique, au début de son déplacement. Lors de la diffusion de la requête, les noeuds, appartenant au même niveau que l'émetteur, doivent traiter la requête, alors que les noeuds des niveaux adjacents se basent sur la vérification du niveau de la source de la requête et de la direction. En spécifiant la direction, on contribue à réduire le nombre de paquets de contrôle échangés durant cette phase de maintenance.

Dans le cas où la distance est inférieure à la moitié du chemin, le noeud doit informer la source afin qu'elle puisse redéclencher le processus de découverte de route en lui envoyant un paquet RERR. En comparant son niveau par rapport au niveau de la source, le noeud peut déduire la localisation de la source par rapport à sa position courante. En spécifiant la direction de diffusion, on limite le nombre de noeuds dominants ayant à propager le message RERR vers la source. L'implication de noeuds dominants, autres que ceux spécifiés sur le chemin menant de la source vers la destination, permet d'offrir plus de garantie quant à la réception de la notification d'erreur par le noeud source, en particulier en présence d'une mobilité élevée pouvant affectée le chemin entre la source et le noeud ayant détecté la rupture. Ainsi, les noeuds du même niveau que l'émetteur du paquet RERR rediffusent la notification reçue, après avoir garder une trace

de cette dernière dans la liste **liste_rerr**. Les noeuds dominants, de niveaux adjacents, vérifient s'ils sont concernés par ce paquet en se basnat sur le champ niveau de l'émetteur initial et le champ direction. Si le champ direction est mis à 2, uniquement les noeuds faisant partie des niveaux supérieurs par rapport à l'émetteur initial sont impliqués dans l'opération de diffusion. Les noeuds dominants de niveaux inférieurs participent à la diffusion du paquet RERR lorsque le champ direction est initialisé à 1. Tout noeud dominant garde trace des paquets RERR traités dans la liste **liste_rerr** afin d'éviter le traitement des duplications.

On reprend l'exemple précédent, on suppose que le noeud dominant i est impliqué dans l'opération de transfert entre la source S et la destination D. Le noeud i a détecté la défaillance d'un lien. Du moment que le niveau de la source est supérieur au niveau du noeud i, il met à 2 le champ direction pour diriger la diffusion vers les couches supérieures. Ainsi, uniquement une partie des noeuds dominants sera impliquée dans la diffusion de cette notification de défaillance de lien. A la réception de ce message RERR, la source déclenche une nouvelle opération de découverte de route en diffusant une nouvelle RREQ. La mise à jour de la table de routage est requise au niveau du noeud source et des noeuds impliqué dans l'opération de transfert entre la source et la destination.

6.2 Evaluation de performances

Pour évaluer les performances de notre algorithme de routage, on va procéder sur deux étapes. La première phase vise à mettre en évidence l'apport de notre mécanisme de routage à assurer le passage de l'échelle. La deuxième partie vise à quantifier le coût en terme de trafic de contrôle relatif au processus de routage.

Lors de la première phase, on a comparé les performances de `LCR` par rapport à celles de `AODV`. L'algorithme `AODV` est un algorithme de routage réactif. Toute source, voulant communiquer avec une destination donnée, initie une phase de découverte de routes à travers la diffusion d'un paquet RREQ. Tous les noeuds participent à la propagation de cette requête à travers le réseau. Ainsi, de proche en proche, la requête transite à travers le réseau jusqu'à atteindre la destination. Cette dernière réplique alors en renvoyant un paquet RREP, qui parcourt le chemin inverse en direction de la source. En cas de défaillance de lien, le noeud, ayant détecté cette rupture, envoie une notification d'erreurs vers la source. A la réception de cette notification, la source déclenche une nouvelle phase de découverte de routes. Lors de ces premières expérimentations, on a focalisé sur deux critères, à savoir le nombre de paquets de contrôle échangés par le processus de routage durant toute la simulation et le nombre de paquets délivrés avec succès vers toutes les destinations.

Pour cette étape, on a choisi différentes topologies de 200 noeuds, répartis sur une aire de simulation de $500 \times 500m$. Tous les noeuds sont mobiles, exception faite pour les noeuds

sources et destinations. Ces derniers sont toujours placés aux extrémités de l'aire de simulation afin d'autoriser une communication multi sauts. Notre algorithme de clusterisation repose sur la présence du noeud DN, responsable de l'initiation du processus de clusterisation. Ce dernier va être aussi stable durant toutes les simulations. Le modèle de mobilité *random way point*, présenté dans le chapitre 3, est aussi déployé dans cette phase avec une période de pause de 10 secondes. La vitesse maximale des noeuds mobiles est de 5 m/s. Durant ces expérimentations, on a fait varier le nombre de sources des flux CBR, chaque flux CBR maintient un taux de transfert de 5 paquet/seconde.

Le tableau suivant (6.1) résume les paramètres adoptés durant cette partie :

Paramètres du réseau	
Portée	250m
Bande passante	2 Mbps
Durée de simulation	500s
Paramètres de TBCA	
Périodicité de MaJ de la clusterisation	10 intervalles beacon
(D_{1max}, D_{1min})	(31, 25)
(D_{1pmin}, D_{1npmin})	(15, 31)
$NbrMax\_gw\_CH$	5
$NombreMoy\_gw/CH$	4
Nombre de périodes de clusterisation successives en tant que dominant	5

Table 6.1 – Paramètres utilisés dans la simulation

D'après les résultats exposés dans la figure (6.3), on remarque une augmentation considérable du nombre de paquets de contrôle pour AODV avec la croissance du nombre de sources. En effet, chaque source initie une phase de découverte à travers la diffusion d'une requête. Cette phase est réitérée à chaque défaillance de lien détectée. Les diffusions excessives des paquets de contrôle relatifs au processus de routage peuvent nuire à l'opération d'établissement des routes, du moment que les éventuelles collisions peuvent empêcher la bonne réception des requêtes et notifications d'erreurs échangées. Lorsque le nombre de paquets de contrôle croît considérablement conjointement avec le nombre de sources, les performances du réseau se détériorent. Cette surcharge excessive encombre le réseau et pénalise le trafic transitant à travers ce dernier. Ce trafic additionnel étouffe le trafic de données, du moment qu'il consomme les ressources en terme de bande passante et maintient le canal constamment occupé. Ce fonctionnement intrinsèque justifie le nombre de paquets délivrés avec succès décroissant avec la croissance du nombre de sources, consolidant le fait qu'un algorithme de routage à plat souffre du problème de passage à l'échelle. Pour notre algorithme de routage, on remarque que le nombre de paquets de contrôle est nettement réduit par rapport à celui mesuré pour AODV. L'assignation des tâches additionnelles à un sous ensemble de noeuds contribue à minimiser le trafic de contrôle. Dans ce contexte, l'algorithme de clusterisation déployé revêt une grande importance, du moment que la taille de

l'ensemble dominant généré est un critère clé pour évaluer l'efficacité du mécanisme de clusterisation. Malgré le nombre croissant de noeuds sources, la grande différence persiste et on maintient notre avantage. Toutefois, limiter des opérations de communications aux noeuds dominants n'est pas l'unique raison ayant contribué à avoir cette réduction importante du nombre de paquets de contrôle.

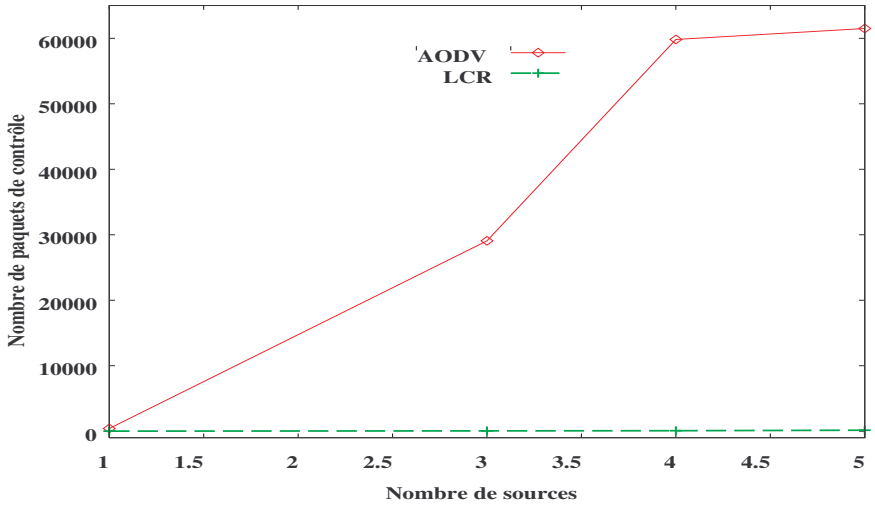


Figure 6.3 – Nombre moyen de paquets de contrôle en fonction du nombre de sources

Les techniques intrinsèques de notre algorithme de routage ont adhéré à l'obtention de ce gain. La différentiation adoptée lors de l'envoi des paquets RREQ et RERR permet d'éviter les collisions éventuelles lors de leur diffusion à travers le réseau. Ce fonctionnement permet d'atténuer les impacts négatifs d'une phase de découverte de chemins basée sur une technique de diffusion, augmentant par conséquence les chances de retrouver un chemin valide. Ceci justifie le gain en terme de débit mesuré lors du déploiement de notre algorithme de routage, comme l'expose la figure (6.4).

Dans la deuxième phase d'évaluation, on a visé la quantification du coût du routage hiérarchique. Pour cette fin, on a comparé les performances du système lors du déploiement d'un acheminement géographique et lors du déploiement de notre algorithme de routage LCR. On a choisi 100 noeuds, confinés dans une aire de simulation de $500 \times 500m$, où les noeuds se déplacent à une vitesse maximale de 5 m/s selon le modèle de mobilité de *random way point*. La période de pause est fixée à 1 seconde entre les déplacements successifs des différents noeuds. A l'instar de l'étape précédente, les noeuds sources et destinations et le noeud DN sont stationnaires. Dix flux CBR transitent à travers le réseau avec un taux de transfert de 5 pkt/sec. Le même scénario a été repris en allongeant la période de pause entre les déplacements successifs à 10 secondes.

D'après les résultats exposés dans les figures (6.5) et (6.6), on remarque que les perfor-

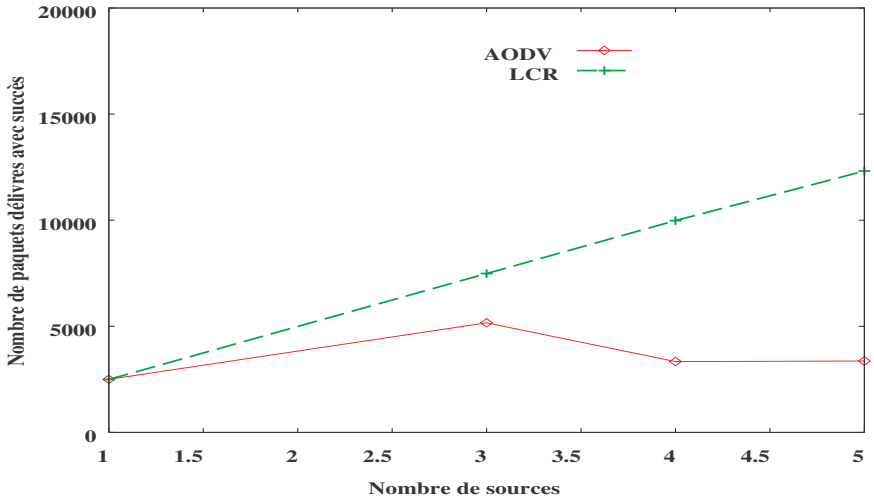


Figure 6.4 – Nombre moyen de paquets délivrés avec succès en fonction du nombre de noeuds sources

mances du réseau sont meilleures lors du déploiement de l'acheminement géographique. Dans une telle situation, le coût du routage est nul, du moment qu'on assume que tout noeud connaît la position géographique de ses voisins sans échange préalable de messages de contrôle. Toutefois, les performances du réseau lors de l'utilisation de notre algorithme de routage `LCR` se rapprochent de celles obtenues avec l'acheminement géographique, malgré le recours à des messages de contrôle additionnels pour l'établissement et la maintenance des routes. En effet, le privilège accordé aux paquets de contrôle dédiés au processus de routage contribue à accélérer l'établissement des chemins et leur réparation suite aux défaillances dues à la mobilité des noeuds. Le traitement différencié accordé aux paquets de contrôle en fonction du statut du noeud permet d'éviter l'occurrence de collisions lors de la diffusion de ces derniers, augmentant ainsi les chances de retrouver rapidement une route valide. La migration du processus de routage au niveau de la couche MAC nous a permis d'éviter l'échange d'informations entre les différentes couches. Ainsi, la détection des départs de voisins sera plus facile et on peut réagir plus rapidement pour s'adapter à la nouvelle topologie. De cette manière, on augmente la réactivité de notre algorithme face aux changements fréquents de la topologie. La périodicité d'actualisation de la structure de clusterisation revêt aussi une grande importance, du moment qu'elle peut affecter la stabilité de cette infrastructure et par conséquence les performances des protocoles reposant sur cette dernière. Une actualisation fréquente affecte la stabilité de la dorsale formée, alors que des mises à jour éloignées contribuent à avoir une infrastructure non homogène. Pour cette raison, il faut trouver la périodicité appropriée pour actualiser l'infrastructure virtuelle. En observant la courbe verte en particulier, on remarque une amélioration du débit en augmentant la périodicité de mise à jour. Toutefois, le nombre de paquets délivrés lorsque les mises à jour sont trop espacées.

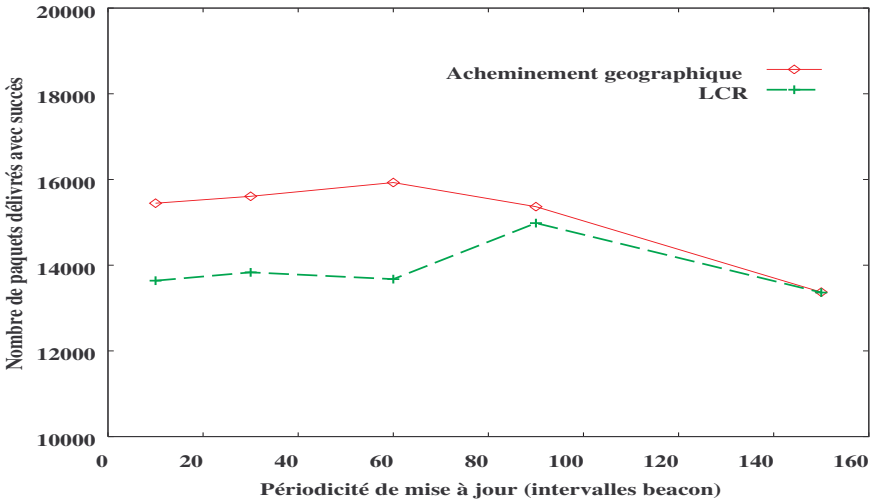


Figure 6.5 – Nombre moyen de paquets délivrés avec succès en fonction de la périodicité de mise à jour de la structure de clusterisation (pause = 1 seconde)

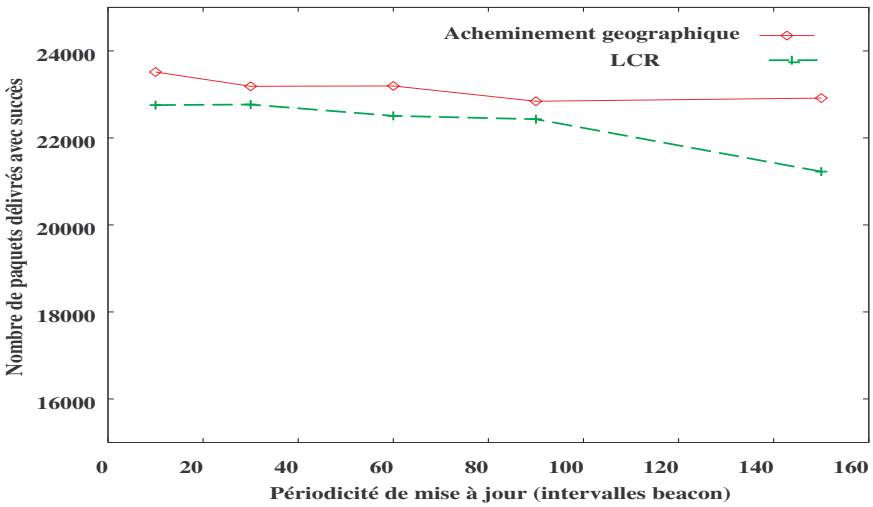


Figure 6.6 – Nombre moyen de paquets délivrés avec succès en fonction de la périodicité de mise à jour de la structure de clusterisation (pause = 10 secondes)

Lorsque la période de pause est allongée, la différence entre les deux courbes relatives aux deux mécanismes s'atténue, comme on peut le noter à travers la figure (6.6). En effet, cette période de pause de 10 secondes augmente la stabilité du réseau, vu que les déplacements éventuels des différents noeuds seront moins fréquents. Pour une période de pause de 1 seconde, les noeuds seront en mouvement continu, impliquant des maintenances fréquentes des chemins établis. Par ailleurs, la fréquence de maintenance de la dorsale affecte considérablement les performances des protocoles reposant sur cette dernière. Des maintenances fréquentes pénalisent la stabilité du système. Les performances du système se détériorent aussi, suite à l'augmentation de la périodicité de mise à jour de la structure de clusterisation lors du déploiement de `LCR` ou de l'acheminement géographique. Dans une telle situation, les noeuds dominants peuvent perdre la connexité avec leurs confrères dominants, ainsi, la dorsale sera incapable de supporter efficacement les autres protocoles de haut niveau.

6.3 Conclusion

Dans ce chapitre, nous avons introduit un nouvel algorithme de routage `LCR` qui repose sur l'existence d'une dorsale générée grâce à notre mécanisme de clusterisation `TBCA`. Notre algorithme `LCR` exploite conjointement les apports d'une dorsale et les caractéristiques de notre mécanisme de clusterisation. En effet, ce dernier nous offre une organisation des noeuds en couches par rapport au noeud de référence DN. De cette manière, tout noeud, impliqué dans une opération d'échange, peut cultiver une idée sur la localisation géographique de la source et la destination par rapport à son emplacement actuel. Cette connaissance lui permet de réagir convenablement en cas de rupture de liens. Cette organisation en couches nous permet aussi de limiter encore plus le nombre de paquets de contrôle échangés lors des phases de découverte de routes et de leur maintenance. Les simulations conduites confirment l'efficacité de notre algorithme de routage, même dans un environnement mobile.

Conclusion Générale et Perspectives

L'aptitude d'un réseau ad hoc à s'organiser et s'auto-configurer de manière autonome sans avoir recours à une infrastructure fixe constitue la principale caractéristique attrayante ayant contribué à proliférer l'intérêt porté à ce type de réseau. Toutefois, l'absence de structure fait que l'ensemble des services, à l'instar du routage, est généralement réparti entre toutes les entités. L'implication continuelle de tout noeud à ces tâches additionnelles peut être pénalisante, étant donnée l'autonomie énergétique réduite des différents hôtes.

Or, l'utilité de tels réseaux se concrétise lors de leur déploiement dans les aéroports, les hôtels, les espaces de rencontres entre professionnels et les campus universitaires. Un tel déploiement fait intervenir un grand nombre d'utilisateurs nomades. Pour cette raison, les performances du réseau ne doivent pas pour autant chuter de façon drastique lorsque le nombre de participants augmente. Il doit passer à l'échelle, et présenter un trafic de contrôle négligeable afin de ne pas perturber le trafic de données. Or, adopter une vue plate du réseau et attribuer des rôles égalitaires à tous les noeuds présentent des limitations au passage à l'échelle, vu la croissance du trafic de contrôle avec le nombre de noeuds.

Dans ce contexte, le concept de clusterisation peut être mis à profit afin de faire face aux problèmes de passage à l'échelle et d'accroître les performances du système. La structuration d'un réseau permet de rendre son exploitation plus aisée et de supporter plus efficacement les protocoles de haut niveau. C'est ainsi que plusieurs mécanismes de clusterisation ont vu le jour pour organiser un réseau ad hoc de grande envergure. Le coût induit en termes de trafic additionnel introduit et de temps n'est pas pris en considération par la plupart des propositions.

A cet effet, nous avons proposé dans cette thèse une approche de clusterisation permettant d'assurer rapidement la formation d'un ensemble dominant connexe, en prenant en compte les caractéristiques intrinsèques à un environnement ad hoc. La négligence des ces caractéristiques peut influencer le déroulement de la phase de clusterisation et poser des problèmes de convergence. Nous avons proposé une solution entièrement distribuée dans laquelle les hôtes coopèrent de manière organisée. Cette approche est adaptée aux réseaux à grande échelle car le nombre de participants à la phase de clusterisation à un instant donné est restreint, ce qui limite le nombre de paquets de contrôle échangés et diminue le risque d'occurrence de collision. Par ailleurs, le

protocole de clusterisation tire profit des collisions entre paquets dédiés à la clusterisation pour accélérer le processus de clusterisation. Ce protocole est périodiquement re-déclenché afin de prendre en considération les changements de topologie survenant dans le réseau, tout en garantissant une certaine stabilité. Ainsi, les noeuds peuvent bénéficier d'une dorsale actualisée apte à assurer le support efficace des autres protocoles. Nous avons réalisé une évaluation des performances de notre protocole de clusterisation pour montrer son applicabilité dans les réseaux à grande envergure. Les résultats de la série d'expérimentations conduites valident l'efficacité de notre protocole et confirment son aptitude à améliorer les performances du système.

La formation d'une dorsale n'est pas une fin en soi. Nous avons exploité la présence de cette dernière pour conserver de l'énergie, étant donnée qu'elle demeure la ressource la plus critique dans un réseau ad hoc. Notre objective est de tirer parti de cette hiérarchie afin de prolonger la durée de vie du réseau. Pour éviter l'extinction des noeuds n'ayant pas une autonomie en énergie élevée, on va leur épargner la participation aux tâches additionnelles en les attribuant à l'ensemble des noeuds dominants. Les noeuds ordinaires sont autorisés à devenir inactifs, du moment qu'ils ne sont pas impliqués dans une opération d'échange de données. Toutefois, l'activation du mode conservation d'énergie peut affecter considérablement certains critères de performance, à l'instar du débit, du délai de séjour ou de l'énergie consommée par paquet. Dans ce contexte, nous avons proposé une approche de conservation d'énergie basée sur une classification des paquets de données en termes de ressources consommées. Cette classification permet de privilégier les paquets de données en cours d'acheminement afin de réduire les délais d'attente au niveau des files d'attente des noeuds intermédiaires. Les performances de notre approche de conservation d'énergie ont été évaluées en terme d'énergie résiduelle et énergie dépensée par les noeuds par paquet de données, outre les mesures du nombre de paquets délivrés avec succès. Les résultats de cette évaluation montrent que notre approche est performante, en particulier dans le cas de réseaux denses à charge élevée.

Le dernier chapitre a introduit un nouvel algorithme de routage `LCR` qui exploite conjointement les apports d'une dorsale et les caractéristiques de notre mécanisme de clusterisation `TBCA`. En effet, ce dernier nous offre une organisation des noeuds en couches par rapport au noeud de référence DN. Cette organisation en couches nous permet de limiter encore plus le nombre de paquets de contrôle échangés lors des phases de découverte de routes et de leur maintenance. En effet, elle permet aussi à tout noeud d'avoir une idée sur la localisation géographique de la source et la destination par rapport à son emplacement actuel. De cette manière, il est apte à orienter la diffusion du paquet de contrôle à émettre en fonction de sa situation courante. Les simulations conduites confirment l'efficacité de notre algorithme de routage, même dans un environnement mobile.

Les travaux futures visent à mieux exploiter les caractéristiques de notre mécanisme de clusterisation pour améliorer la qualité de service dans un environement ad hoc.

Abréviations

ACK : ACKnowledgment

AF : Availability Factor

ANT : Active Neighbors Table

ATIM : Announcement Traffic Indication Message

BT : Busy Tone

candCH : Candidate for ClusterHead role

candGW : Candidate for GateWay role

Cbeacon : Clustering beacon

CBR : Constant Bit Rate

CH : ClusterHead

CPPCM : Cluster based Prioritized Power Conservation mechanism

CSATIM : Carrier Sense ATIM

CSMA/CA : Carrier Sense Multiple Access with Collision Avoidance

CTS : Clear To Send

CW : Contention Window

DCF : Distributed Coordination Function

DCPPCM : Dynamic Cluster based Prioritized Power Conservation mechanism

DCS-ATIM : Dynamic CS-ATIM

DIFS : Distributed InterFrame Space

DN : Designated Node

DPSM : Dynamic Power Saving Mechanism

GPS : Global Positioning System

GW : GateWay

IBSS : Independent Basic Service Set

LCR : Layered Cluster based Routing

M : Noeud Membre

MAC : Medium Access Control

MIS : Maximal Independent Set

MTCDS : Mac-layer Timer-based Connected Dominanting Set

MTIM : Multi hop ATIM

N : Rôle Non assigné

PDA : Personal Digital Assistant

PS : Power saving

PSM : Power Saving Mechanism

RTS : Request To Send

SIFS : Short InterFrame Space

TBCA : Tiered Based Clustering Algorithm

TBTT : Target Beacon Transmission Time

TECDS : Timer-based Energy aware Connected Dominanting Set

TSF : Timing Synchronization Function

UTA-PSM : Unicast Topology Aware Power Saving Mechanism

WCA : Weighted Clustering Algorithm

Bibliographie

[1] Abolhasan, M., Wysocki, T., Dutkiewicz, E., "A review of routing protocols for mobile ad hoc networks", Ad Hoc Networks, Vol. 2, No. 1, Elsevier, January 2004, pp. 1-22.

[2] Akkari, w., Belghith, A., "Enhancing power saving mechanisms for ad hoc networks", in IFIP International Federation for Information Processing, Volume 284, Wireless and Mobile Networking ;(Boston, Springer), pp 83-94, 2008.

[3] Akkari, w., Belghith, A., Ben Mnaouer, A., "Enhancing power saving mechanisms for ad hoc networks using neighbourhood information", Accepted to the International Wireless Communications and Mobile Computing Conference 2008 (IWCMC 2008), Crete, Greece, August 6-8, 2008.

[4] Akkari, w., Belghith, A., "Enhancing power saving mechanisms for ad hoc networks", Accepted to the 10th IFIP International Conference on Mobile and Wireless Communications Networks, MWCN'08, Toulouse, France, September 29-30, 2008.

[5] Akkari, w., Belghith, A., "Improving Neighborhood Aware Power Saving Mechanism using Carrier Sensing for Dynamic Advertising Windows", 1st IFIP WirelessDays International Conference 2008, Dubai, UAE, November 24-27, 2008.

[6] Alzoubi, K., Wan, P.-J., Frieder, O., "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks". Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing, June 09-11, 2002, Lausanne, Switzerland.

[7] Alzoubi, K. M., Wan, P.-J., Frieder, O., "Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks", in The 23rd International Conference on Distributed Computing Systems (IEEE, ICDCS), May 2003.

[8] Amis A. D., Prakash R., Vuong T. H.-P., Huynh D. T., "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks". Actes de Conference on Computer Communications IEEE INFOCOM 2000, no. 1, mars 2000, pp. 32-41.

[9] Banerjee, S., Misra, A., "Minimum Energy Paths for Reliable Communication in Multi-hop Wireless Networks", MOBIHOC'02, June 9-11, 2002, Lausanne, Switzerland.

[10] Basagni S., "Distributed Clustering for Ad Hoc Networks" , Actes de International Symposium on Parallel Architectures, Algorithms, and Networks I-SPAN'99, IEEE Computer Society, pp.310-315, 23-25 juin 1999, Australia.

[11] Basu, P., Khan, N., Little, T. D. C., "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," in Proc. IEEE ICDCSW'01, Apr. 2001, pp. 413-18.

[12] Benjie C., Jamieson H. K., Balakrishnan H., Morris R., "Span : An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", 7th ACM MOBICOM, July 2001, Rome, Italy.

[13] Belghith, A., Jemili, I., Mosbah, M., "A Distributed Clustering Algorithm without an Explicit Neighborhood Knowledge", International Journal of Computing and Information Sciences (IJCIS),Vol. 4, No. 1, ISSN 1708-0479, Avril 2007.

[14] Belghith, A., Akkari, W., Bonin, J.M., "Traffic Aware Power Saving Protocol in Multi-hop Mobile Ad-hoc networks", The Network Journal, Volume 2, Issue 4, August 2007.

[15] Belghith, A., Akkari, "Neighborhood Aware-Power Saving Mechanisms for ad hoc networks", The 33rd IEEE Conference on Local Computer Networks LCN'08, Montreal, Canada, October 14-17, 2008.

[16] Belghith, A., Akkari, "Power Saving Mechanisms for Ad hoc Networks Based on Handshaking Information Tapping", Accepted to The International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS), Leeds, United Kingdom, July 2-3, 2008.

[17] Belghith, A., Akkari, "Performance Evaluation of Power Save Protocols Using Carrier Sensing in Multihop Ad Hoc Networks ", The Seventh ACS/IEEE International Conference on Computer Systems and applications, AICCSA'09, Rabat, Morocco, May 10-13, 2009.

[18] Bergamo, P., Giovanardi, A., Travasoni ,A., Maniezzo, D., Mazzini, G., M Zorzi, M., " Distributed Power Control for Energy Efficient Routing in Ad Hoc Networks", Wireless Networks 10, 29-42, 2004.

[19] Blasi, D., Cacace, V., Casone, L. and Losacco. A., "Availability Clustering : A spine-based clustering scheme to exploit nodes heterogeneity in ad hoc networks". Proceedings of the Med-hoc Net 2003 Workshop, Mahdia, Tunisia, June 2003.

[20] Bo, H., Fu, H., Lin, L., Jia, W., "Efficient Construction of Connected Dominanting Set in Wireless Ad Hoc Networks", IEEE 2004.

[21] Camp, T., Boleng, J., Davies, V., "A Survey of Mobility Models for Ad Hoc Network Research", in Wireless Communication and Mobile Computing (WCMC) : Special issue on Mobile Ad Hoc Networking : Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.

[22] Chatterjee, M., Das, S. K., Turgut, D., "An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks," in Proceedings of the IEEE Globecom'2000, pp. 1697-701, 2000.

[23] Chen, B., Jamieson, K., Balakrishnan, H., Morris, R., "Span : an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks". ACM Wireless Networks, Vol. 8, No. 5, Septembre 2002, p.481-494.

[24] Chen, G., Nocetti, F.G., Gonzalez, J.S., Stojmenovic, I, "Connectivity Based k-hop Clustering in Wireless Networks", Proceedings of the 35th Annual Hawaii International Conference on System Sciences(HICSS'02), Vol. 7, p.188.3, January 07-10, 2002.

[25] Chen, Y. P., Liestman, A. L., "Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks", MobiHoc 2002, the Third ACM International Symposium on Mobile AdHoc Networking and Computing, pp. 165-172, Lausanne, Switzerland, June 2002.

[26] Chen, Y., P. and Liestman, A., L. : A Zonal Algorithm for Clustering Ad Hoc Networks, International Journal of Foundations of Computer Science, (2003), Vol. 14, No. 2, pp. 305-322.

[27] Chen, Y., P. and Liestman, A., L., Liu, J. : Clustering Algorithms For Ad hoc Wireless Networks, Nova Science Publishers, (2004).

[28] Chen, Y.P., Liestman, A.L., "Maintaining weakly-connected dominating sets for clustering ad hoc networks". Ad Hoc Networks, Vol. 3, No. 5, pp. 629-642, 2005 Elsevier B.V.

[29] Chiang, C.C., Gerla, M., "Routing and Multicast in Multihop, Mobile Wireless Networks", Proc. IEEE ICUPC'97, San Diego, CA, Oct. 1997.

[30] Chlamtac I., Conti M., Lui J.N., "Mobile Ad Hoc Networking : Imperatives and Challenges". Ad Hoc Network Journal, vol. 1, n. 1, janvier-février-mars 2003, éditions Elsevier. 2003.

[31] Dai, F., Wu, J., "Distributed dominant pruning in ad hoc wireless networks," in Proc. IEEE Int. Conf. on Communications (ICC 2003), Anchorage, AK2003, May 2003, vol. 1, pp. 353–357.

[32] Dow C.R., Lin J.H., Hwang S.F., Wang Y.W., "An Efficient Distributed Clustering Scheme for Ad-hoc Wireless Networks", Transactions Communications IEICE, vol. E85-R, no. 8, Hawaii, août 2002.

[33] Das, B., Sivakumar, R., Bhargavan, V., "Routing in Ad-Hoc Networks Using a Spine", Proceedings of the 6th International Conference on Computer Communications and Networks '97, Las Vegas, NV. September 1997, pp. 34.

[34] Dow C.R., Lin J.H., Hwang S.F., Wang Y.W., "An Efficient Distributed Clustering Scheme for Ad-hoc Wireless Networks", Transactions Communications IEICE, vol. E85-R, no. 8, Hawaii, août 2002.

[35] Er, I.I., Seah, W.K.G., "Mobility-based d-hop clustering algorithm for mobile ad hoc networks", Wireless Communications and Networking Conference (WCNC'04), March 2004.

[36] Er, I.I., Seah, W.K.G., "Performance analysis of mobility-based d-hop (MobDHop) clustering algorithm for mobile ad hoc networks", Computer Networks : The International Journal of Computer and Telecommunications Networking, Vol. 50, No. 17, pp. 3375-3399, 5 December 2006, Elsevier.

[37] Er, I.I, Seah, W.K.G., "Clustering overhead and convergence time analysis of the mobility-based multi-hop clustering algorithm for mobile ad hoc networks", Journal of Computer and System Sciences, Vol. 72, No. 7, p.1144-1155, November 2006, Elsevier.

[38] Gandhi, R., Parthasarathy, S., "Distributed algorithms for connected domination in wireless networks". In Journal of Parallel and Distributed Computing, Volume 67, Issue 7, July 2007, Pages 848-862.

[39] Gerla, M., Tsai, T. J.-C., Multicluster, Mobile, Multimedia Radio Network. Wireless Networks, 1 (1995), pp. 255-265.

[40] Ghosh, R., Basagni, S., "Limiting the Impact of Mobility on Ad Hoc Clustering", PE-WASUN'05, October 10-13, 2005, Montreal, Quebec, Canada.

[41] Gomez, J., Campbell, T., Naghshineh, M., Bisdikian, C., "A Distributed Contention Control Mechanism for Power Saving in Random-access Ad Hoc Wireless Local Area Networks", IEEE International Workshop on Mobile Multimedia Communications, pp. 114-123, 1999.

[42] Guha, S. and Khuller, S., "Approximation algorithms for connected dominating sets", Technical Report 3660, University of Maryland Institute for Advanced Computer Studies-Department of Computer Science, University of Maryland, College Park (June 1996).

[43] Guha, S. and Khuller, S., "Approximation algorithms for connected dominating sets", Algorithmica, 20 (1998),pp. 374-387.

[44] Guo, H., Ingelrest, F., Simplot-Ryl, D., Stojmenovic, I., "Performance Evaluation of Broadcasting Protocols for Ad Hoc and Sensor Networks". In Proc. 4th IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2005), Ile de Porquerolles, France, 2005.

[45] Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H., "Energy efficient communication protocols for wireless microsensor networks", in Proceedings of the Hawaaian International Conference on Systems Science (January 2000).

[46] Hong, X., Gerla, M., Yi, Y., Xu, K., Kwon, T.J., "Scalable Ad Hoc Routing in Large, Dense Wireless Networks Using Clustering and Landmarks", IEEE International Conference on Communications (ICC'2002),New York City, NY, April 2002.

[47] Hong, X., Xu, K., Gerla, M., "Scalable Routing Protocols for Mobile Ad Hoc Networks", IEEE Network magazine. Juillet/ Aout 2002. P-P 11-21.

[48] Hou, T., Li, V.O.K., " Transmission range control in multihop radio networks ", IEEE Transactions on Communications, COM-34, No. 1, pp. 38-44, January, 1986.

[49] Iwata, A., Chiang, C., Pei, G., Gerla, M., Chen, T., "Scalable Routing Strategies for Ad Hoc Wireless Networks", IEEE Journal on Selected Areas in Communcations, Vol. 17, No. 8, August 1999, pp. 1369-79.

[50] Jemili, I., Mosbah, M., Belghith, A., "Algorithme distribué à deux sauts pour la clusterisation dans les réseaux Ad Hoc". Colloque Francophone de l'Ingénierie des Protocoles de Communication, du 29 Mars au 1 Avril 2005. Bordeaux. France.

[51] Jemili, I., Belghith, A., Mosbah, M., "Clusterisation pour le routage dans les réseaux ad hoc". Journées scientifiques des jeunes chercheurs en Génie Electrique et Informatique (GEI'06). Mars 2006. Hammamet. Tunisie.

[52] Jemili, I., Belghith, A., Mosbah, M., "Algorithme Distribué de Clusterisation sans connaissance du voisinage : principe et évaluation". NOTERE'2007, Juin 2007. Marrakech, Maroc, pp. 253-260.

[53] Jemili, I., Belghith, A., Mosbah, M., "Evaluating the efficiency of clustering on routing and network performance", ICWN /Worldcomp'08, 14th to 17 July, Las vegas, USA.

[54] Jemili, I., Belghith, A., Mosbah, M., "A Synchronous Tiered Based Clustering Algorithm for large-scale Ad hoc Networks", in IFIP International Federation for Information Processing, Vol. 284, Wireless and Mobile Networking, Boston, Springer, pp. 41-55.

[55] Jemili, I., Belghith, A., Mosbah, M., "Cluster based Prioritized Power Conservation mechanism", IFIP Wireless days 2008, November 2008, Dubai.

[56] Jemili, I., Belghith, A., Mosbah, M., "Exploiting a Clustering Mechanism for Power Saving in Ad hoc Networks : Performance Evaluation", Accepted in AICCSA 2009, Maroc.

[57] Jia, L., Rajaraman, R., Suel, R., "An Efficient Distributed Algorithm for Constructing Small Dominating Sets". In Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC), pages 33-42, 2001.

[58] Jiang, J.R., Tseng, Y.C., Hsu, C.S., Lai, T.H., "Quorum-based asynchronous power-saving protocols for 802.11 ad hoc networks", Proceedings of the International Conference on Parallel Processing, pp. 812-817, Nice, France, April 22-26, 2003.

[59] Jiang, J.R., Tseng, Y.C., Hsu, C.S., Lai, T.H., "Quorum-based asynchronous power-saving protocols for 802.11 ad hoc networks", ACM mobile networking and applications (MONET), special issue on algorithmic solutions for wireless, mobile, ad hoc and sensor networks, vol. 10, No.1/2, pp. 169-181, 2005.

[60] Jones, C. E., Sivalingam, K.M., Agrawal, P., Chen, J. C., "A survey of energy efficient network protocols for wireless networks", Wireless Networks 7, pp. 343-358, 2001.

[61] J-Sim simulator : on-line at www.j-sim.org, last visited in 20-10-2008.

[62] Jung, E. S., Vaidya, N. H., "An energy efficient MAC protocol for wireless LANs" inProc. IEEE INFOCOM, June(23-27) 2002, vol. 3, pp. 1756-1764, New York, USA.

[63] Kim, B., Yang, J., Zhou, D., Sun, M., " Energy-Aware Connected Dominating Set Construction in Mobile Ad hoc Networks", Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN 2005), 17 to 19 October 2005, pp. 229-234.

[64] Kim, I.L., Kim, Y.S., Kim, K.C., " Zone-Based Clustering for Intrusion Detection Architecture in Ad-Hoc Networks". APNOMS 2006. pp. 253-262.

[65] Kuhn, F., Wattenhofer, R., "Constant-Time Distributed Dominating Set Approximation". Proceedings of the twenty-second annual symposium on Principles of distributed computing, p.25-32, July 13-16, 2003, Boston, Massachusetts.

[66] Kuhn, F., Moscibroda, T., Wattenhofer, R., "Initializing newly deployed ad hoc and sensor networks", Proceedings of the 10th annual international conference on Mobile computing and networking (MOBICOM), pp. 260-274, September 26-October 01, 2004, Philadelphia, PA, USA.

[67] LAN MAN Standards Committee of the IEEE Computer Society, "IEEE Std 802.11-1999, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," IEEE, 1999.

[68] Lin, C.R., Gerla, M., "Adaptive clustering for mobile wireless networks", IEEE Journal on Selected Areas in Communication, vol. 15, 7, September 1997, pp. 1265-1275.

[69] Li, Y., Peng, S., Chu, W., " An Efficient Algorithm for Finding an Almost Connected Dominating Set of Small Size on Wireless Ad Hoc Networks ", IEEE 2006.

[70] Miller, M. J., Vaidya, N. H., "Improving Power Save Protocols Using Carrier Sensing and Dynamic Advertisement Window," Technical Report, University of Illinois at Urbana-Champaign, Dec. 2004.

[71] Minoux, M., Bartnik, G., "Graphes algorithmes logiciels". Edition DUNOD informatique. Novembre 1986.

[72] Nair, R. S., "JSIM : A Java-Based Query Driven Simulation and Animation Environment", Masters Thesis (M.S. in CS Degree), 1997.

[73] Pei G., Gerla M., Hong X., Chiang C.C., "A Wireless Hierarchical Routing Protocol with Group Mobility", IEEE Wireless Communications and Networking Conference WCNC'99, N. 1, New Orleans, LA, Septembre 1999, pp. 1538-1542.

[74] Pearlman, M. R., Haas, Z. J., "Determining the Optimal Configuration for the Zone Routing Protocol," IEEE JSAC, vol. 17, Aug. 1999, pp. 1395-414.

[75] Perkins, C. E., Belding-Royer, E. M., Das, S. R., "Ad hoc On-Demand Distance Vector (AODV) Routing". Internet draft. IETF MANET Working Group. Février 2003.

[76] Ramanathan, R., Rosales-Hain, R., "Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment". In Proc. of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2000.

[77] Rieck, M.Q., Pai, S., Dhar, S., "Distributed routing algorithms for wireless ad hoc networks using d-hop connected dominating set", International Conference of High Performance Computing in Asia Pacific Region, December 2002.

[78] Singh, S., Raghavendra, C., "PAMAS : Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks". ACM SIGCOMM Computer Communication Review, vol. 28, no. 3, pp. 5-26, Juillet 1998.

[79] Singh, S., Woo, M., Raghavendra, C., "Power-aware Routing in Mobile Ad Hoc Networks", Proceedings of the 4th annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 181-190, October 25-30, 1998, Dallas, Texas, United States.

[80] Sivakumar, R., Das, B., Bharghavan, V., "An improved spine-based infrastructure for routing in ad hoc networks", Proceedings of the International Symposium on Computers and Communications (ISCC'98), Athens, Greece. June 1998.

[81] Stojmenovic, I., Lin, X., "Power-aware Localized Routing in Wireless Networks", IEEE International Parallel and Distributed Processing Symposium, pp. 371-376, 2000.

[82] Stojmenovic, I., Seddigh, M., Zunic, J., "Dominating sets and neighbor elimination-based broadcastings in wireless networks". IEEE Transactions on Parallel and Distributed Systems, 13(1) :14-25, January 2001.

[83] Stojmenovic, I., Seddigh, M., Zunic, J., "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks", IEEE TransactionsonParallel and Distributed Systems, Vol. 13, N. 1, January 2002, p.14-25.

[84] Sucec, J., Marsic, I., "Clustering Overhead for Hierarchical Routing in Mobile Ad hoc Networks", IEEE INFOCOM 2002, pp. 1698-1706.

[85] Suh, C., Ko, Y.-B., Kim, J.-H., "Enhanced Power Saving for IEEE 802.11 WLAN with Dynamic Slot Allocation". Actes de International Conference on Mobile Ad-hoc and Sensor Networks (MSN'05), pp. 498-507, Dec. 2005.

[86] Takagi, H., Kleinrock, L., "Optimal transmission ranges for randomly distributed packet radio terminals", IEEE Transactions on Communications, COM-32, No. 3, March 1984.

[87] Theoleyre, F., Valois, F., "A self-organization structure for hybrid networks", Ad Hoc Networks. (2007), doi :10.1016/j.adhoc.2007.02.013.

[88] Tseng, Y.-C., Hsu, C.-S., Hsieh, T.-Y., " Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks". Journal of Computer Networks, Vol. 43, No. 3, pp. 317-337, Elsevier 2003.

[89] Wan, P.J., Alzoubi, K.M., Frieder, O., "Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks", IEEE 2002.

[90] Wattenhofer, R., Li, L., Bahl, P., Wang, Y.-M., "Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks". In Proceedings of the 20 th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2001.

[91] Wattenhofer, R., Zollinger, A., "XTC : A Practical Topology Control Algorithm for Ad-Hoc Networks". Technical report 407, Department of computer science, ETH Zurich, 2003.

[92] Lou, W., Wu, J., "A K-hop Zone-Based Broadcast Protocol in Mobile Ad Hoc Networks". Globecom 2004. IEEE Communications Society, pp. 1665-1669.

[93] West, D., "Introduction to Graph Theory". Second edition, Prentice Hall, Upper Saddle River, N.J., 2001.

[94] Wu, J., Li, H., "On Calculating Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks", Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DialM'99), pp. 7-14, August 1999, Seattle, Washington, United States.

[95] Wu, J., Dai, F., Gao, M., Stojmenovic, I., "On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks," Journal of Communications and Networks, Vol. 4, No. 1, Mar. 2002, pp. 59-70.

[96] Wu, J., "Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 9, pp. 866-881, Sep. 2002.

[97] Xu, Y., Heidemann, J., Estrin, D., "Geography-Informed Energy Conservation for Ad Hoc Routing", in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'2001), pp. 70-84, Rome, Italy, July 2001.

[98] Xu, Y., Heidemann, J., Estrin, D., "Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks," Tech. Rep. 527, USC/Information Sciences Institute, 2000.

[99] Ye, W., Heidemann, J., Estrin, D.,. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In IEEE Infocom 2002, June 2002.

[100] Yu, J. Y., Chong, J., " A Survey of Clustering Schemes for Mobile Ad Hoc Networks", IEEE Communications Surveys and Tutorials, Vol. 7, N. 1, 2005.

[101] Yu, J. Y., Han, P., Chong, J., "An efficient clustering scheme for large and dense mobile ad hoc networks (MANETs)". Journal of Computer Communications, Vol. 30, No. 1, pp. 5-16, 2006 Elsevier B.V. doi :10.1016/j.comcom.2006.07.014

[102] Zheng, R., Hou, J., Sha, L., "Asynchronous Wakeup for Ad Hoc Networks", MobiHoc 2003, June 1-3, 2003, Annapolis, Maryland, USA.

[103] Zhou, D., Sun, M., Lai, T., " A Timer-based Protocol for Connected Dominating Set Construction in IEEE 802.11 Multihop Mobile Ad hoc Networks", SAINT2005 Workshops, Trento, Italy, January 31- February 4, 2005.

[104] Zou, S., Wu, H., Cheng, S., " Adaptative Power Saving Mechanisms for DCF in IEEE 802.11 ", Mobile Networks and Applications, Vol. 10, No. 5, pp. 763-770, Kluwer Academic Publishers 2005.