algorithmes de clustérisation et routage dans les réseaux Ad Hoc

Badreddine Guizani

Numéro d'ordre : **176** Année : 2012

THÈSE

Université de Technologie de Belfort-Montbéliard
École Doctorale Sciences pour l'Ingénieur et Microtechniques

École Nationale des Sciences de l'Informatique (Tunisie)

pour obtenir le grade de
DOCTEUR

DISCIPLINE : **INFORMATIQUE**

ALGORITHME DE CLUSTERISATION ET PROTOCOLES DE ROUTAGE DANS LES RÉSEAUX AD HOC

Présentée par
Badreddine GUIZANI
Laboratoire Systèmes et Transport

Soutenue le 04 Avril 2012 devant le Jury composé de :

Président de jury:

Monsieur **Kokou YETONGNON**, Professeur des Universités, Université de Bourgogne

Rapporteurs :

Monsieur **Aref MEDDEB**, Maître de Conférences HDR , ISITC, Hammam-Sousse (Tunisie)
Monsieur **Cyrille BERTELLE**, Professeur des Universités, Université du Havre
Monsieur **Djamal BENSLIMANE**, Professeur des Universités, Université Claude Bernard, Lyon I

Examinateurs :

Monsieur **Olivier SIMONIN**, Maître de Conférences HDR, Université Henri Poincaré, Nancy I
Monsieur **Pablo GRUER**, Professeur des Universités, UTBM

Directeurs de Thèse :

Monsieur **Abderrafiaa KOUKAM**, Professeur des Universités, UTBM
Monsieur **Béchir el AYEB**, Professeur des Universités, Faculté des Sciences Monastir (Tunisie)

Sommaire

1 Introduction générale **13**
1.1 Contexte et problématique 13
1.2 Première analyse et objectifs de ces travaux 14
1.2.1 Clusterisation à structure stable 15
1.2.2 Routage à état de liens scalable 16
1.3 Plan de la thèse 16

I Clusterisation dans les réseaux mobiles sans-fil **19**

2 État de l'art des algorithmes de clusterisation **21**
2.1 Introduction 22
2.2 Intérêts et coûts de la clusterisation 22
2.3 Classification des algorithmes de clusterisation 24
2.3.1 Rayon des clusters 26
2.3.2 Métrique de sélection des cluster-heads 26
2.3.3 Structure de l'ensemble des cluster-heads 27
2.3.4 Autres critères de classification 29
2.4 Métriques d'évaluation des performances des algorithmes de clusterisation . 30
2.5 Quelques approches de clusterisation 31
2.5.1 Les algorithmes de clusterisation à 1 saut 31
2.5.2 Les algorithmes de clusterisation à k sauts 38
2.6 Conclusion 43

3 Proposition d'un algorithme de clusterisation à structure stable α-SSCA **45**
3.1 Introduction 46
3.2 Description de l'algorithme α-SSCA 46
3.2.1 Détection du voisinage 47
3.2.2 Formation des clusters 48
3.2.3 Maintenance des clusters 50
3.3 Analyse expérimentale 54
3.3.1 Choix de la métrique de sélection des cluster-heads 55
3.3.2 Variation du nombre de nœuds 56

3.3.3 Variation de la densité des nœuds 58
3.4 Impact de la stabilité sur le routage 60
3.5 Conclusion 63

4 Proposition d'un algorithme de clusterisation générique à K-sauts : SKCA 65
4.1 Introduction 66
4.2 Présentation générale de l'algorithme SKCA 66
4.3 Les métriques utilisées par α-SKCA 68
4.4 La formation des clusters 68
4.4.1 Phase 1 : Premier tour d'élection des cluster-heads 70
4.4.2 Phase 2 : Deuxième tour d'élection des cluster-heads 71
4.5 La maintenance des clusters 73
4.6 Algorithmes couverts par SKCA 74
4.7 Analyse expérimentale 75
4.7.1 Impact du rayon de la zone de présélection (r) 76
4.7.2 Impact du rayon des clusters (K) 78
4.7.3 Impact de la taille la zone d'extension (ϵ) 79
4.7.4 Impact du seuil du facteur de stabilité (α) 81
4.7.5 Impact de la taille de la zone critique (ρ) 82
4.8 Conclusion 83

II Le routage dans les réseaux mobiles sans-fil 85

5 État de l'art des protocoles de routage 87
5.1 Introduction 89
5.2 Classification des protocoles de routage unicast 89
5.2.1 Structuration du réseau 90
5.2.2 Découverte de routes 91
5.2.3 Calcul des routes 91
5.2.4 Alternance des routes 92
5.2.5 Stockage des données 92
5.2.6 Circulation des informations de contrôle 93
5.2.7 Acheminement des données 93
5.2.8 Sélection des routes 94
5.2.9 Maintenance des routes 94
5.3 Protocoles de routage réactifs 95
5.3.1 Le protocole AODV 95
5.3.2 Le protocole DSR 97
5.3.3 Le protocole DYMO 99

5.3.4 Autres protocoles réactifs 100
5.4 Protocoles de routage proactifs 101
5.4.1 Le protocole DSDVR 101
5.4.2 Le protocole OLSR 101
5.4.3 Autres protocoles proactifs 103
5.5 Protocoles de routage hybrides 104
5.5.1 Le protocole CBRP 104
5.5.2 Le protocole ZRP 106
5.6 Scalabilité du routage à état de liens 107
5.6.1 Approche plate : le protocole F-OLSR 107
5.6.2 Approches basées sur une structure des clusters 107
5.6.3 Approches hiérarchiques 113
5.7 Conclusion 115

6 Proposition d'un protocole de routage à état de liens des clusters : CLSR 117
6.1 Introduction 118
6.2 La découverte du voisinage 119
6.3 La structuration du réseau 119
6.3.1 Construction de la structure des clusters 120
6.3.2 Construction de la dorsale virtuelle 120
6.4 La gestion de la topologie 122
6.5 Calcul de la table de routage 124
6.5.1 Routage au voisinage 124
6.5.2 Routage inter-cluster 127
6.6 Acheminement des paquets 129
6.7 Analyse expérimentale 129
6.7.1 Paramètres de simulation 130
6.7.2 Métriques d'évaluation 131
6.7.3 Trafic de contrôle *(Overhead)* 131
6.7.4 Ratio des paquets délivrés 133
6.7.5 Délai de bout-en-bout 135
6.7.6 Nombre moyen de sauts 137
6.8 Conclusion 139

7 Proposition d'un protocole de routage hiérarchique : HCLSR 141
7.1 Introduction 142
7.2 Structuration du réseau 142
7.2.1 Structuration du premier niveau 142
7.2.2 Clusterisation hiérarchique 142
7.3 Le routage dans HCLSR 144

7.3.1 La gestion de la topologie 144
7.3.2 Calcul des routes 146
7.3.3 Acheminement des données 146
7.4 Analyse théorique 148
7.4.1 Borne supérieure du protocole F-OLSR 148
7.4.2 Borne supérieure du protocole SA-OLSR 149
7.4.3 Borne supérieure du protocole L-HCLSR 149
7.4.4 Comparaison des bornes supérieures 150
7.5 Analyse expérimentale 153
7.6 Conclusion 157

8 Conclusion **159**
8.1 Bilan et apports 159
8.1.1 Proposition d'un algorithme de clusterisation à structure stable . . . 159
8.1.2 Proposition d'un algorithme de clusterisation générique à K-sauts . 160
8.1.3 Proposition d'un protocole de routage à état de liens des clusters . . 161
8.1.4 Proposition d'un protocole de routage hiérarchique 161
8.2 Perspectives 162

III Bibliographie **165**

Sommaire des Figures

1.1 Organisation des contributions dans la thèse. 17

2.1 Exemple d'une structure des clusters. 23
2.2 Classification des algorithmes de clusterisation. 25
2.3 Exemple de structures de l'ensemble des cluster-heads 29

3.1 Un exemple de topologie du réseau. 48
3.2 Diagramme de transition d'état de l'algorithme SSCA. 48
3.3 Exemples de jonction à la structure des clusters. 50
3.4 Exemple de maintenance de la structure des clusters en cas d'apparition d'un nouveau lien 51
3.5 Exemples de maintenance de la structure des clusters lors de la perte de lien 53
3.6 Le nombre moyen de clusters formés en fonction des métriques. 56
3.7 Résultats de simulation de l'expérimentation 1. 57
3.8 Résultats de simulation de l'expérimentation 2. 59
3.9 Ratio des paquets délivrés. 61
3.10 Trafic de contrôle total (nombre de paquets). 62
3.11 Délai moyen de bout-en-bout. 63

4.1 Exemple des zones prédéfinies autour des cluster-heads. 68
4.2 Diagramme de transitions d'état. 69
4.3 Exemple de formation des clusters ($K = 3$, $r = 1$, $\epsilon = 1$). 72
4.4 Résultats de simulation en fonction du paramètre r. 76
4.5 Trafic de contrôle en fonction du paramètre r. 77
4.6 Trafic de contrôle en fonction du paramètre K. 77
4.7 Résultats de simulation en fonction du paramètre K. 78
4.8 Résultats de simulation en fonction du paramètre ϵ. 80
4.9 Trafic de contrôle en fonction du paramètre ϵ. 81
4.10 Résultats de simulation en fonction du paramètre α. 82
4.11 Résultats de simulation en fonction du paramètre ρ. 83

5.1 Taxonomie des protocoles de routage unicast. 90
5.2 Exemple d'une structure des clusters créé par OLSR-Trees. 108
5.3 Le partitionnement du réseau en clusters dans HSR [Iwata et al., 1999]. . . 114

6.1 Exemple de la Structure du réseau dans CLSR. 121
6.2 Exemple de calcul des routes au voisinage du nœud 1 127
6.3 Exemple de la vue de la topologie du réseau (par le nœud 5) 128
6.4 Trafic de contrôle du routage en fonction du nombre de nœuds. 132
6.5 Trafic de contrôle du routage en fonction de la densité des nœuds. 132
6.6 Taux de livraison des paquets en fonction du nombre de nœuds. 134
6.7 Taux de livraison des paquets en fonction de la densité des nœuds. 134
6.8 Délai de bout en bout en fonction du nombre de nœuds. 136
6.9 Délai de bout en bout en fonction de la densité des nœuds. 136
6.10 Nombre moyen de retransmissions des paquets en fonction du nombre de nœuds. 138
6.11 Nombre moyen de retransmissions des paquets en fonction de la densité des nœuds. 138

7.1 Exemple de structuration du premier niveau. 144
7.2 Exemple d'élection de cluster-heads de niveau 2. 144
7.3 Exemple de la connaissance de topologie au niveau du nœud (1). 146
7.4 Exemple de routes de données pour les protocoles HSR, HOLSR et L-HCLSR. 147
7.5 Overhead généré par L-HCLSR, cas des réseaux larges 151
7.6 CTC généré par L-HCLSR, cas des réseaux très larges. 151
7.7 Comparaison du trafic de contrôle généré par TC-CTC de F-OLSR, SA-OLSR et L-HCLSR pour (L=2 et L=3), densité=30 152
7.8 Comparaison du trafic de contrôle généré par TC-CTC de F-OLSR, SA-OLSR et L-HCLSR pour (L=2 et L=3), Nombre de nœuds=5000 153
7.9 Comparaison des bornes supérieures théoriques et les résultats de simulation du trafic de contrôle du routage en fonction du nombre de nœuds. . . . 154
7.10 Trafic de contrôle du routage en fonction du nombre de nœuds. 155
7.11 Taux de livraison des paquets en fonction du nombre de nœuds. 155
7.12 Nombre moyen de retransmissions des paquets en fonction du nombre de nœuds. 156
7.13 Délai de bout en bout en fonction du nombre de nœuds. 157

Sommaire des Définitions

2.1 Ensemble dominant (DS) 27
2.2 Ensemble dominant connecté (CDS) 28
2.3 Ensemble dominant faiblement connecté (WDS) 28
2.4 Ensemble indépendant (IS) 28
2.5 Ensemble indépendant maximal (MIS) 28
3.1 Facteur de stabilité 51
3.2 Condition α-abandon 51
6.1 Cluster-heads voisins 123

Sommaire des Règles

3.1 Création d'un nouveau cluster 49
3.2 Affiliation à un cluster 49
3.3 Attente de priorité 49
3.4 Création contrainte d'un nouveau cluster. 49
3.5 Rapprochement de deux cluster-heads 52
3.6 Perte de lien 53
4.1 Atteinte d'éligibilité locale 70
4.2 Abandon de la concurrence 70
4.3 Attente de priorité 70
4.4 Déblocage de l'attente 71
4.5 Création d'un nouveau cluster 71
4.6 Affiliation à un cluster 71
4.7 Adhésion à un cluster en mode invité 71
4.8 Création contrainte d'un nouveau cluster 72
4.9 Préparation d'une création contrainte d'un nouveau cluster 72
4.10 Perte de lien : nœud membre 73
4.11 Détection d'un nouveau lien 73
4.12 Perte de lien : nœud membre-invité 74
4.13 Rapprochement de deux cluster-heads : zone interdite 74
4.14 Rapprochement de deux cluster-heads : zone critique 74

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 Contexte et problématique

Grâce à l'évolution des technologies de communication sans fil, l'utilisation des systèmes d'information a changé et a exprimé notamment des besoins de mobilité et d'autonomie chez les utilisateurs. Les réseaux filaires ne pouvant plus assurer une telle possibilité d'utilisation, les réseaux auto-organisables sans fil ont permis de satisfaire en partie ces nouveaux besoins. Les utilisateurs des réseaux auto-organisables souhaitent aussi avoir les mêmes services que ceux offerts par les réseaux filaires. Ainsi, les applications utilisées dans les réseaux filaires doivent rester fonctionnelles sur les réseaux auto-organisables. En particulier, les applications faisant intervenir deux utilisateurs (courrier électronique, téléchargement de fichiers, etc.) ou les applications de groupe d'utilisateurs (télé-séminaire, téléconférence, jeux interactifs, etc.).

Les réseaux auto-organisables fonctionnent de façon autonome, sans configuration, sans intervention et sans avoir besoin d'une infrastructure fixe. Ils réunissent plusieurs types de réseaux tels que les réseaux ad hoc (MANET), les réseaux de capteurs (Sensor Networks) et les réseaux véhiculaires (VANET). De tels réseaux se composent de nœuds mobiles ou non qui peuvent communiquer directement entre eux s'ils sont situés à portée radio. Comme la portée des nœuds est relativement limitée, le déploiement d'un réseau à grande échelle nécessite que le réseau soit multi-sauts, c'est-à dire que des nœuds intermédiaires jouent le rôle de relais. Avec l'état imprévisible des liens et les changements continus de la topologie, les réseaux auto-organisables soulèvent de nombreuses questions notamment concernant le routage dynamique. Dans un réseau auto-organisable, chaque nœud est tenu à jouer le rôle de relais et d'être hôte et routeur à la fois afin d'assurer l'acheminement du trafic transitant dans le réseau. De ce fait, chaque nœud intermédiaire doit participer à la phase de découverte nécessaire pour l'établissement du chemin, au transfert des paquets échangés et à la maintenance de ce chemin. Or, l'absence d'une infrastructure fixe et d'une administration centralisée rendent difficile la tâche de maintenance. En plus, vu la limitation des ressources dans les réseaux auto-organisables, en termes de bande passante et d'énergie, la construction des routes et leur maintenance doivent se faire avec un minimum de trafic de contrôle.

L'utilité des réseaux auto-organisables 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 importante lorsque le nombre de participants augmente et le réseau doit passer à l'échelle. Or, adopter une structure plate du réseau et attribuer des rôles égalitaires à tous les nœuds présentent des limitations au passage à l'échelle, vu la croissance du trafic de contrôle avec le nombre de nœuds [Pearlman and Haas, 1999]. Nous focalisons nos travaux sur la fonction de routage et nous nous intéressons à améliorer les performances du réseau, en réduisant le trafic de contrôle, dans le but de faire face aux problèmes de passage à l'échelle. Le protocole de routage utilise des informations de contrôle pour établir les routes et les maintenir. Ces informations de contrôle sont échangées entre tous les nœuds du réseau. Plus le nombre de nœuds du réseau augmente, plus le trafic de contrôle augmente et peut par la suite gêner le trafic de données. En outre, avec les changements continus de la topologie, ce trafic de contrôle se multiplie pour mettre à jour les routes en fonction de l'état des liens entre les nœuds. Ainsi, le routage dynamique dans les réseaux auto-organisables soulève de nombreuses questions notamment à large échelle. De ce fait, nous avons orienté nos travaux sur la technique de clusterisation, qui est largement déployée dans les réseaux auto-organisables, pour accroître les performances du réseau et faire face aux problèmes de passage à l'échelle. En effet, la clusterisation permet de structurer le réseau et de distinguer des nœuds particuliers qui vont assumer des fonctionnalités essentiellement dans le routage. Ainsi, la génération et la diffusion des informations de routage peuvent se limiter à cet ensemble de nœuds [Kozat and al., 2001, Pearlman and Haas, 1999]. Ceci évite les diffusions fréquentes de ces informations pouvant surcharger le réseau et dégrader considérablement les performances du réseau. C'est ainsi que plusieurs mécanismes de clusterisation ont vu le jour dans le but d'organiser un réseau auto-organisable de grande envergure. Dans ce contexte, nous nous sommes intéressés à l'élaboration d'une fonction de routage efficace dans les réseaux auto-organisables à grande échelle. Cette fonction de routage repose sur la technique de clusterisation pour tirer profit de ses avantages surtout dans sa capacité de passage à l'échelle.

1.2 Première analyse et objectifs de ces travaux

Cette thèse se situe dans le cadre générale de l'auto-organisation et de la gestion des réseaux mobiles sans fil. Elle traite le problème de passage à l'échelle de la fonction de routage dans ce type de réseaux. Elle reprend l'idée d'introduire une hiérarchie dans le réseau afin de réduire le trafic de contrôle relatif au processus de routage. Dans ce contexte, nous nous intéressons aux algorithmes de clusterisation dans le but de satisfaire les besoins

liés au problème de routage.

Dans ce qui suit, nous présentons en quelque mots l'objectif principale de cette thèse :

> Proposer une approche de routage à état de liens scalable basé sur une auto-organisation en un ensemble de clusters pour les réseaux sans fil mobiles.

Cet objectif global peut se décomposer en deux sous-objectifs. Le premier consiste à concevoir un algorithme de clusterisation qui construit et maintient une structure stable des clusters sans engendrer un trafic de contrôle important. En effet, la clusterisation est une technique qui permet de structurer et organiser les réseaux larges. La structuration du réseau rend plus aisé son exploitation et permet de supporter plus efficacement les protocoles de haut niveau notamment le routage. Le deuxième sous-objectif consiste à tirer profit de la clusterisation pour concevoir un protocole de routage capable d'assurer le passage à l'échelle.

1.2.1 Clusterisation à structure stable

Dans ce travail, nous nous intéressons à la technique de clusterisation qui a été couramment utilisée dans les réseaux mobiles sans fil à large échelle. Nous proposons un algorithme de clusterisation qui va être intégrer plus tard dans la fonction de routage pour faire face au problème de passage à l'échelle.

De ce fait, le choix de cet algorithme demeure très important puisqu'il doit tenir en compte les enjeux de la fonction de routage. En particulier, nous proposons un algorithme qui crée et maintient une structure stable des clusters pour prolonger la validité des routes. En plus, nous concevons un algorithme à 1-saut afin de réduire les messages explicites de clusterisation. En effet, les messages de découverte de voisinage HELLO propres à la fonction de routage sont utilisés dans les phases de la formation et de la maintenance des clusters.

Pour une meilleure scalabilité, l'algorithme de clusterisation doit aussi être en mesure de réduire le nombre de clusters générés dans le réseau. Deux techniques sont possibles pour atteindre cette fin : il s'agit de la clusterisation à K-sauts et la clusterisation hiérarchique. Dans ce travail, nous proposons une généralisation à K-sauts de notre première proposition. L'enjeu de cette approche est d'élaborer une solution générique admettant plusieurs paramètres. Ces derniers sont utilisés pour modeler le comportement de l'algorithme selon les besoins. Nous explorons également l'approche hiérarchique de clusterisation pour réduire le nombre de clusters en regroupant les clusters voisins dans un même cluster au niveau supérieur.

1.2.2 Routage à état de liens scalable

Nous proposons un protocole de routage pour les réseaux mobiles sans fil à large échelle. La stratégie de routage choisie est le routage à état de liens. Ce choix se justifie par le succès de ce type de routage dans le cas des réseaux filaires. En plus, il possède plusieurs avantages, notamment, sa robustesse, sa convergence bornée, son court délai et sa flexible capacité de supporter les protocoles de routage à qualité de services. Le recours à ce type de routage s'est avéré particulièrement intéressant pour le cas des réseaux mobiles sans fil. C'est ainsi que des protocoles à état de liens ont été proposés et ont été largement étudiés tel que le protocole OLSR [Clausen and Jacquet, 2003].

Pour ces raisons, nous envisageons de concevoir un protocole de routage à état de liens pour les réseaux mobiles sans fil. Notre objectif est alors d'adapter cette stratégie de routage pour la rendre scalable à large échelle. En particulier, dans notre approche, nous appliquons le routage à états de liens sur une topologie de réseau structurée en clusters. Grâce aux apports indéniables de la clusterisation, le trafic de contrôle inhérent de la stratégie à état de liens peut être réduit considérablement. En effet, réduire le trafic de contrôle représente une issue clé pour le problème de scalabilité dans les réseaux larges.

Le principe général de notre approche est de déterminer les chemins vers les destinations se trouvant à l'intérieur ou à l'extérieur du cluster moyennant des échanges de messages d'état des liens entre les nœuds du réseau. Le but de notre approche est d'optimiser la diffusion de ces messages topologies tout en permettant à chaque nœud du réseau d'avoir les informations nécessaires pour atteindre toute destination dans le réseau. Dans ce contexte, nous limitons la diffusion de ces messages de contrôle à un ensemble particulier de nœuds. Cet ensemble est construit à la base de la topologie des clusters. En plus, nous choisissons un mécanisme de clusterisation qui génère une structure stable des clusters et qui n'introduit pas un trafic supplémentaire pour la formation et la maintenance des clusters. Nous exploitons également la clusterisation hiérarchique dans l'objectif de réduire encore plus le trafic de contrôle. Ainsi, nous proposons une version hiérarchique du protocole à routage. En plus de la réduction spatiale de la diffusion des messages de contrôle de la topologie, cette version intègre une réduction temporelle de l'envoi de ces messages spécialement pour les destination les plus éloignés.

1.3 Plan de la thèse

Après cette brève description des objectifs de la thèse, cette section présente le plan de la thèse. La figure 1.1 souligne les chapitres qui caractérisent notre contribution.

Ce mémoire de thèse est organisé en 8 chapitres. Cette introduction a permis d'illustrer la problématique du routage dans les réseaux auto-organisables à large échelle et d'expliquer les motivations qui ont poussé à la conception de nouvelles solutions.

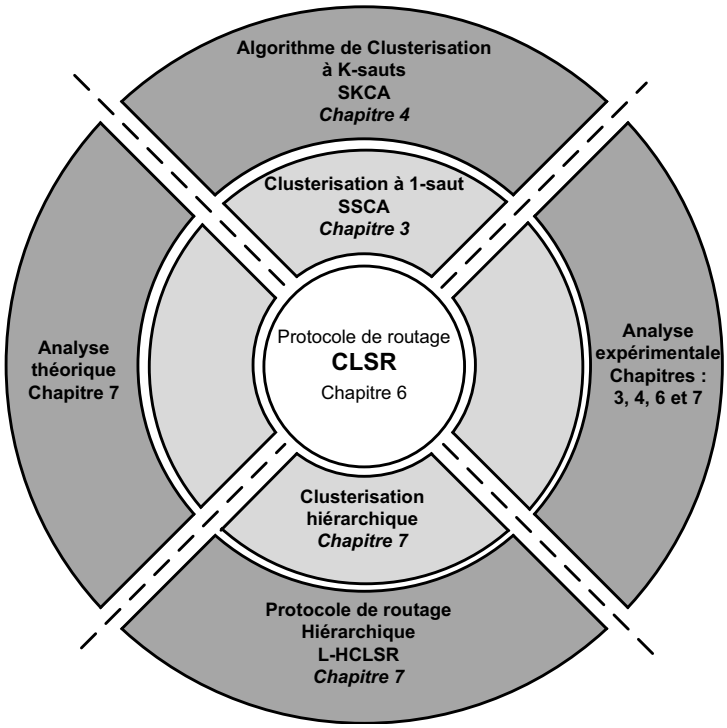


Figure 1.1 – Organisation des contributions dans la thèse.

La technique de clusterisation, largement connue et utilisée dans les réseaux auto-organisables, est présentée dans le chapitre 2. Nous énumérons les intérêts de cette technique et les coûts qui sont introduits par son déploiement. Nous présentons également les critères pertinents des algorithmes existants de clusterisation dans l'objectif d'élaborer une classification de ces algorithmes. Une étude détaillée de quelques approches proposées dans la littérature est aussi élaborée.

Le chapitre 3 présente notre proposition d'un algorithme de clusterisation, nommé α-SSCA, qui a pour objectif de générer une structure stable des clusters. Cet algorithme est planifié d'être introduit dans un protocole de routage afin d'améliorer ses performances à large échelle. Une étude des performances de notre algorithme est élaborée par simulation.

Le chapitre 4 propose un autre algorithme de clusterisation qui est une généralisation à K-sauts du précédent algorithme. Ce nouvel algorithme générique s'inspire aussi de l'algorithme connu 3hBAC[Yu and Chong, 2003]. La contribution principale est d'améliorer la stabilité de la structure des clusters quand la topologie du réseau change fréquemment tout en réduisant le nombre de clusters dans les réseaux.

Une étude approfondie des protocoles de routage est réalisée dans le chapitre 5. Cette étude a permis de proposer des critères de classification dans le but d'élaborer une nouvelle

taxonomie. Nous exposons en détails plusieurs protocoles de routage existants. Une attention particulière est portée aux protocoles à état des liens basés sur la clusterisation.

Dans le chapitre 6, nous proposons un nouveau protocole de routage, appelé CLSR, basé sur la clusterisation. Ce protocole utilise notre algorithme de clusterisation α-SSCA, déjà présenté dans le chapitre 3. Le plus important apport de α-SSCA, pour un protocole de routage, est de ne pas introduire de trafic de contrôle supplémentaire de clusterisation puisqu'il exploite les informations de routage dans la formation et la maintenance de la structure des clusters. Nous évaluons notre proposition CLSR à travers des simulations.

Le chapitre 7 propose une extension hiérarchique de notre protocole CLSR, appelé HCLSR. L'objectif principal de l'introduction de la hiérarchie dans CLSR est de rendre notre protocole plus scalable. En effet, regrouper les clusters proches dans un même cluster d'un niveau logique supérieur permet de réduire considérablement le trafic de contrôle et d'améliorer par la suite la scalabilité du routage. Nous effectuons une analyse théorique et expérimentale de notre proposition.

Le chapitre 8 présente un bilan des travaux de recherche décrits dans cette thèse et donne des perspectives possibles afin de guider les travaux futurs.

PREMIÈRE PARTIE

Clusterisation dans les réseaux mobiles sans-fil

CHAPITRE 2

ÉTAT DE L'ART DES ALGORITHMES DE CLUSTERISATION

LES ALGORITHMES de clusterisation sont utilisés dans le cadre de plusieurs problèmes liés aux réseaux informatiques. Ils permettent de découper le réseau en un ensemble de groupe afin d'optimiser des fonctions et des services du réseau. Ce chapitre ne couvre pas tous les algorithmes de clusterisation, mais il tente de caractériser ces algorithmes et cible ceux les plus importants et les plus connus dans la littérature.

Sommaire

2.1 Introduction 22
2.2 Intérêts et coûts de la clusterisation 22
2.3 Classification des algorithmes de clusterisation 24
2.3.1 Rayon des clusters 26
2.3.2 Métrique de sélection des cluster-heads 26
2.3.3 Structure de l'ensemble des cluster-heads 27
2.3.4 Autres critères de classification 29
2.4 Métriques d'évaluation des performances des algorithmes de clusterisation 30
2.5 Quelques approches de clusterisation 31
2.5.1 Les algorithmes de clusterisation à 1 saut 31
2.5.2 Les algorithmes de clusterisation à k sauts 38
2.6 Conclusion 43

2.1 Introduction

Dans les réseaux auto-organisables, la clusterisation consiste à diviser le réseau en un ensemble de nœuds géographiquement proches. Elle présente alors une solution intéressante pour simplifier et optimiser les fonctions et les services du réseau. En particulier, elle permet au protocole de routage de fonctionner plus efficacement en réduisant le trafic de contrôle dans le réseau et en simplifiant le processus d'aiguillage des données. Plusieurs algorithmes de clusterisation ont été proposés et évalués. Ces algorithmes ont des caractéristiques différentes et sont conçus pour satisfaire certains objectifs selon le contexte dans lequel la clusterisation est déployée (routage, sécurité, conservation de l'énergie, etc.). Plusieurs études [Abbasi and Younis, 2007, Wei and Chan, 2006, Yu and Chong, 2005] se sont intéressées à établir une classification des approches existantes de clusterisation.

Dans ce chapitre, nous présentons une étude des algorithmes de clusterisation dans les réseaux "auto-organisables ". Cette étude distingue les principales caractéristiques des approches de clusterisation et donne une classification des algorithmes proposés. Une classification systématique de ces approches permet de mieux comprendre et de proposer des améliorations. Nous détaillons également dans ce chapitre quelques algorithmes de clusterisation.

2.2 Intérêts et coûts de la clusterisation

La technique de clusterisation consiste à organiser les nœuds du réseau en des groupes virtuels appelés "clusters". Les nœuds, géographiquement voisins, sont regroupés dans un même cluster selon certaines règles. Dans un cluster, on retrouve généralement trois types de nœuds comme le montre la figure 2.1 : "cluster-head", nœuds "passerelles"et nœuds "ordinaires "dite aussi membre. Dans chaque cluster, un nœud est élu comme chef de groupe, appelé cluster-head, qui possède des fonctions supplémentaires (routage, accès au médium,etc.). Une passerelle est un nœud non-cluster-head qui possède des liens inter-clusters et peut donc accéder à des clusters voisins et acheminer les données entre eux, tandis qu'un nœud ordinaire est un nœud non-cluster-head qui ne possède pas des liens avec les autres clusters.

La clusterisation représente une solution prometteuse pour les réseaux "auto-organisables"à grand nombre de nœuds [Per, Belding-Royer, 2002]. Cette technique de structuration du réseau possède au moins trois avantages. En premier, la structure de clusters permet la réutilisation des ressources du réseau [Lin and Gerla, 1995, 1997]. Deux clusters peuvent utiliser la même fréquence ou le même jeu de code s'ils ne sont pas voisins [Hou and Tsai, 2001]. En plus, les nœuds de chaque cluster sont supervisés par leur cluster-head qui peut coordonner l'accès au canal, épargnant ainsi les ressources gaspillées dans la retransmis-

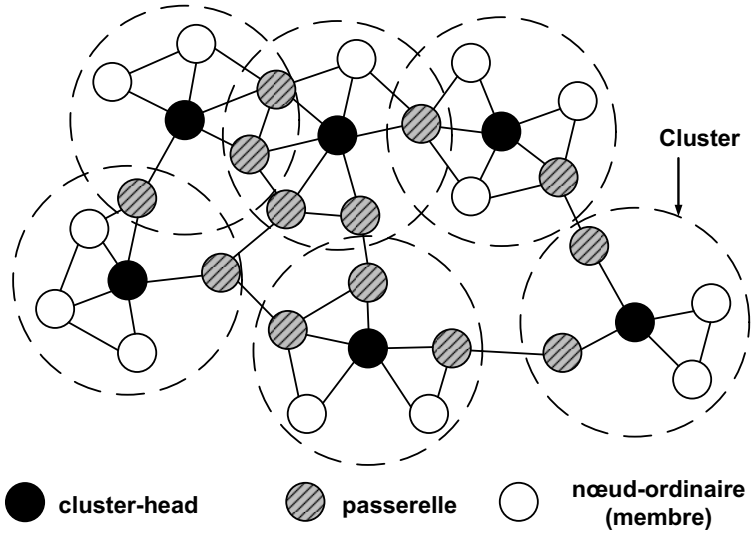


Figure 2.1 – Exemple d'une structure des clusters.

sion due aux collisions. Le deuxième avantage réside dans le routage. En effet, l'ensemble des cluster-heads et des passerelles peut former une dorsale virtuelle pour le routage inter-cluster. Ainsi, la génération et la diffusion des informations de routage peuvent se limiter à cet ensemble de nœuds [Kozat and al., 2001, Pearlman and Haas, 1999], ce qui évite les diffusions fréquentes des informations de routage pouvant surcharger le réseau et dégrader considérablement ses performances. Enfin, la structure de cluster fait apparaître le réseau plus petit et plus stable aux yeux de chaque nœud du réseau [McDonald and Znati, 1999]. Quand un nœud mobile change son cluster d'attache, seuls les nœuds, faisant partie du même cluster du nœud ayant quitté, actualisent leurs informations [Chen et al., 1999, Iwata et al., 1999]. Donc les changements locaux n'affectent pas le réseau entier mais uniquement une partie. Ainsi, l'information traitée et stockée par chaque nœud sera réduite.

La clusterisation a l'avantage d'être une alternative attrayante pour remédier aux problèmes de passage à l'échelle. Cependant, cette technique possède des inconvénients. En effet, construire et maintenir une structure de clusters demande un coût supplémentaire comparé à un réseau plat. L'analyse du coût de la clusterisation permet d'évaluer son efficacité et son passage à l'échelle. Cette analyse peut être quantitative ou qualitative. D'après les travaux de [Yu and Chong, 2005], nous mettons l'accent sur trois coûts de la clusterisation :

- Le trafic de contrôle périodique requis pour la maintenance de la structure des clusters [Amis et al., 2000, Hong et al., 2002, Lin and Gerla, 1997, Ohta et al., 2003] constitue une charge supplémentaire du réseau. Comme le nombre de messages de contrôle croît avec le nombre de nœuds, dans un réseau dense et large, le trafic considérable de contrôle va puiser une partie des ressources limitées en termes de bande passante, de temps de traitement et d'énergie. Les messages de contrôle vont concur-

rencer conjointement avec le trafic de données pour accéder au canal, surchargeant ainsi le réseau. Dans de telles conditions, les applications des couches supérieures ne peuvent plus fonctionner correctement.

- L'effet cascade de la re-clusterisation est aperçu dans certaines propositions de clusterisation et peut affecter considérablement les performances des protocoles des couches supérieures. Cet effet consiste à déclencher la reconstruction de la totalité de la structure des clusters sur tout le réseau si un certain événement local survient (le mouvement ou le départ d'un nœud) et résulte à la réélection du cluster-head [Amis et al., 2000, Basu et al., 2001, Chatterjee et al., 2000]. Cet effet aura lieu lorsqu'une réélection d'un seul cluster-head affecte la structure de plusieurs clusters et déclenche une chaîne de réélection des cluster-heads sur tout le réseau [Chiang et al., 1997].

- La phase de formation des clusters est une phase de collecte d'informations et de prise de décision du rôle à jouer par chaque nœud. La durée de cette phase mesurée en nombre de tours demeure une métrique importante et représente un coût considérable dans un algorithme de clusterisation. Certaines propositions supposent que les nœuds restent statiques pendant la phase de formation. Ainsi, plus cette phase est longue plus est le temps où les nœuds restent non mobiles est long. Pour la majorité des approches de clusterisation, la procédure de formation peut s'exécuter en parallèle sur tout le réseau ce qui rend la convergence de formation des clusters plus rapide. Mais dans ces approches, les nœuds ne prennent pas le même temps de décision de leur rôle. En plus, pour les mécanismes de clusterisation où la décision d'un nœud se base sur les décisions diffusées dans son voisinage, parfois des nœuds attendent indéfiniment dans des réseaux denses ou à grande envergure. Ainsi, la phase de formation peut ne pas être bornée et peut prendre des valeurs variables en fonction de la topologie du réseau.

2.3 Classification des algorithmes de clusterisation

Plusieurs classifications des mécanismes de clusterisation ont été proposées dans la littérature [Abbasi and Younis, 2007, Agerwal and Motwani, 2009, Wei and Chan, 2006, Yu and Chong, 2005]. La classification des approches existantes peut se baser sur plusieurs critères. La première classification évidente repose sur les critères et les métriques utilisés dans le choix du cluster-head. Dans cet axe, certains mécanismes de clusterisation ont choisi des critères simples comme l'identifiant [Amis and Prakash, 2000, Chen et al., 2002a, Chen and Liestman, 2002, DOW et al., 2002, Gerla and Tsai, 1995, Lin and Gerla, 1997], alors que d'autres approches ont adopté des sélections plus raffinées. Ces sélections combinent cer-

tains critères dans le but de sélectionner les nœuds les plus appropriés pour jouer le rôle de cluster-head [Basagni, 1999, Chatterjee et al., 2000]. Une autre classification peut prendre en compte le nombre de sauts qui séparent un nœud ordinaire du cluster-head auquel il est rattaché. Dans cette classification, on retrouve deux catégories : les algorithmes à 1 saut et ceux à plusieurs sauts.

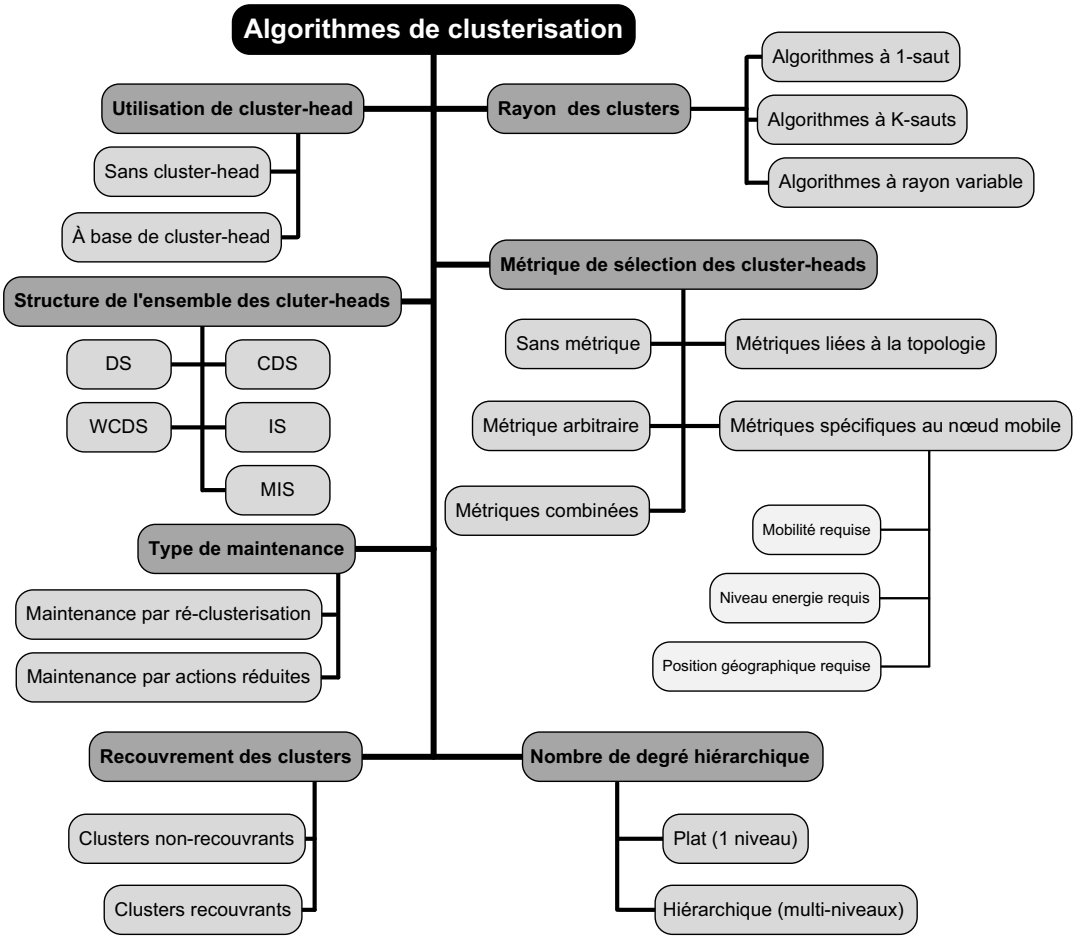


Figure 2.2 – Classification des algorithmes de clusterisation.

L'étude de classification [Yu and Chong, 2005] faite par Yu et Chong classe les algorithmes de clusterisation pour les réseaux ad hoc par objectifs et distingue six catégories d'algorithmes : les algorithmes se basant sur la construction d'un ensemble dominant, les algorithmes à charge équilibrée, les algorithmes à faible coût de maintenance, les algorithmes conscients de la mobilité, les algorithmes d'économie d'énergie et les algorithmes à métriques combinées. La classification proposée par Agarwal et Motwani [Agerwal and Motwani, 2009] distingue également six classes d'algorithmes en reprenant les quatre dernières classes de l'étude de Yu et Chong et en ajoutant les deux classes suivantes : les algorithmes à base d'identifiant, les algorithmes à base de connectivité. Les auteurs dans [Wei and Chan, 2006] ont classé les algorithmes de clusterisation par type de réseaux :

réseaux ad hoc et réseaux de capteurs. Dans les réseaux de capteurs, l'énergie des nœuds est limitée et généralement non rechargeable. Ainsi, les approches de clusterisation visent à maximiser l'efficacité de l'utilisation de l'énergie. Dans les réseaux ad hoc, la mobilité des nœuds peut changer rapidement la topologie ce qui augmente l'overhead des messages de maintenance. Les approches de clusterisation pour ce type de réseau s'intéressent alors à gérer la maintenance de la topologie, à la mobilité des nœuds ou à la réduction de l'overhead. D'autres études se sont focalisées uniquement sur les algorithmes de clusterisation pour les réseaux de capteurs [Abbasi and Younis, 2007, Dechene et al., 2006].

Dans cette section, nous allons présenter et distinguer des critères de classification des approches existantes de clusterisation, illustrés dans la figure 2.2.

2.3.1 Rayon des clusters

Le rayon d'un cluster exprime la valeur maximale de la distance qui sépare le cluster-head à l'ensemble de ses membres. Cette distance est exprimée en nombre de sauts. Nous distinguons alors trois classes d'algorithmes : les algorithmes à 1 saut, les algorithmes à K sauts et les algorithmes à rayon variable.

2.3.2 Métrique de sélection des cluster-heads

Plusieurs métriques ont été proposées dans la littérature pour élire l'ensemble des cluster-heads. Nous proposons de classer les algorithmes de clusterisation dans cinq classes selon le type des métriques employé lors de la sélection des cluster-heads. Ces cinq classes sont les suivantes :

- ***Sans métrique :*** regroupe les algorithmes qui déclare les cluster-heads sans avoir recours à aucune métrique de sélection.
- ***Métrique arbitraire :*** représente une valeur choisie arbitrairement et généralement non significative. Dans cette classe, nous trouvons les algorithmes qui utilisent des valeurs aléatoires ou l'identifiant des nœuds.
- ***Métriques liées à la topologie :*** dans cette classe nous groupons tous les algorithmes qui utilisent une métrique issue de la topologie du réseau. Parmi les métriques qui appartiennent à cette classe nous citons : le degré de connectivité, le k-degré de connectivité et la k-densité.
- ***Métriques spécifiques au nœud mobile :*** les algorithmes de cette classe utilisent des métriques très spécifiques aux nœuds mobiles. Dans le cas général, ces algorithmes supposent connaitre certaines informations sur chaque nœud tel que sa mobilité, la puissance des signaux reçus, le niveau d'énergie ou encore sa position géographique. Dans cette classe nous avons distingué trois sous classes qui sont :

- ⊙ ***Mobilité requise :*** regroupe les algorithmes qui supposent que les nœuds ont une connaissance de leur mobilité.
- ⊙ ***Niveau energie requis :*** regroupe tous les algorithmes qui utilisent la valeur de l'énergie restante dans les batteries des nœuds comme métrique de sélection des cluster-heads.
- ⊙ ***Position géographique requise :***cette classe d'algorithmes suppose que chaque nœud a la capacité de déterminer sa position géographique par exemple à travers un système GPS.

■ ***Métriques combinées :*** plusieurs algorithmes combinent plusieurs métriques de différents types pour élire l'ensemble de cluster-heads.

2.3.3 Structure de l'ensemble des cluster-heads

Les cluster-heads élus forment des structures particulières parmi lesquelles nous citons :

- ■ **DS** (Dominating Set) : ensemble dominant ;
- ■ **CDS** (Connected Dominating Set) : ensemble dominant connecté ;
- ■ **WCDS** (Weakly Connected Dominating Set) : ensemble dominant faiblement connecté ;
- ■ **IS** (Independant Set) : ensemble indépendant ;
- ■ **MIS** (Maximal Independant Set) : ensemble indépendant maximal.

Dans ce qui suit, nous allons donner la définition de ces différentes structures. Pour ce faire, nous avons besoin d'introduire quelques notions nécessaires. Le réseau des nœuds est modélisé par un graphe non orienté $G = (V, E)$, où V représente l'ensemble des nœuds et l'ensemble E correspond à l'ensemble des arêtes représentant les liens existants entre les différents nœuds. La notation $N(v)$ correspond au voisinage d'un nœud v et inclut les sommets adjacents à v en comptabilisant le nœud v aussi. Les structures présentées se basent sur la notion de dominance. Ainsi, un nœud appartenant à ces structures est appelé dominant, les autres nœuds étant des dominés.

Définition 2.1 Ensemble dominant (DS)

Un ensemble dominant (DS : Dominating Set) est un ensemble de nœuds tel que tout nœud du réseau est voisin d'au moins un nœud de cet ensemble. Soit V' un ensemble dominant du grapghe G(V,E), alors :

$$\forall\, u \in V, \exists\, v \in V' / v \in N(u) \tag{2.1}$$

Définition 2.2 Ensemble dominant connecté (CDS)

Un ensemble dominant connecté (CDS : Connected Dominating Set) est un ensemble dominant où tous les nœuds dominants sont directement connectés. Soit V' un ensemble dominant connecté du graphe G(V,E) alors :

$$\forall\, u \in V, \exists\, v \in V'/v \in N(u) \tag{2.2}$$

$$\forall (u,v) \in V'^2, \exists\, c = chemin_{u \to v}/\forall\, w \in c, w \in S \tag{2.3}$$

Définition 2.3 Ensemble dominant faiblement connecté (WDS)

Un ensemble dominant faiblement connecté (WDS : Weakly Connected Dominating Set) est un ensemble dominant tel que l'ensemble des dominants avec les arêtes dont une des extrémités est un dominant forme un ensemble connexe [Dunbar et al., 1997].

Définition 2.4 Ensemble indépendant (IS)

Un ensemble indépendant (IS : Independent Set) est un ensemble de nœuds du graphe tel qu'aucune paire de nœuds de l'IS ne se trouvent voisins dans le graphe. Soit S un sous ensemble indépendant du graphe G(V,E) alors :

$$\forall\, u \in S, \neg(\exists\, v \in S/v \in N(u)) \tag{2.4}$$

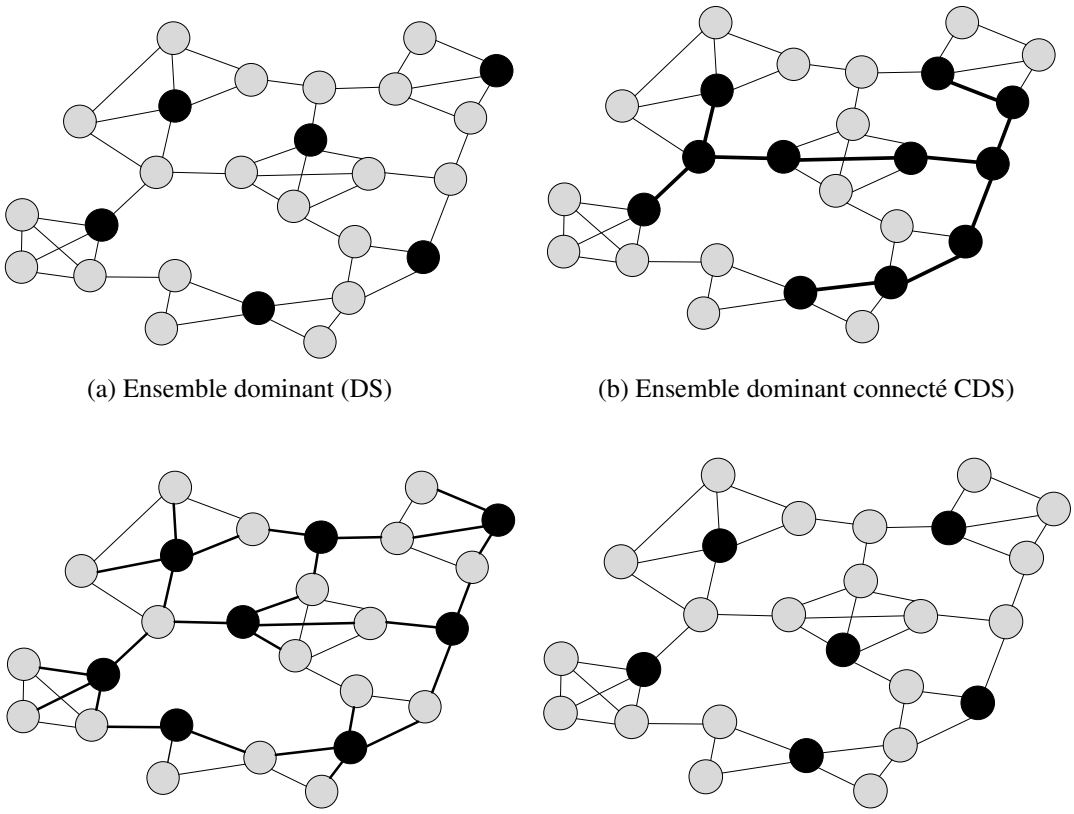
Définition 2.5 Ensemble indépendant maximal (MIS)

Un ensemble indépendant maximal (MIS : Maximum Independent Set) est un ensemble indépendant présentant une cardinalité maximale.

Nous notons qu'un ensemble indépendant est complet pour l'inclusion si un nœud du graphe ne peut être ajouté à l'ensemble indépendant sans qu'il perde sa propriété d'indépendance. En plus par définition, un MIS ne possède pas dans le cas général la propriété d'unicité.

La figure 2.3 illustre quatre types de structures. Dans la figure 2.3a les nœuds noirs forment un sous ensemble dominant du réseau. La figure 2.3b présente un ensemble dominant connecté formé par les nœuds noirs. Dans la figure 2.3c nous avons représenté un

ensemble dominant faiblement connecté. Tandis que le dernier cas 2.3d de la figure montre un cas possible d'un ensemble indépendant.



(a) Ensemble dominant (DS)

(b) Ensemble dominant connecté CDS)

(c) Ensemble dominant faiblement connecté (WCDS)

(d) Un ensemble indépendant (IS)

Figure 2.3 – Exemple de structures de l'ensemble des cluster-heads

2.3.4 Autres critères de classification

- Nombre de degré hiérarchique : Les algorithmes de base se contentaient d'élire un ensemble de cluster-heads. On parle alors d'algorithme plat à un seul niveau. D'autres algorithmes élisent des cluster-heads dans un nouveau niveau parmi l'ensemble des cluster-heads déjà sélectionnés au niveau précédent. Ils introduisent alors une hiérarchie de cluster-heads. Dans ce cas, nous parlons des algorithmes hiérarchiques multi-niveaux
- Type de maintenance : Dans certain nombre d'algorithmes la maintenance se fait par re-clusterisation. D'autres algorithmes proposent d'éliminer l'effet cascade introduit par la re-clusterisation. Ils proposent certaines règles et actions de maintenance qui agissent localement et limitent le diamètre des modifications dans la structure des clusters. Nous avons alors distingué deux classes selon le type de maintenance qui sont : maintenance par re-clusterisation, maintenance par actions réduites.
- Recouvrement des clusters : Les algorithmes de clusterisation peuvent créer des clus-

ters recouvrants où non recouvrants. Un algorithme construit des clusters recouvrants s'il existe des nœuds qui sont liés à plus qu'à cluster-head. Les clusters sont dits non recouvrants si chaque nœud appartient à un seul cluster.

2.4 Métriques d'évaluation des performances des algorithmes de clusterisation

Vu la multitude et la diversité des algorithmes de clusterisation proposés, il est difficile de trouver les bonnes métriques d'évaluation de performance. En se basant sur les travaux de [Sharmila and Rajsingh, 2008, Wei and Chan, 2006, Yu and Chong, 2005], nous avons déduit les groupes de métriques suivants :

- Messages explicites du processus de clusterisation :
  - *Overhead* : l'*overhead* encouru durant le temps d'opération du réseau.
  - Complexité en message pour un changement de topologie : le nombre de messages échangés entre les nœuds pour accomplir une ré-organisation valide des clusters après un changement dans la topologie du réseau.
- Caractéristiques des clusters :
  - Taille moyenne d'un cluster : le nombre moyen de nœuds gérés par un cluster-head.
  - Le nombre moyen de clusters : le nombre moyen de clusters formés dans le réseau.
  - Distance totale aux cluster-heads : la somme des distances entre les nœuds et leurs cluster-heads.
- Complexité des algorithmes : Complexité en temps pour un changement de topologie qui représente le temps passé pour accomplir une ré-organisation valide des clusters après un changement dans la topologie du réseau.
- Stabilité de la structure des clusters :
  - Durée moyenne de vie d'un cluster-head : la durée moyenne pendant laquelle un nœud joue le rôle de cluster-head.
  - Durée moyenne de vie d'un membre : la durée moyenne pendant laquelle un nœud est associé à son cluster-head.
  - Nombre de changements de cluster-head : le nombre de fois qu'un cluster-head change de statut.
  - Nombre de mise à jour de l'ensemble dominant : l'ensemble des cluster-heads est appelé un ensemble dominant si tous les nœuds dans le réseau sont dans cet ensemble ou sont voisins des nœuds de cet ensemble.
  - Nombre de ré-affiliation : le compteur de ré-affiliation est incrémenté quand un nœud est dissocié de son cluter-head et se rattache comme membre à un autre cluster-head à l'intérieur de l'ensemble dominant courant.

- Facteur d'équilibre de charge :
  - Durée moyenne de vie d'un cluster-head : la durée moyenne pendant laquelle un nœud joue le rôle de cluster-head.
  - Variance de la durée de vie d'un cluster-head : la variance de la durée pendant laquelle un nœud joue le rôle de cluster-head.

2.5 Quelques approches de clusterisation

Dans cette section, nous présentons et nous analysons les principaux algorithmes proposés de construction de clusters dans les réseaux auto-organisables. L'analyse détaillée de ces algorithmes nous permet de tirer profits de leurs avantages et améliorer leurs limitations. De nombreuses approches de clusterisation ont été proposées. Une grande partie des ces approches construisent des clusters à 1 saut où chaque nœud est à un saut de son cluster-head. D'autres algorithmes génèrent des clusters à k-sauts où chaque nœud est à au plus k sauts de son cluster-head. La majorité des alogorithmes proposent l'utilisation d'une métrique de base qui permet d'élire un cluster-head et invoque parfois d'autres métriques pour rompre les égalités. Néaumoins, il existe des algorithmes qui n'utilisent aucune métrique particulière pour structurer le réseau en clusters. Nous avons distingué quatre types de métriques de base utilisées par les algorithmes proposés. La métrique arbitaire est une métrique qui est donnée de façon arbitaire ou aléatoire à un nœud comme l'identifiant d'un nœud. Le deuxième type de métrique conserne les métriques calculées à partir de la topologie du réseau comme la connectivité d'un nœud appelée aussi degré d'un nœud. D'autres métriques sont plutôt spécifiques au nœud mobile comme sa mobilité relative et son énergie résiduelle. Il y a des algorithmes qui combinent plusieurs métriques pour calculer un poids utilisé dans l'éléction du cluster-head.

2.5.1 Les algorithmes de clusterisation à 1 saut

2.5.1.1 Les algorithmes à métrique arbitraire

L'un des algorithmes de clusterisation les plus anciens est LCA (Linked Cluster Algorithm) [Ephremides et al., 1987], connu sous le nom de l'algorithme du plus petit identifiant (ID) ou du plus petit ID. Chaque nœud se déclare cluster-head ou non en se basant sur son identifiant et ceux de ses voisins. Un nœud peut être dans l'un de ces quatre statuts : ordinaire, cluster-head, membre, ou passerelle. Initialement, tous les nœuds ont un statut de nœud ordinaire. Les nœuds diffusent périodiquement la liste des nœuds qu'ils sont capables de détecter. La formation des clusters suit les règles suivantes :

① Si un nœud u possède le plus petit identifiant dans son voisinage à un saut, il se déclarera comme cluster-head et ses voisins à un saut dont les identifiants sont supérieurs à celui du cluster-head le joignent et deviennent des nœuds membres.
② Si non, il attendra que tous ses voisins à un saut déclareront leurs statuts. Ainsi si un parmi eux se déclare cluster-head alors le nœud u déclare à son voisinage à un saut son statut de nœud membre.
③ Une fois que tous les nœuds ont soit le statut de membre ou de cluster-head, alors si un nœud a parmi ses voisins à un saut plus qu'un cluster-head, il se déclarera passerelle.

L'algorithme du plus petit ID considère uniquement les nœuds ayant le plus petit identifiant qui sont choisis arbitrairement sans prendre en considération d'autres caractéristiques sur les nœuds élus comme cluster-head. Comme l'identifiant du nœud ne change pas au cours du temps, les nœuds ayant un petit identifiant sont les plus probables à être choisis comme cluster-head que les nœuds ayant un grand identifiant. Ainsi, si ces cluster-heads gardent leur statut de cluster-head pour une longue durée, il en résulte qu'ils consomment rapidement leur énergie et par conséquent ils causent des goulets d'étranglement dans leurs clusters.

La maintenance de la structure des clusters créée par LCA s'avère coûteuse car le mouvement d'un nœud peut engendrer des réactions en chaîne et nécessiter la reconstruction totale de la structure. LCC (Least Cluster Change) [Chiang et al., 1997] est alors proposé comme une version améliorée de l'algorithme LCA qui ajoute une étape de maintenance permettant de minimiser le coût de la restructuration du réseau des clusters. L'algorithme LCC distingue deux phases : la phase de formation des clusters et la phase de maintenance des clusters. La formation des clusters est la même que celle dans LCA, c'est-à-dire que les nœuds ayant le plus petit identifiant dans leur voisinage sont élus des cluster-heads. Les clusters ne sont reconstruits que dans les deux cas suivants :

❶ Si deux cluster-heads i et j se trouvent voisins, alors l'un des deux (i ou j) devra abandonner le rôle de cluster-head selon le critère "Plus Grand Degré" et/ou " Plus Petit ID".
❷ Si un nœud non-cluster-head se déplace en dehors des clusters formés et ne migre pas vers un cluster existant alors il deviendra cluster-head et formera un nouveau cluster.

Ainsi, si un nœud non-cluster-head u migre d'un cluster $C(i)$ vers un cluster $C(j)$ alors la réélection du cluster-head dans le cluster $C(j)$ n'aura pas lieu même si le nœud u possède

les capacités d'être cluster-head dans $C(j)$. De cette façon, LCC améliore considérablement la stabilité des clusters. Mais le coût de la restructuration du réseau reste toujours un peu onéreux car les réactions en chaînes de reconstruction ne sont que limitées et ne sont pas complètement supprimées vu qu'un seul nœud peut relancer le processus de clusterisation s'il n'existe pas de cluster-head dans son voisinage.

L'algorithme Adaptive Clustering (AC) [Lin and Gerla, 1997] utilise le statut de cluster-head uniquement pour la formation des clusters. Une fois les clusters formés , la notion de cluster-head disparait et les nœuds du cluster jouent le même rôle. La motivation est que les cluster-heads peuvent devenir des goulots d'étranglement et dépensent leurs ressources rapidement que les autres nœuds. Cet algorithme utilise la même métrique que l'algorithme LCA (le plus petit identifiant) pour le choix des cluster-heads. Pour construire les clusters, chaque nœud i maintient initialement son propre ID et les IDs de ses voisins à 1-saut dans un ensemble noté Γ_i. L'algorithme AC est différent de l'algorithme LCA par le non-recouvrement des clusters. En plus, l'algorithme se termine après que chaque nœud envoie un et un seul message de clusterisation. En effet, chaque nœud n'est autorisé à diffuser son état (cluster-head, membre) que s'il possède le plus petit identifiant dans tous les nœuds de l'ensemble Γ_i.
Les règles de formation de l'algorithme AC sont les suivantes :

---

① Un nœud i se déclare cluster-head s'il possède l'identifiant le plus petit dans Γ_i. Si c'est le cas, il doit diffuser sa décision à tous ses voisins dans un message d'état (ID,C-ID).

② A la réception d'un message d'état d'un nœud i, les voisins de i supprime i de leur ensemble Γ.

③ Si un nœud i annonce qu'il est cluster-head (ID = C-ID), ses voisins s'attachent à lui si leur statut est non spécifié ou si le cluster-head auquel ils sont membres a un identifiant plus grand que i.

④ Le processus continue jusqu'à ce que l'ensemble Γ de chaque nœud soit vide.

Après la phase de formation, les cluster-heads ne jouent plus aucun rôle dans la phase de maintenance. Chaque nœud doit connaître son voisinage à 2-sauts. Par suite, il sait si les membres de son cluster sont à 2-sauts de lui. Si un nœud i trouve que la distance entre lui et un nœud j du même cluster devient à 3-sauts, il invoque la procédure de maintenance dont les règles sont :

---

❶ Si i est un nœud à 1 saut du nœud ayant la plus haute connectivité, il reste dans le cluster et le nœud j quitte le cluster et rejoint un cluster voisin.

❷ Si non, i rejoint un cluster voisin

❸ S'il n'existe pas un cluster convenable, i crée son propre cluster.

Bien que n'utilisant pas la notion de cluster-head, la maintenance de cet algorithme maintient le nœud de plus grand degré au centre du cluster, ce qui peut revenir au même que l'élire comme cluster-head.

2.5.1.2 Les algorithmes à métrique liée à la topologie

L'algorithme HCC (Highest Connectivity Cluster) [Gerla and Tsai, 1995] choisit le nœud ayant le plus grand nombre de voisins (plus grand degré) comme cluster-head. Initialement, tous les nœuds sont non couverts. Chaque nœud diffuse la liste des nœuds qu'il peut détecter. La phase de formation des clusters applique les règles suivantes :

---

① Un nœud est élu cluster-head s'il a le plus grand nombre de voisins. Dans le cas d'égalité, le nœud ayant le plus petit identifiant ID deviendra cluster-head.
② Un nœud qui n'a pas encore choisi son cluster-head demeure dans son état non couvert, si non il devient un nœud couvert.
③ Un cluster-head qui a élu un autre nœud comme son cluster-head, abandonne son rôle de cluster-head.
④ Les nœuds appartenant à plusieurs clusters jouent le rôle de passerelle.

L'idée de HCC est que des nœuds ayant un grand degré sont de bons candidats pour être cluster-head car ils couvrent un grand nombre de nœuds et ainsi le nombre de clusters sera réduit. Dans un environnement mobile, cet algorithme produit des cluster-heads qui ne sont pas susceptibles de jouer leur rôle comme cluster-heads pour très longtemps puisque leur degré changent très fréquemment contrairement à l'algorithme du plus petit ID où les nœuds de faible identifiant ont tendance à garder le statut de cluster-head plus longtemps.

L'algorithme LCC déjà présenté possède deux versions LCC-lowest-ID et LCC-hc. Dans la première version, c'est la métrique identifiant qui est utilisée alors que dans la deuxième version, le degré d'un nœud est employé pour élire le cluster-head. C'est ainsi que LCC-hc est un algorithme à métrique liée à la topologie tandis que LCC-lowest-ID est à métrique arbitraire.

Les algorithmes de clusterisation génèrent soit des clusters recouvrants ou des clusters non-recouvrants. Dans des clusters recouvrants, un nœud peut appartenir à plusieurs clusters. L'inconvénient des algorithmes à clusters recouvrants est que la ré-affiliation d'un nœud mobile d'un cluster à un autre provoque la restructuration des clusters concernés et dans certains cas la restructuration de tout le réseau. Pour remédier à ce problème, des algorithmes de clusterisation ont été proposés, permettant la création de clusters non-recouvrants c'est-à-dire un nœud ne peut appartenir qu'à un seul cluster. Ceci permet de

limiter l'opération de maintenance au niveau d'un seul cluster lors de la ré-affiliation d'un nœud d'un cluster à un autre.

L'algorithme 3hBAC (*3-hop Between Adjacent Cluster-heads*) [Yu and Chong, 2003] génère des clusters non-recouvrants et impose que les cluster-heads de deux clusters adjacents se trouvent à trois sauts les uns des autres. Pour assurer cette condition, 3hBAC introduit un nouveau statut qui est nœud membre-invité dans la structure des clusters. Il définit aussi une priorité lors de la déclaration des nœuds à travers un attribut HP (*Head Priority*). Cet attribut peut avoir l'une des trois valeurs suivantes :

- "SET" : signifie que le nœud est fixé à l'état cluster-head.
- "DECLINED" : indique que le nœud ne peut pas être cluster-head. Il a un statut non-cluster-head (membre, membre-invité, non-couvert).
- "UNSET" : utilisé dans le cas d'un nœud qui n'est pas encore couvert et peut devenir un cluster-head.

Avant de procéder à la formation des clusters, chaque nœud collecte les informations de son voisinage à 1-saut à travers un échange de messages HELLO. Le message HELLO contient les informations suivantes : l'identifiant du nœud, son degré, son statut ainsi que son attribut HP.

Initialement tous les nœuds mobiles ont un attribut HP fixé à la valeur UNSET. Les procédures qui assurent la formation des clusters sont les suivantes :

---

① Le processus de formation commence toujours par le voisinage du nœud ayant le petit identifiant. Ainsi, le nœud n_i avec le plus petit identifiant se décide en premier. Par conséquent, c'est le nœud n_j possédant le plus haut degré de connectivité dans le voisinage du nœud n_i qui crée le premier cluster. Après la création du premier cluster, la procédure de formation des clusters peut se faire en parallèle dans le réseau.

② Tous les nœuds ayant un HP=UNSET et qui sont membres ou voisins directs d'un nœud membre sont battus. Ils ne peuvent plus servir comme cluster-head. Dans ce cas, leur HP passe à la valeur DECLINED.

③ Un nœud mobile ayant le plus haut degré de connectivité dans son voisinage et procédant un HP=UNSET, se déclare cluster-head s'il est voisin à un nœud dont l'attribut HP est DECLINED.

④ Tous les nœuds voisins d'un cluster-head deviennent des nœuds membres de son cluster.

⑤ Si un nœud mobile ayant un HP=DECLINED trouve que tous ses voisins ont l'attribut HP égal à DECLINED, alors il choisit l'un de ses voisins membres comme AP (*Access Point*) et joint le cluster correspondant à ce voisin comme membre-invité.

La maintenance des clusters a pour objectif de minimiser les changements de la structure. Elle est assurée par les règeles suivantes :

❶ Lorsque deux cluster-heads se déplacent dans la même portée de transmission, celui qui possède le degré de connectivité le plus faible abandonne son rôle de cluster-head et devient membre du cluster-head gagnant. Ses voisins deviennent soit des nœuds membres s'ils se trouvent dans le voisinage du cluster-head gagnant, soit des nœuds non spécifiés et tentent de joindre un autre cluster.

❷ Quand un nœud mobile non-cluster-head se déplace en dehors de la portée de transmission de tous les cluster-heads existants, s'il peut communiquer avec des nœuds à l'état membre, il choisit l'un comme étant son AP et devient membre-invité du cluster correspondant.

❸ Sinon, si tous les voisins sont à l'état membre-invité, il crée alors un nouveau cluster et le serve comme cluster-head. Cette situation est la seule qui permet la création de nouveau cluster. Ainsi, les nouveaux clusters déclarés sont au minimum à 3-sauts des autres clusters.

❹ 3hBAC introduit une fonction de fusion. Chaque cluster-head vérifie si tous les membres de son cluster peuvent accéder à un autre cluster-head comme étant membre ou membre-invité. Si c'est le cas, il abandonne son rôle de cluster-head et joint un cluster voisin comme membre-invité.

2.5.1.3 Les algorithmes à métrique spécifique au nœud mobile

MOBIC (Lowest Relative Mobility Clustering Algorithm) [Basu et al., 2001] est algorithme distribué qui applique le même algorithme que LCA et HCC mais implique une autre métrique qui est la mobilité relative des nœuds. La mobilité relative d'un nœud représente le rapport des niveaux de puissance des transmissions successives reçues par un nœud de ses voisins. Ainsi, le nœud ayant la plus faible mobilité dans son voisinage gardera un voisinage plus stable au cours du temps et sera donc un bon candidat pour être cluster-head.

MOBIC utilise la même procédure de maintenance que LCC à laquelle il ajoute une règle supplémentaire pour minimiser le coût de l'opération de maintenance. Si deux cluster-heads i et j se trouvent voisins, alors l'un des deux n'abandonnera son statut de cluster-head qu'après une certaine période Δt, sinon ils garderont tous les deux leur statut. Cela permet de ne pas faire appel à l'opération de maintenance une fois que deux cluster-heads se trouvent dans le même voisinage.

L'optimisation introduite dans MOBIC dans la procédure de maintenance a contribué de réduire à plus du tiers le nombre de changements des cluster-heads relativement au protocole LCC. Cependant, les limitations de l'algorithme LCC ne sont pas totalement éliminées.

De plus, MOBIC exige que les nœuds doivent être capables d'estimer le niveau de signal avec leurs voisins pour calculer la mobilité relative.

2.5.1.4 Les algorithmes à métriques combinées

Plutôt qu'impliquer une seule métrique pour élire les cluster-heads, une catégorie d'algorithmes de clusterisation utilisent une fonction qui combine plusieurs métriques. Cette fonction représente le poids d'un nœud. Cette catégorie d'algorithmes vise à élire un ensemble de cluster-heads qui sera le plus adapté à une topologie pour satisfaire un ou plusieurs besoins. Par exemple, dans un réseau de capteurs où l'énergie est un facteur important, le paramètre représentant l'énergie peut avoir un poids plus important dans la fonction combinée de la métrique résultante.

WCA (Weighted Clustering Algorithm) [Chatterjee et al., 2002] est un exemple classique dans cette classe. En effet, il utilise une somme pondérée de quatre métriques dans le calcul du poids d'un nœud u. Ces métriques sont la différence de degré $D(u)$, la somme des distances avec ses voisins $P(u)$, la mobilité relative moyenne $M(u)$ et le temps de service en tant que cluster-head $T(u)$ tel que :

$$Poids(u) = \alpha \times D(u) + \beta \times P(u) + \gamma \times M(u) + \delta \times T(u) \tag{2.5}$$

$$\text{Avec } \alpha + \beta + \gamma + \delta = 1$$

- La différence de degré $D(u)$ est la différence entre le degré du nœud u et une valeur M qui représente le nombre de nœuds qu'un cluster head peut servir. Les auteurs dans [Chatterjee et al., 2002] n'explicitent pas le moyen de déterminer la valeur de M.
- Les distances $P(u)$ entre les nœuds sont calculées à l'aide d'un GPS.
- La mobilité relative $M(u)$ est calculée de la même manière que dans MOBIC.

Le nœud ayant le plus petit poids dans son voisinage devient cluster-head. Les nœuds couverts par un cluster-head ne peuvent pas participer à l'élection des cluster-heads. Cette procédure est répétée jusqu'à ce que chaque nœud soit assigné à un cluster. La maintenance des clusters est décrite par les règles suivantes :

- Si un nœud membre change son cluster, il n'a pas de re-clusterisation même si le cluster-head ne possède plus le plus petit poids.
- Si un nœud se déplace dans une région non couverte par un cluster-head, la procédure de réélection de cluster-heads est alors déclenchée.

Bien que WCA ait de meilleures performances que les autres algorithmes que nous avons présentés en termes du nombre de clusters produits et du nombre de changements de cluster-heads, il a un inconvénient pour connaître le poids de tous les nœuds avant de commencer le processus de clusterisation. En conséquence, l'overhead induit par WCA est très élevé. En plus, WCA épuise rapidement les batteries des cluster-heads.

2.5.1.5 Les algorithmes sans métrique

Les algorithmes classiques de clusterisation requièrent que tous les nœuds envoient des messages explicites pour former et maintenir la structure des clusters ce qui laisse la clusterisation un des principaux sources d'overhead.

PC (Passive Clustering) [Kwon et al., 2003] est un algorithme de clusterisation qui n'utilise pas des paquets explicites pour le contrôle de la clusterisation, d'où le nom PC qui veut dire clusterisation passive. En plus, PC n'utilise aucune métrique particulière. Dans PC, un nœud mobile peut être dans un de ces quatre états : initial, cluster-head, gateway ou nœud ordinaire. Au départ, tous les nœuds mobiles sont dans l'état initial. Seuls les nœuds mobiles dans l'état initial qui ont la possibilité d'être cluster-heads. Quand un cluster-head potentiel avec l'état "initial" veut envoyer quelque chose, il se déclare comme un cluster-head en indiquant son état dans le paquet envoyé. Les voisins peuvent s'apercevoir de la demande du cluster-head en regardant l'état du cluster dans le paquet, puis ils enregistrent l'identifiant du cluster CID et le temps de réception du paquet. Un nœud mobile qui reçoit une demande d'un seul cluster-head devient un nœud ordinaire, et un nœud mobile qui écoute plus qu'une demande devient un gateway. Comme PC n'envoie aucun message explicite de clusterisation pour maintenir la structure des clusters, chaque nœud est responsable d'actualiser son propre état en gardant un temporisateur. Par exemple, quand un nœud ordinaire qui ne reçoit plus de paquet de son cluster-head pour une période donnée, revient à l'état initial. PC peut former et maintenir sa structure de clusters sans échanger des paquets explicites de contrôle. Donc, il peut éliminer complètement l'overhead perçu dans la clusterisation active.

2.5.2 Les algorithmes de clusterisation à k sauts

2.5.2.1 Les algorithmes à métrique arbitraire

Les auteurs de [Chen et al., 2002b] proposent un algorithme qui construit des k-clusters recouvrants en généralisant l'algorithme à un saut de Lin et Gerla [Lin and Gerla, 1997]. Chaque nœud doit connaître ses voisins à k sauts. Tout nœud non affilié possédant le plus petit ID parmi ses k-voisins non encore affiliés devient cluster-head. Lorsque tous les nœuds de son k-voisinage ayant un plus petit identifiant que lui ont diffusé leur décision d'être cluster-head ou de s'attacher à un autre cluster-head, un nœud peut prendre sa propre décision de s'attacher au nœud de son k-voisinage du plus petit identifiant s'étant déclaré cluster-head s'il existe, ou de créer son propre cluster si non. L'opération de maintenance reprend celle utilisée dans [Lin and Gerla, 1997] en prenant en compte le rayon du cluster. Cependant, nous retrouvons toujours les mêmes inconvénients que présentent les algorithmes générant des 1-clusters, à savoir un petit changement dans la topologie du réseau peut provoquer une restructuration de tout le réseau. En outre, dans les réseaux denses tels que les réseaux de capteurs, la taille des clusters peut être importante, ce qui peut épuiser

rapidement les batteries des cluster-heads et par la suite causer des goulets d'étranglement dans les clusters.

L'algorithme Max-Min-d-cluster [Amis et al., 2000] construit des d-clusters non recouvrants avec d un paramètre de l'heuristique. Cet algorithme utilise l'identifiant du nœud comme métrique d'élection du cluster-head et se déroule en trois tours. Lors du premier tour, chaque nœud diffuse son identifiant à ses voisins à d sauts et collecte leurs identifiants. Il en garde le plus grand des identifiants qu'il le diffuse de nouveau dans le deuxième tour. Chaque nœud garde le plus petit des identifiants reçus dans le deuxième tour (le plus petit parmi les plus grands). Le troisième tour consiste au choix du cluster-head basé sur les identifiants enregistrés lors des deux tours précédents. Si un nœud u a reçu son identifiant dans la deuxième phase, il devient cluster-head. Si non, si u a reçu un identifiant durant chacun des tours 1 et 2, il choisira le nœud ayant cet identifiant comme cluster-head. Si non, u élit cluster-head le nœud ayant le plus grand identifiant dans son voisinage à d-sauts. Cet algorithme produit une structure de clusters robuste, cependant, la durée du déroulement de l'algorithme n'est pas négligeable. En outre, cette technique augmente considérablement l'overhead du réseau puisqu'elle utilise un nombre important de messages pour élire les cluster-heads. D'autre part, cette technique ne tient pas compte de la mobilité des nœuds et du changement de la topologie.

Les mêmes auteurs de [Amis et al., 2000] ont proposé une version améliorée de leur propre algorithme Max-Min d-cluster appelée Cluster-head Load-Balancing [Amis and Prakash, 2000]. Le but est d'assurer un équilibre de charge entre les nœuds en évitant qu'un nœud reste cluster-head pour une longue durée et épuise ainsi ses ressources rapidement mais en assurant aussi qu'il lui reste suffisamment pour apporter une stabilité à la structure.

L'algorithme Cluster-head Load-Balancing introduit une notion d'identifiant virtuel VID (Virtual IDentifier) utilisée pour l'élection des cluster-heads. Ce VID est à la base l'identifiant du nœud auquel est ajouté une information de temps liée à la période pour laquelle un nœud peut reste cluster-head. Initialement, l'identifiant virtuel de chaque nœud est initialisé par son propre identifiant. A chaque période de temps, chaque nœud non cluster-head incrémente de 1 son identifiant virtuel jusqu'à ce que son VID atteigne une valeur maximale seuil MAX-VID. Le nœud ayant le plus grand identifiant virtuel dans son k-voisinage devient cluster-head. En cas d'égalité, le nœud ayant opéré le moins de fois comme cluster-head sera élu cluster-head et en cas d'égalité toujours, c'est le nœud qui a le plus grand identifiant ID deviendra cluster-head. Un nœud joue le rôle de cluster-head pour une période de temps TimeCH et après cette période, il initialisera son VID à zéro et abandonnera le statut de cluster-head. Lorsque deux cluster-heads se trouvent à moins de k sauts l'un de l'autre, le cluster-head ayant le plus faible VID abandonnera son statut.

Cet algorithme assure une certaine stabilité de la structure des k-clusters mais il nécessite durant l'élection des cluster-heads une synchronisation des nœuds (non cluster-heads) pour l'incrémentation de leurs VID et des cluster-heads pour comptabiliser la période pour

jouer ce rôle. Cette synchronisation est coûteuse en nombre de messages échangés entre les différents nœuds.

Dans [Nagpal and Coore, 1998], Nagpal et Coore ont proposé CLUBS, un algorithme qui forme des clusters à partir des diffusions locales et converge dans un temps proportionnel à la densité locale des nœuds. La formation des clusters dans CLUBS est basée sur les trois caractéristiques suivantes :

- Chaque nœud dans le réseau doit être connecté à un cluster.
- Le diamètre maximal de tous les clusters dans le réseau doit être le même.
- Les clusters doivent supporter la communication intra-cluster.

L'algorithme forme des clusters avec un maximum de deux sauts. Chaque nœud dans le réseau participe à la formation des clusters en choisissant un nombre aléatoire dans un intervalle d'entiers fixe. Ensuite, il compte à rebours de ce nombre. Si le compte régressif n'avait pas été interrompu par un autre nœud voisin et il atteint le zéro, il s'annonce comme cluster-head et diffuse un message d'invitation. Quand un nœud voisin reçoit le message d'invitation qui vient depuis un diamètre de deux sauts, il arrête le compte à rebours, accepte l'invitation et joint le cluster. Un nœud qui a joint un cluster "partisan" n'a plus le droit d'être cluster-head. Comme CLUBS autorise le chevauchement des clusters, les nœuds partisans restent en écoute des messages d'invitation supplémentaires et peuvent être partisan à plus qu'un cluster-head. Si un nœud est en concurrence pour devenir cluster-head détecte une collision ou a reçu un message corrompu, il devient un nœud partisan et assume que plusieurs cluster-heads essayent de le recruter en même temps. Il peut trouver son cluster-head plus tard. L'algorithme se termine quand tous les nœuds dans le réseau joignent quelques clusters comme cluster-head ou comme un partisan. Quand deux cluster-heads se retrouvent à un saut, leurs clusters s'écrouleront et le processus d'élection de cluster-head sera relancer.

2.5.2.2 Les algorithmes à métrique liée à la topologie

Les auteurs de [Ramalingam et al., 2002] introduisent une métrique qu'ils appellent *associativité* qui représente la stabilité relative des nœuds dans leur voisinage. Pour chaque nœud, l'associativité comptabilise le temps que chacun des nœuds de son voisinage reste effectivement dans son voisinage et en fait la somme sur chaque voisin. A chaque période de temps, un nœud u considère quels sont ses voisins actuels déjà présents lors de la période précédente et ajoute 1 à la valeur associée à chacun d'eux. Si un voisin apparait, il prend la valeur 1 et si un voisin a disparu, la valeur qui lui était associée prend 0. A chaque période de temps, l'associativité de u est la somme des valeurs associées à chacun de ses voisins. Cette valeur prend donc en compte la stabilité de u (si u est relativement stable dans son voisinage, il aura une forte associativité) et le degré des nœuds. La formation des clusters suit le procédé suivant. Un nœud considère les nœuds de son k-voisinage ayant un degré supérieur à une valeur seuil et élit parmi eux celui ayant la plus forte associativité. Dans

le cas d'égalité, il choisit celui ayant le plus fort degré et si encore il y'a égalité, c'est celui ayant le plus faible identifiants qui sera élu. Les clusters résultants sont des k-clusters recouvrants mais qui visent à être plus stables dans le temps et dans l'espace que ceux se basant sur le simple degré.

2.5.2.3 Les algorithmes à métrique spécifique au nœud mobile

HEED [Younis and Fahmy, 2004] est un algorithme distribué de clusterisation proposé pour les réseaux de capteurs. HEED construit un graphe connecté de cluster-heads à plusieurs sauts. Les clusters formés sont disjoints. Les cluster-heads sont élus en se basant sur l'énergie résiduelle de chaque nœud. Les nœuds ayant une haute énergie résiduelle deviennent cluster-heads. Cet algorithme invoque aussi une métrique secondaire utilisée pour rompre l'égalité. Le cas d'égalité dans ce contexte inclut le cas quand un nœud se trouve à proximité de deux cluster-heads et quand deux cluster-heads se trouvent dans le même rayon. Cette métrique est une métrique de coût de communication qui peut être fonction de la proximité des voisins ou la densité du cluster. L'algorithme d'élection des cluster-heads suit une méthode probabiliste avec itérations. À la première itération, pour chaque nœud i , le seuil d'éligibilité est fixé à :

$$CH_{prob} = max(C_{prob} \times (\frac{E_i}{E_{max}}), p_{min}) \tag{2.6}$$

où C_{prob} est le ratio recherché de cluster-heads sur le réseau, E_i l'énergie résiduelle du nœud i et *pmin* une probabilité minimale d'éligibilité pour assurer la terminaison de l'algorithme. À chaque itération CH_{prob} est multiplié par 2 (pour atteindre un maximum de 1 où le nœud est garanti d'être élu cluster-head). Les études de performances de HEED [Younis and Fahmy, 2004] montrent que pour certaines situations, l'autonomie d'un réseau de capteurs (temps avant l'indisponibilité du premier capteur par insuffisance d'énergie) est plus que doublée par rapport à d'autres algorithmes. À première vue, HEED s'avère plus adapté, par la prise en compte de l'énergie résiduelle des nœuds, pour des réseaux de capteurs d'énergie résiduelle hétérogène.

2.5.2.4 Les algorithmes à métriques combinées

MobDhop (Mobility-based D-hop) [ER and Seah, 2004] est un algorithme distribué qui forme des clusters de diamètre variable avec la mobilité du nœud. Cet algorithme suppose que chaque nœud peut mesurer la puissance du signal reçu. Cinq paramètres doivent être calculés :

- La distance estimée entre les nœuds : basée sur la puissance du signal reçu du voisin.
- La mobilité relative entre les nœuds : est la différence de la distance estimée d'un nœud pour deux moments successifs. Elle indique si les deux nœuds s'éloignent ou se rapprochement.

- La variation de la distance estimée en fonction du temps
- La stabilité locale : elle est fonction de la variation de la distance estimée et de la mobilité relative.
- La distance moyenne estimée.

Cet algorithme se déroule en trois étapes : étapes de découverte, de fusion et de maintenance. L'étape de découverte a pour but de former des clusters à 2-sauts. Dans cette étape, les nœuds échangent périodiquement des messages Hello qui incluent la valeur de stabilité locale (initialisée à l'infini). Après, chaque nœud calcule sa valeur de stabilité locale et la diffuse à ses voisins. Le nœud ayant la plus petite valeur de stabilité locale est élu cluster-head et sa valeur de stabilité locale sera la valeur de stabilité du groupe GS. Le nœud qui entend des messages de plusieurs cluster-heads devient gateway, si non il est membre. L'étape de fusion peut être initiée par un nœud non-spécifié qui demande de se joindre à des clusters voisins ou quand deux gatways voisins demandent de fusionner leur clusters. Les deux conditions de fusion doivent être accomplies :

- La variation de la distance estimée entre deux nœuds fusionnés doit être inférieure ou égale au minimum de la valeur de stabilité du groupe des deux clusters.
- La distance moyenne entre deux gateways doit être inférieure ou égale au maximum de la distance moyenne estimée des deux clusters.

La maintenance des clusters est invoquée quand un nœud arrive ou quitte le réseau.

- Quand un nœud arrive, il y'a fusion.
- Quand un cluster-head quitte le réseau , ses voisins exécutent l'étape de découverte et un nouveau cluster-head est élu. Durant la période sans cluster-head, les voisins à 2-sauts initialisent le processus de fusion pour joindre d'autres clusters si les conditions de fusion sont satisfaites.

2.5.2.5 Les algorithmes sans métrique

Lin et Chu [Lin and Chu, 2000] ont proposé un autre algorithme pour générer des k-clusters qui n'utilise aucune métrique pour le choix des cluster-heads et s'exécute comme suit :

- A l'arrivée d'un nœud u, il passe à l'état "Initialisation". Puis, il demande l'état de ses voisins : sont-ils à l'état "Initialisation" ou sont-ils affiliés à des clusters, et dans ce dernier cas, à quelle distance se situe leur cluster-head correspondant ?
- Si tous les voisins de u sont à l'état "Initialisation" alors le nœud u se déclarera comme cluster-head et diffusera sa décision dans son voisinage. Tous les k-voisins de u qui ne sont pas attachés à aucun cluster plus proche que u, s'attachent au cluster formé par le nœud u.
- Sinon, le nœud u joint le cluster de son voisin dont le cluster-head est le plus proche et qui se trouve à au plus k sauts de lui.
- Si tous les cluster-heads de ses voisins sont à k sauts du nœud u, alors u se déclarera

cluster-head et invitera ses k-voisins qui ont des cluster-heads plus éloignés que u à le rejoindre.

Dans cet algorithme, l'opération de maintenance se déclenchera quand deux cluster-heads se trouvent à une distance D qui est inférieure à k sauts (D < k). Le cluster-head ayant le plus petit identifiant cèdera son rôle et ses membres doivent s'affilier à d'autres cluster-heads.

Cet algorithme génère un nombre réduit de k-clusters non recouvrants où les cluster-heads sont éloignés d'au moins D sauts ce qui assure une certaine stabilité à la structure. Cependant, l'opération de maintenance est coûteuse quand un cluster-head cède son rôle et déclenche alors des réactions en chaîne. D'autre part, l'algorithme proposé ne prend pas en compte la contrainte taille dans la formation des clusters, ce qui permet de générer des clusters ayant une taille importante dans le cas des réseaux denses tels que les réseaux de capteurs.

Fernandess et Malkhi [Fernandess and Malkhi, 2002] ont proposé un algorithme original pour générer des k-clusters. Cet algorithme s'exécute en deux phases. La première phase consiste à construire un arbre couvrant du réseau (Spanning Tree) basé sur un ensemble dominant connecté de cardinalité minimale (MCDS). Ce MCDS est construit par l'algorithme présenté dans [Alzoubi et al., 2002], néanmoins les auteurs indiquent que n'importe quelle autre méthode peut être utilisée. La deuxième phase consiste à partitionner l'arbre couvrant en 2k-sous arbres (un 2k sous-arbre est un arbre de diamètre au plus 2k sauts). Chaque sous-arbre correspond à k-cluster. Cependant, la complexité temporelle et la complexité en nombre de message est en O(n) (n est le nombre de nœud du réseau). Par conséquent, cet algorithme n'est pas scalable. En plus, la maintenance de la structure des clusters n'est pas abordée par les auteurs.

2.6 Conclusion

Dans ce chapitre, nous avons étudié la technique de clusterisation qui est largement utilisée dans les réseaux auto-organisables. Cette technique bien qu'elle soit attrayante pour remédier aux problèmes de passage à l'échelle, elle introduit des coûts supplémentaires comparée à un réseau plat. Plusieurs variétés d'algorithmes de clusterisation ont été proposées dans la littérature. Nous avons présenté une classification de ces algorithmes qui met en évidence les principales caractéristiques des approches proposées. Nous avons également cerné les métriques les plus pertinentes pour évaluer les performances des algorithmes de clusterisation. Dans la dernière partie du chapitre, nous avons présenté une analyse détaillée du fonctionnement de certains algorithmes de clusterisation proposés dans la littérature. L'inconvénient majeur de la plupart des approches est de ne pas garder la structure des clusters stable dans un environnement dense, mobile et à large échelle. Fournir un mécanisme de clusterisation stable est important pour le bon fonctionnent des applications des

couches supérieures. Cette étude nous a permis de tirer profits des algorithmes proposés et de prendre en considération leurs carences pour proposer un nouvel algorithme de clusterisation. Cet algorithme, planifié pour être intégré dans un protocole de routage, a pour objectif de générer une structure de clusters stable. La stabilité de la structure des clusters permet de garantir la stabilité des routes et donc réduire le coût de leur maintenance et améliorer les performances du réseau. La conception et l'évaluation de cet algorithme fait l'objet du chapitre suivant.

CHAPITRE 3

PROPOSITION D'UN ALGORITHME DE CLUSTERISATION À STRUCTURE STABLE α-SSCA

CE CHAPITRE est consacré à la présentation d'un nouvel algorithme de clusterisation. Cet algorithme a comme objectif d'améliorer la stabilité de la structure des clusters tout en générant un nombre assez réduit de clusters dans le réseau. Il se base sur le degré de connectivité pour élire l'ensemble des cluster-heads et il introduit un nouveau facteur de stabilité pour la maintenance des clusters.

Sommaire

3.1 Introduction 46

3.2 Description de l'algorithme α-SSCA 46

3.2.1 Détection du voisinage 47

3.2.2 Formation des clusters 48

3.2.3 Maintenance des clusters 50

3.3 Analyse expérimentale 54

3.3.1 Choix de la métrique de sélection des cluster-heads 55

3.3.2 Variation du nombre de nœuds 56

3.3.3 Variation de la densité des nœuds 58

3.4 Impact de la stabilité sur le routage 60

3.5 Conclusion 63

3.1 Introduction

Dans ce chapitre, nous proposons un nouvel algorithme de clusterisation pour le routage dans les réseaux sans fils "auto-organisables " nommé α-SSCA (α-Stability Structure Clustering Algorithm) [Guizani et al., 2011a]. La clusterisation est une solution prometteuse pour les protocoles de routage puisqu'elle permet de minimiser l'utilisation des ressources et de réduire la quantité d'informations de contrôle diffusée dans le réseau. Dans ce cadre, un algorithme de clusterisation doit prendre en considération les particularités de la fonction de routage et doit alors :

- réduire le trafic de contrôle lié à la clusterisation ;
- réduire le nombre de clusters : afin de réduire le trafic de contrôle du protocole de routage ;
- augmenter la stabilité de la topologie des clusters : ce qui augmente la validité des routes, et par la suite augmenter l'efficacité du protocole de routage. L'amélioration de la stabilité permet aussi de réduire les mises à jour des informations de routage lors des changements de la topologie ;
- être simple : pour réduire le temps de calcul relatif à la formation et la maintenance des clusters ;
- avoir une convergence rapide : qui permet de réduire les effets des changements de topologie et avoir des routes fiables aussi tôt que possible.

Le principe de base de notre algorithme α-SSCA est d'augmenter modérément le nombre de clusters dans l'objectif d'améliorer la stabilité de la topologie des clusters générée. α-SSCA est un algorithme de clusterisation à un saut. Il se base sur les informations locales de la topologie pour la sélection des cluster-heads. Ce choix est important puisque ces informations sont partagées avec l'algorithme de routage. Par la suite, il réduit considérablement les informations explicites introduites par l'algorithme de clusterisation.

3.2 Description de l'algorithme α-SSCA

L'algorithme α-SSCA opère en trois étapes. La première étape consiste à collecter les informations nécessaires à l'élection des cluster-heads. La seconde étape permet d'élire les cluster-heads et former la structure virtuelle des clusters. Enfin, il maintient la structure des clusters lors des changements de la topologie du réseau à cause des mouvements des nœuds. La procédure de formation des clusters s'inspire de l'algorithme 3hBAC qui éloigne les cluster-heads lors de la formation.

Les algorithmes HCC, LCC et 3hBAC utilisent le degré de connectivité d'un nœud comme métrique pour élire les cluster-heads. Cette métrique correspond au nombre de voi-

sins de chaque nœud. Dans notre proposition, α-SSCA utilise une fonction *score* comme métrique d'élection des cluster-heads. Cette fonction *score* correspond au nombre de voisins qui ne sont pas encore décidés de leurs états dans la structure des clusters. Cette métrique est exploitée pour faire éloigner le maximum possible les cluster-heads les uns des autres. L'algorithme 3HBAC interdit lors de la formation des clusters que deux cluster-heads soient à une distance inférieure à trois sauts. Si c'est le cas, il doit résoudre ce conflit par l'abandon de l'un des deux cluster-heads. En revanche, l'algorithme α-SSCA accepte ce dernier cas de figure ce qui lui permet de réduire le nombre de changements d'état et d'accélérer la convergence de la procédure de clusterisation.

L'algorithme α-SSCA utilise une heuristique distribuée et tente de minimiser les informations explicites de la formation des clusters. En effet, notre proposition n'utilise que des messages de type "HELLO "comme ceux utilisés par les protocoles de routage à état de liens. Les messages HELLO permettent à chaque nœud de découvrir son voisinage à deux sauts et de se décider de son rôle dans la structure des clusters.

3.2.1 Détection du voisinage

Pour prendre sa décision, chaque nœud du réseau doit connaître son voisinage à deux sauts. Le message HELLO est échangé périodiquement entre les voisins à un saut. Comme nous l'avons introduit, ce message est celui utilisé par les protocoles de routage à état de liens auquel nous avons ajouté les informations nécessaires à la formation des clusters. Le paquet HELLO contient alors les informations suivantes :

- L'adresse de la source ;
- L'adresse du cluster auquel appartient la source ;
- Nombre de voisins de la source ;
- La liste des adresses des voisins de la source ;
- La liste des adresses des clusters des voisins de la source.

En collectant les paquets HELLO reçus, chaque nœud maintient deux listes, une liste pour les voisins à un saut et une deuxième liste pour les voisins à deux sauts.

Afin d'illustrer la découverte de voisinage, nous allons appliquer cette procédure à l'exemple de topologie du réseau présentée dans la figure 3.1. Initialement, aucun nœud n'appartient à la structure des clusters. L'adresse de diffusion (Max-Add) est utilisée alors comme identificateur de son cluster qui signifie qu'il n'est pas encore décidé. Chaque nœud doit alors diffuser dans son voisinage à un saut le message HELLO. Par exemple, le nœud 1 envoie le message HELLO suivant (1, Max-Add , 0, ϕ), ce message va permettre aux nœuds 2, 3 et 5 de découvrir la présence du nœud 1 et de l'ajouter dans leur listes des voisins à 1-saut. Après le premier tour d'échange des messages HELLO, chaque nœud détecte tous ces liens directs. Un deuxième tour sera alors nécessaire pour permettre la découverte des voisins à 2-sauts. En effet, le deuxième message HELLO envoyé par le nœud 1 est le suivant

(1, Max-Add , 3, ((2, Max-Add), (3, Max-Add), (5, Max-Add))). Ce message va permettre aux voisins de découvrir la présence des voisins à deux sauts via le nœud 1.

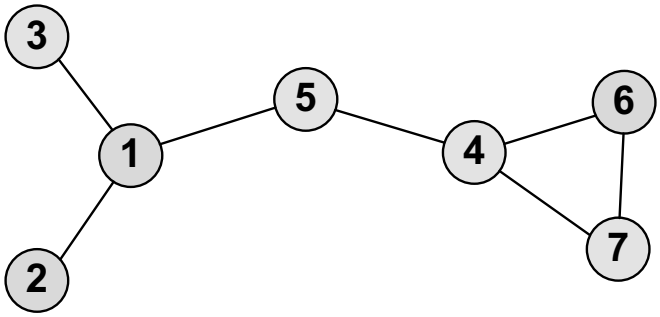


Figure 3.1 – Un exemple de topologie du réseau.

L'échange périodique des messages HELLO permet aussi de détecter la perte de lien entre les voisins directs. En effet, chaque nœud associe un temporisateur à chaque voisin direct. Ce temporisateur est fixé à une valeur supérieure à deux fois la période d'envoi d'un message HELLO. Si le temporisateur expire avant la réception d'un message HELLO du voisin auquel il est associé, le nœud peut alors considérer que le lien avec ce voisin est perdu. Il l'enlève de sa liste des voisins et informe les autres voisins à travers l'envoi d'un message HELLO.

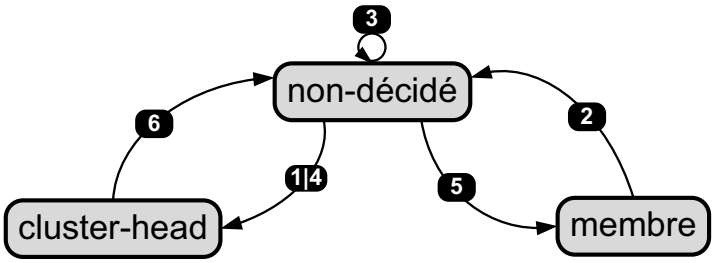


Figure 3.2 – Diagramme de transition d'état de l'algorithme SSCA.

3.2.2 Formation des clusters

Dans α-SSCA, chaque nœud peut prendre l'un des états suivants :

- ***non-décidé :*** cet état est l'état initial pris par chaque nœud. Il indique que le nœud n'appartient pas à la structure des clusters. Il est utilisé lorsque le nœud vient juste d'arriver ou il vient de quitter son cluster.
- ***cluster-head*** : cet état indique que le nœud est élu comme chef du cluster. Ce nœud va gérer son cluster.
- ***membre :*** lorsqu'un nœud se trouve voisin d'un cluster-head, il se joint à son cluster. Il devient alors un nœud membre de ce cluster.

Initialement tous les nœuds sont à l'état *non-décidé*. Lorsque le temps progresse, chaque nœud essaye de joindre la structure des clusters en prenant l'un des deux états finaux *Cluster-Head* ou *Membre*. Chaque nœud doit surveiller son voisinage pendant un temps supérieur à deux fois la période d'envoi d'un message HELLO ce qui lui permet de construire sa topologie locale à 2-sauts. Il peut ensuite choisir son rôle dans la structure des clusters.

Les clusters sont formés par transition d'état de tous les nœuds participants. La figure 3.2 illustre le diagramme des transitions d'état dans le processus de formation des clusters. Chaque nœud doit choisir son état conformément à l'une des règles suivantes :

Règle 3.1 Création d'un nouveau cluster

Si un nœud possède le plus haut *score* dans son voisinage à 1-saut, il crée un nouveau cluster et il se déclare comme étant *cluster-head* .

La règle 3.1 correspond à la transition 1 de la figure 3.2). Dans le cas où deux nœuds ont le plus haut *score*, le nœud ayant le plus petit identificateur se déclare cluster-head. À la suite de sa décision, chaque cluster-head envoie un message HELLO pour informer ses voisins de sa décision. Ce message invite les voisins non encore décidés pour se joindre à ce nouveau cluster.

Règle 3.2 Affiliation à un cluster

Si un nœud *non-décidé* est voisin d'un *cluster-head* , il se joint à son cluster. Il passe à l'état *membre*

La régle 3.2 correspond à la transition 2 de la figure 3.2. Chaque nouveau membre doit envoyer un message HELLO pour informer son cluster-head de son adhésion. Ce message permet aussi d'informer le reste de ses voisins de sa décision. Si un nœud est voisin à plus qu'un cluster-head, la cardinalité des clusters est utilisée comme affinité pour choisir le cluster. En cas d'égalité, l'identificateur des nœuds est utilisé comme deuxième métrique, le nœud choisit alors le cluster-head avec le plus petit identificateur.

Règle 3.3 Attente de priorité

Si un nœud n'a pas le plus haut *score* dans son voisinage, il doit attendre que tous les nœuds ayant un meilleur *score* dans son voisinage se décident.

Règle 3.4 Création contrainte d'un nouveau cluster.

Si un nœud n'est pas voisin d'un *cluster-head* et tous les nœuds avec un meilleur *score* sont décidés, le nœud se déclare alors comme étant *cluster-head* .

Afin d'illustrer la formation des clusters, la figure 3.3 représente un cas où deux nœuds 8 et 12 à l'état *non-décidé* essayent de joindre la structure des clusters. Comme le nœud 12 est voisin à deux cluster-heads, alors, conformément à la règle 3.3, il choisit de joindre le cluster 6 puisqu' il possède la plus haute cardinalité. Tandis que le nœud 8 n'est voisin d'aucun cluster-head et tous ses voisins sont déjà décidés, alors en appliquant la règle 3.4, il crée son propre cluster et se déclare comme nœud cluster-head.

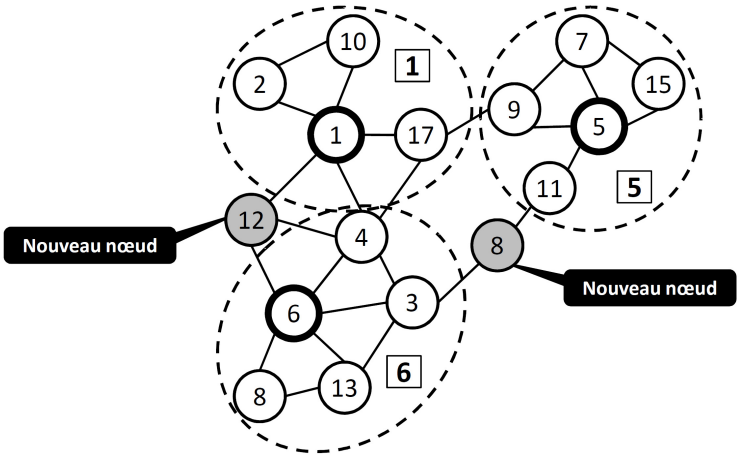
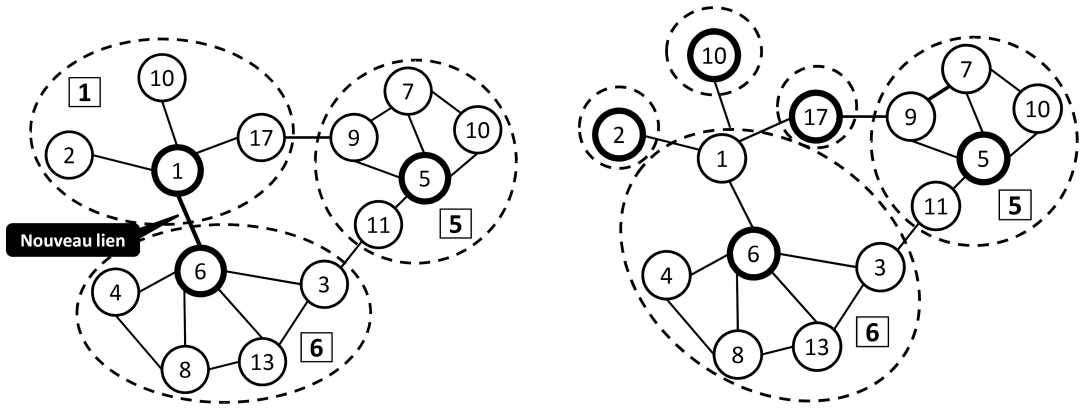


Figure 3.3 – Exemples de jonction à la structure des clusters.

3.2.3 Maintenance des clusters

Le processus de maintenance des clusters essaye d'adapter la structure des clusters à tous les changements de la topologie qui peuvent avoir lieu. Dans α-SSCA, la maintenance des clusters est inspirée de l'algorithme LCC. Elle a comme objectif d'améliorer la stabilité de la structure des clusters. Deux différences sont détectées entre l'algorithme LCC et le notre. La première est la métrique utilisée pour choisir les nœuds cluster-heads. LCC utilise le degré de connectivité des nœuds tandis que α-SSCA utilise le nombre de voisins non encore décidés. La deuxième différence est que α-SSCA relaxe plus la condition pour laquelle un cluster-head abandonne son rôle de cluster-head.

La figure 3.4 présente un cas de figure de maintenance des clusters lorsque deux cluster-heads se trouvent dans la même portée de transmission. Elle montre l'effet de l'abandon d'un cluster-head sur la stabilité de la structure des clusters. En effet, quand les deux cluster-heads 1 et 6 deviennent voisins, les algorithmes LCC-hc et 3HBAC imposent que le cluster-head numéro 1 avec le plus faible degré de connectivité abandonne son rôle. La figure 3.4b montre la nouvelle structure des clusters maintenue après le renoncement du cluster-head 1. En effet, cet action de maintenance va causer la suppression d'un cluster et l'apparition de trois nouveaux clusters ce qui entraîne en total quatre changements d'état. Ces changements



(a) Structure des clusters avant l'apparition d'un nouveau lien

(b) Structure des clusters après l'abandon du cluster-head 1

Figure 3.4 – Exemple de maintenance de la structure des clusters en cas d'apparition d'un nouveau lien

réduisent la stabilité de la structure des clusters et favorisent d'autres changements à l'avenir. De ce fait, nous allons relaxer plus la condition d'abandon dans notre proposition et nous allons accepter la présence de deux cluster-heads voisins dans certaines circonstances qui seront définies par un nouveau facteur de stabilité (définition 3.1). Ce facteur exprime le degré de chevauchement entre deux clusters. Par suite, l'abandon des cluster-head sera conditionné (3.2) par le seuil α qui est une valeur prédéfinie dans l'algorithme SSCA.

Définition 3.1 Facteur de stabilité

Soit u et v deux cluster-heads voisins, le facteur de stabilité $FS(u, v)$ est définit comme le rapport du nombre de membres de u qui sont aussi voisins de v et le nombre total de membres du cluster de u.

$$FS(u, v) = \frac{|\{X_i \in Cl(u)/X_i \in N(v)\}|}{|Cl(u)|} \tag{3.1}$$

Notons que $Cl(u)$ désigne l'ensemble des nœuds appartenant au cluster formé par le cluster-head u et que $N(u)$ correspond au voisinage à 1-saut du nœud u.

Définition 3.2 Condition α-abandon

On désigne par la condition α-abandon la condition pour laquelle un cluster-head u renonce à son rôle. u renonce à son rôle s'il est voisin d'un cluster-head v et si son facteur de stabilité $FS(u, v)$ est supérieur ou égal à une valeur prédéfinie α.

Nous allons présenter l'opération de maintenance par des actions à prendre dans chaque cas de changement de la topologie du réseau. En effet, les changements qui peuvent avoir

lieu dans la topologie du réseau peuvent être décomposées en deux événements élémentaires. Un nœud du réseau peut détecter la présence d'un nouveau nœud voisin ou il peut détecter la perte de lien avec un voisin existant.

3.2.3.1 Détection d'un nouveau lien

A cause de la mobilité des nœuds, certains nœuds qui n'étaient pas voisins peuvent se trouver dans la même portée de transmission. Chaque nœud qui reçoit un paquet HELLO de la part d'un nœud qui n'est pas dans sa liste des voisins détecte l'apparition d'un nouveau lien avec ce voisin. Il doit alors maintenir son état selon l'état de ce nouveau voisin. Dans le cas où les deux nœuds voisins ne sont pas des cluster-heads, la structure des clusters reste valide et ne sera pas modifiée. En effet, si les deux nœuds sont membres du même cluster ou appartiennent à deux clusters différents, l'apparition de ce nouveau lien ne fait qu'augmenter la connectivité inter-cluster ou intra-cluster.

Pour augmenter la stabilité de la structure des clusters, une issue est d'accepter la coexistence de deux cluster-heads dans la même portée de transmission dans certaines circonstances. Pour relaxer la condition pour laquelle un cluster-head abandonne son rôle, nous allons définir un facteur de stabilité entre deux cluster-heads voisins. Nous désignons par le cluster-head de faible métrique le cluster-head ayant le plus petit nombre de nœuds membres. En cas d'égalité, le cluster-head de plus haut identificateur est choisi . Ainsi, le cluster-head ayant la plus faible métrique vérifie la condition d'abandon et applique la règle 3.5.

Règle 3.5 Rapprochement de deux cluster-heads

Si deux cluster-heads u et v se trouvent dans la même portée de transmission, le cluster-head u renonce à son rôle si :

$$\mid Cl(u) \mid < \mid Cl(v) \mid \wedge FS(u,v) \geq \alpha \tag{3.2}$$

Notons qu'en cas d'égalité des cardinalités des deux cluster-heads l'identifiant des cluster-heads est utilisé comme deuxième métrique. Lorsqu'un nœud cluster-head abandonne son rôle, il revient à l'état *non-décidé* et informe ses voisin à travers l'envoi d'un message HELLO. Il tente, ainsi que ses membres, de rejoindre la structure des clusters en utilisant les mêmes règles que la formation des clusters.

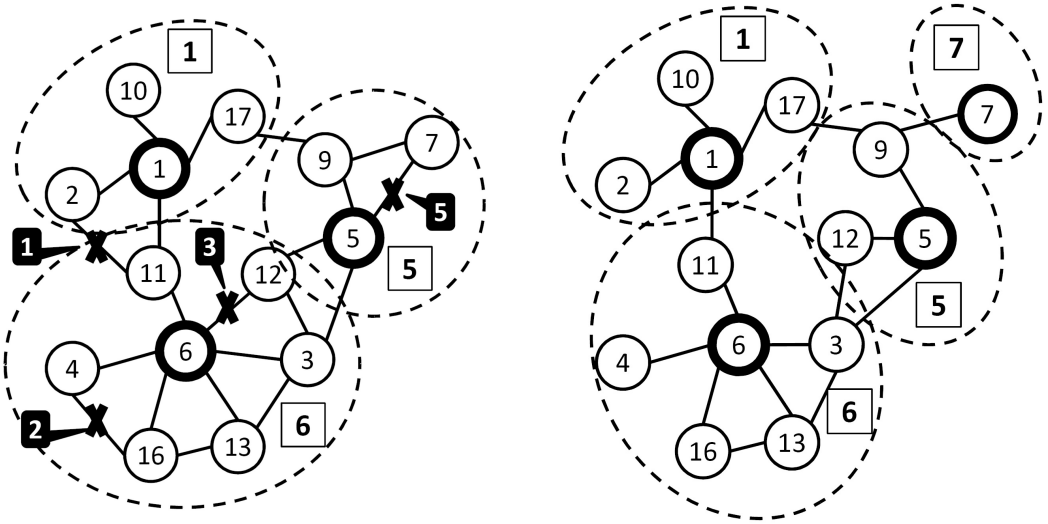
3.2.3.2 Perte de lien

La perte de lien entre deux nœuds voisins a lieu si ces deux nœuds ne sont plus dans la même portée de transmission. Dans ce cas, la structure des clusters n'est modifiée que si

cette structure n'est plus valide. En effet, la seule contrainte qu'il faut respecter est que tous les membres d'un cluster sont à 1-saut de leur cluster-head. Ainsi, la seule modification de la structure des clusters aura lieu lorsqu'un membre n'est plus voisin de son cluster-head. En effet, si le lien relie deux nœuds membres d'un même cluster ou deux nœuds appartenant à deux clusters différents la structure de cluster reste valide.

Règle 3.6 Perte de lien

Si le lien perdu relie un nœud membre à son cluster-head, ce nœud quitte le cluster. Il revient à l'état *non-nécidé* et tente alors de rejoindre la structure des clusters en utilisant les mêmes règles décrites lors de la formation des clusters. En plus, si le cluster-head est voisin d'un ou plusieurs autres cluster-heads il doit vérifier la condition d'abandon avec ses cluster-heads voisins.



(a) Structure avant la perte des liens (b) Structure après la maintenance

Figure 3.5 – Exemples de maintenance de la structure des clusters lors de la perte de lien

La figure 3.5 illustre la maintenance de la structure des clusters pour plusieurs cas de figure de perte de lien. Cette figure présente quatre cas différents de perte de lien selon l'état des deux nœuds liés.

- ***Cas 1 :*** présente la perte d'un lien entre les nœuds 2 et 11 appartenant à deux clusters différents respectivement 1 et 6. Dans ce cas, aucune modification ne sera apportée à la structure des clusters puisqu'elle reste valide. En effet, cette perte de lien ne fait que diminuer l'inter-connectivité entre les clusters voisins.
- ***Cas 2 :*** présente la perte d'un lien entre deux nœuds membres 4 et 16 du même cluster 6. Dans ce cas, la structure des clusters n'est pas aussi modifiée. Et la perte de ce lien a comme effet de diminuer l'intra-connectivité dans un même cluster.

- ***Cas 3 :*** expose la perte de lien entre le nœud membre 12 et son cluster-head le nœud 6. Cette perte de lien viole la contrainte que chaque nœud membre est à un saut de son cluster-head. Par la suite, le nœud 12 quitte le cluster et revient à l'état *non-décidé*. Pour rejoindre la structure des clusters, il applique les mêmes règles de la phase formation. Puisqu'il est voisin du cluster-head 5, il devient alors membre du cluster 5 conformément à la règle 3.2.
- ***Cas 4 :*** illustre aussi le cas de perte de lien entre un nœud membre et son cluster-head. Dans ce cas le nœud 7 membre du cluster 5 quitte son cluster. Puisque qu'il n'est voisin à aucun cluster-head, il applique la règle 3.4 et devient nœud cluster-head.

3.3 Analyse expérimentale

Dans cette section, nous évaluons par simulation les performances de notre proposition α-SSCA pour deux valeurs du facteur α ($\alpha = \frac{1}{2}$ et $\alpha = 1$). Nous comparons les résultats de simulation obtenus à ceux des algorithmes HCC et LCC-hc. Ces algorithmes ont été choisis puisqu'ils utilisent la même métrique (le degré de connectivité) que α-SSCA. Nous avons implémenté les différents algorithmes sous le simulateur NS-2 [Doe].

Dans cette première partie, nous évaluons seulement les algorithmes de clusterisation. Pour cette raison, nous n'avons pas envisagé un trafic de données. La durée de simulation a été fixée à 2000 secondes. Les résultats présentés tout au long de cette partie correspondent à une valeur moyenne des résultats obtenus par simulation de plusieurs scénarios choisis d'une manière aléatoire.

Dans les expériences qui seront présentées dans cette section, nous utilisons IEEE 802.11 comme modèle pour la couche liaison de donnée. Nous supposons que la bande passante de tous les nœuds du réseau est de 2Mbits/s et que la portée de transmission radio est de 250m. Nous supposons aussi que tous les nœuds ont la même adéquate capacité de mémoire-tampon.

Nous définissons la densité des nœuds comme le nombre de nœuds se trouvant dans un cercle de rayon égal à la portée de transmission (250 m). Cette densité correspond au nombre moyen de voisins à un saut de chaque nœud. Le modèle de la mobilité est choisi le même que dans des travaux similaires [Boukerche, 2004, Broch et al., 1998, Das et al., 2000]. Les nœuds se déplacent suivant le modèle RWP (Random Waypoint) [Johnson and Maltz, 1996] dans une zone carrée. La dimension de la zone de déplacement est calculée à partir du nombre de nœuds et de la valeur de la densité des nœuds.

Étant donné que notre proposition α-SSCA a comme objectif d'améliorer la stabilité de la structure des clusters formés, nous avons choisi de présenter les quatre métriques suivantes :

- ***Le nombre moyen de clusters :*** il s'agit du nombre moyen de cluster-heads formés

dans le réseau. Il représente la moyenne des valeurs instantanées prises tout au long de la durée de la simulation. Cette métrique est la plus répondue dans la comparaison des algorithmes de clusterisation.

- ***Le nombre de changements de cluster-head :*** le nombre de fois qu'un cluster-head change de statut (de cluster-head vers membre ou de membre vers cluster-head).
- ***Le nombre de ré-affiliations :*** le compteur de ré-affiliation est incrémenté quand un nœud est dissocié de son cluster et se rattache comme membre à un autre cluster-head.
- ***Durée moyenne de vie des cluster-heads :*** durée moyenne pendant laquelle un nœud joue le rôle de cluster-head.

Nous notons aussi que nous n'avons pas représenté la taille du trafic de contrôle explicite relative au processus de clusterisation pour la simple raison que les trois algorithmes utilisent la même métrique. Dans la suite, la taille des messages échangés est très proche pour les trois algorithmes.

Les résultats présentés tout au long de cette partie correspondent à une valeur moyenne des résultats obtenus par simulation de plusieurs scénarios choisis de manière aléatoire. En effet, les scénarios sont générés par le générateur Mobgen qui utilise le modèle de mobilité RWP (Random WayPoint).

Comme notre but est d'améliorer les performances à large échelle, nous allons représenter les métriques d'évaluation mesurées pour des valeurs assez grandes du nombre de nœuds dans le réseau. La densité des nœuds est aussi fixée à des valeurs moyennes et grandes. Nous avons choisi de représenter trois cas de figures :

- ***Expérimentation 1 :*** Étant fixée la valeur de la densité à 20 nœuds et la valeur de la mobilité dans l'intervalle [3-5] m/s, nous présentons les résultats des différentes métriques en fonction du nombre de nœuds dans le réseau.
- ***Expérimentation 2 :*** Le nombre de nœuds étant fixé à 400 nœuds. Les nœuds se déplacent avec une vitesse dans l'intervalle [2-5] m/s et prennent un temps de pause dans l'intervalle [20-60] seconds. Nous varions la valeur de la densité des nœuds.

3.3.1 Choix de la métrique de sélection des cluster-heads

L'amélioration de la stabilité de la structure des clusters peut ,dans la phase de maintenance, augmenter le nombre de clusters générés. Pour cette raison, l'algorithme α-SSCA essaye de réduire le nombre de clusters lors de phase de formation des clusters. Il adopte alors la métrique du nombre de voisins non encore décidés afin d'éloigner les clusters, et par la suite réduire le nombre total de clusters. Dans cette section, nous nous intéressons à l'effet du choix de cette métrique pendant la phase de formation des clusters. Nous avons établi une expérimentation de cette phase en utilisant trois différentes métriques connues

par leur nombre réduit de clusters générés qui sont :

- Le nombre de voisins directs (HC : degré de connectivité) : utilisé dans la sélection des cluster-heads par plusieurs algorithmes tels que HCC, LCC-hc et 3HBAC.
- La 1-densité (1-D) [Mitton et al., 2004] : d'un nœud u est le rapport du nombre de liens entre u et ses voisins augmenté du nombre de liens entre les voisins de u par le nombre de ses voisins.
- Le nombre de voisins non-décidés (HC-ND) : est la métrique choisie dans notre proposition α-SSCA et elle est utilisée par l'algorithme Guha et Khuller [Guha and Khuller, 1998].

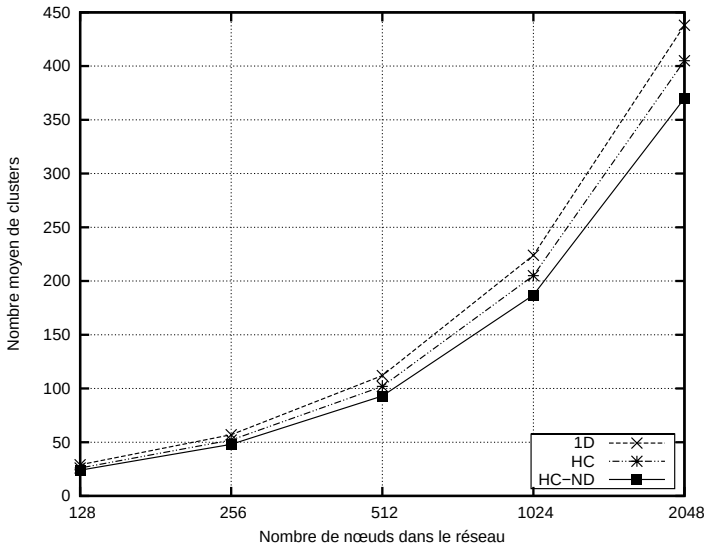
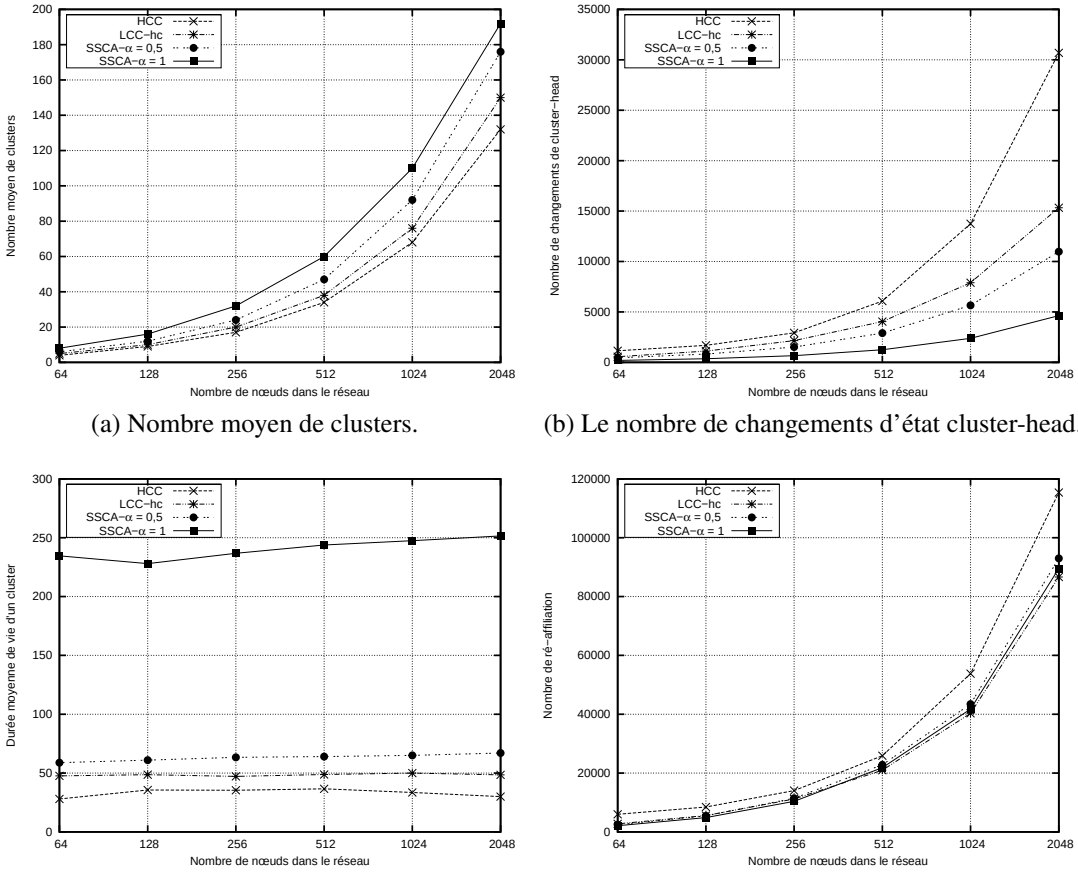


Figure 3.6 – Le nombre moyen de clusters formés en fonction des métriques.

La figure 3.6 illustre le nombre de clusters générés pendant la phase de formation des clusters en utilisant les trois métriques précédentes. Elle présente les résultats enregistrés en fonction du nombre de nœuds dans le réseau, pour le cas des paramètres de l'expérimentation 1. Les résultats obtenus montrent que le nombre de clusters le plus réduit est généré en utilisant la métrique du nombre de voisins non encore décidés. Ceci confirme le choix de cette métrique dans notre proposition α-SSCA.

3.3.2 Variation du nombre de nœuds

La figure 3.7 montre les résultats des simulations en fonction du nombre de nœuds dans le réseau pour le cas des paramètres de l'expérimentation 1 (moyenne densité et moyenne mobilité).



(a) Nombre moyen de clusters.

(b) Le nombre de changements d'état cluster-head.

(c) Durée de vie moyenne d'un cluster.

(d) Le nombre de ré-affiliations des nœuds membres.

Figure 3.7 – Résultats de simulation de l'expérimentation 1.

La figure 3.7a représente le nombre moyen de clusters formés dans le réseau. Les courbes de la figure 3.7a montrent que l'algorithme HCC fournit la meilleure performance. Il génère le plus petit nombre de clusters. En effet, ce résultat est logique puisque l'algorithme HCC révise la structure des clusters à chaque changement de la topologie du réseau afin de réduire le nombre de clusters. Nous remarquons aussi que les performances des autres algorithmes suivent le même ordre concernant relaxation de la condition de renoncement des cluster-heads. En effet, en deuxième position vient l'algorithme LCC-hc qui définit un seul cas de figure pour lequel un cluster-head abandonne son rôle. En dernière position, nous trouvons l'algorithme α-SSCA puisqu'il ajoute une condition au seul cas de figure présenté par LCC-hc.

Pour le cas de α-SSCA, nous remarquons que lorsque la valeur du facteur α augmente le nombre de clusters formés augmente. Ceci est expliqué par le fait que lorsque α est grand la chance qu'un cluster-head satisfasse la condition de stabilité est plus importante. Ceci réduit l'abandon des cluster-heads et favorise la création d'autres clusters. Malgré que α-SSCA génère un nombre plus important de clusters par rapport aux autres algorithmes, ce nombre reste réduit par rapport au nombre total de nœuds dans le réseau. En effet, 1-SSCA déclare 15% des nœuds comme cluster-heads et $\frac{1}{2}$-SSCA fournit un rapport de 10%.

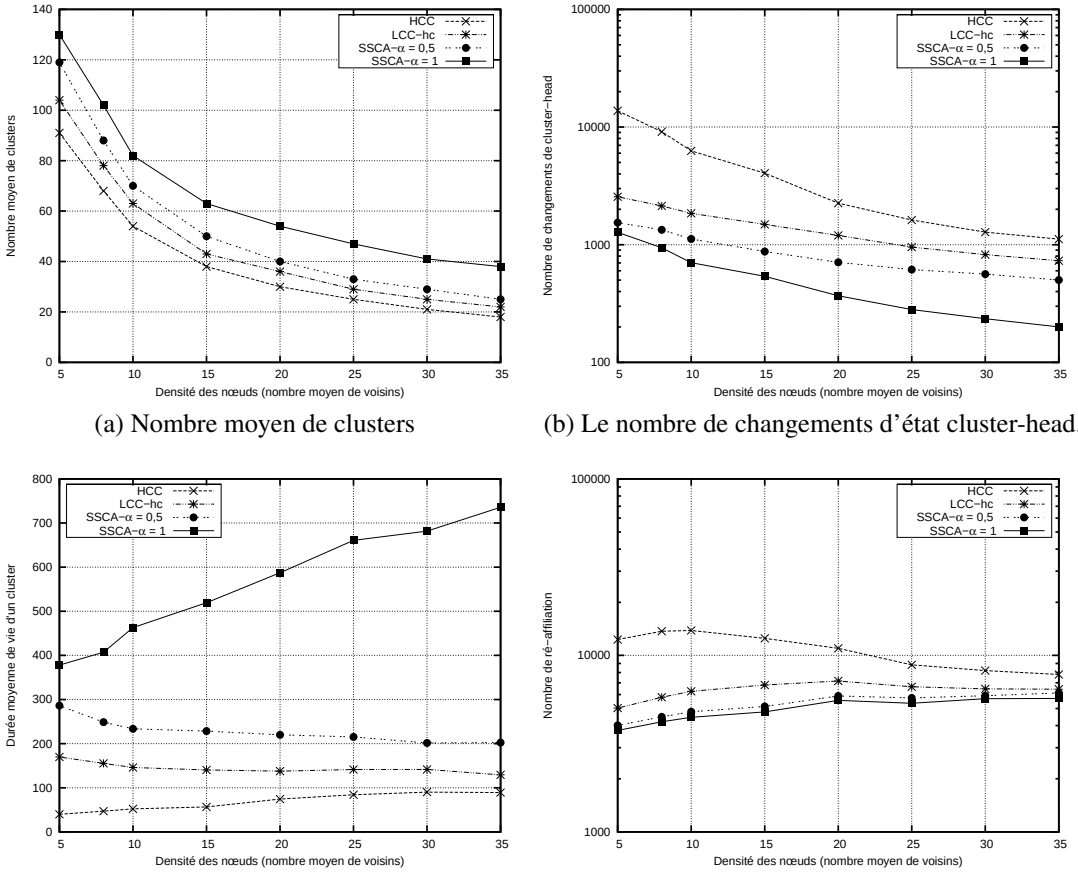
La figure 3.7b montre le nombre de changements de cluster-head tout au long de la durée de la simulation. Nous remarquons que l'ordre des courbes est l'inverse de l'ordre constaté dans la figure 3.7a représentant le nombre moyen de clusters. En effet, chaque fois que l'abandon des cluster-heads est plus relaxé, le nombre de changements de cluster-head est plus réduit. Ainsi, l'algorithme α-SSCA présente le nombre le plus réduit de changements de cluster-head. La version 1-SSCA fournit de loin la meilleure performance en terme de nombre de changements de cluster-head. Elle est suivie par la version $\frac{1}{2}$-SSCA. Les algorithmes LCC-hc et HCC présentent un nombre plus élevé de changements de cluster-head avec un avantage de l'algorithme LCC-hc.

La figure 3.7c présente la durée moyenne de vie d'un cluster. Nous remarquons que les courbes de cette figure gardent des valeurs presque constantes quand le nombre de nœuds augmente. Ceci nous permet de conclure que la durée moyenne de vie des clusters ne dépend pas du nombre de nœuds. Les valeurs enregistrées confirment bien les conclusions prises à propos du nombre de changements de cluster-head. En effet, l'algorithme α-SSCA qui présente le nombre de changements le plus réduit offre la meilleure durée de vie des clusters. Tandis que les autres algorithmes offrent une durée de vie dans le même ordre que le nombre de changements de cluster-head. Pour les paramètres utilisés dans cette expérimentation, nous remarquons que les valeurs de la durée de vie des clusters sont assez réduites surtout pour le cas des algorithmes HCC (de l'ordre de 30 secondes) et LCC-hc (de l'ordre de 50 secondes). Si l'algorithme de clusterisation est employé dans un protocole de routage, ces valeurs réduites vont nuire au fonctionnement du protocole. Chaque fois que le protocole converge, il rencontrera de nouveaux changements dans la structure des clusters. Ceci va réduire la validité des routes calculées et diminuer les performances du routage.

La figure 3.7d montre le nombre de ré-affiliations des nœuds membres. Dans cette figure, nous remarquons, excepté l'algorithme HCC qui produit un nombre important de changements des nœuds membres, que les valeurs enregistrées par les autres algorithmes sont très proches avec un léger avantage pour l'algorithme LCC-hc. En effet, deux facteurs interviennent dans ce résultat. La conservation d'un cluster va éviter que plusieurs membres de ce cluster soient obligés de changer de cluster. Ceci réduit normalement le nombre de ré-affiliations des nœuds membres. Mais en contre partie, lorsque les nœuds se déplacent assez rapidement, la conservation du cluster-head va aussi augmenter la probabilité que ces nœuds s'éloignent de leurs cluster-heads et augmente le nombre de ré-affiliations. La compensation de ces deux facteurs rend les valeurs du nombre de ré-affiliations assez proches pour les algorithmes α-SSCA et LCC-hc.

3.3.3 Variation de la densité des nœuds

Dans cette section nous allons étudier les performances des trois algorithmes de clusterisation en fonction de la densité des nœuds. La figure 3.8 présente les valeurs mesurées des différentes métriques pour le cas des paramètres de l'expérimentation 2.



(a) Nombre moyen de clusters

(b) Le nombre de changements d'état cluster-head.

(c) Durée de vie moyenne d'un cluster.

(d) Le nombre de ré-affiliations des nœuds membres.

Figure 3.8 – Résultats de simulation de l'expérimentation 2.

La figure 3.8a montre que le nombre de clusters formés dans le réseau diminue lorsque la densité des nœuds augmente. En effet, le nombre moyen des voisins des cluster-heads augmente ce qui fait augmenter à son tour la cardinalité des clusters. Par conséquent, ceci réduit le nombre total de clusters créés. Nous remarquons aussi que les algorithmes étudiés suivent le même ordre que la relaxation de la condition d'abandon et ceci pour toutes les valeurs de la densité des nœuds. Ainsi, HCC génère le nombre le plus réduit des clusters et il est suivi respectivement par LCC-hc, $\frac{1}{2}$-SSCA et 1-SSCA. Enfin, nous remarquons que 1-SSCA ne profite pas assez de la densité des nœuds. En effet, la différence entre cet algorithme et les autres algorithmes augmente considérablement lorsque la densité des nœuds augmente. Par exemple, comparé à LCC -hc, 1-SSCA présente une augmentation de 40% du nombre moyen de clusters formés pour le cas d'une densité égale à 10 nœuds. Tandis que pour une densité de 35 nœuds, il enregistre une augmentation de 95%. Ce résultat est valide par rapport aux deux autres algorithmes.

L'importante augmentation en terme de nombre de clusters formés est nettement récompensée en terme de stabilité de la structure des clusters. En effet, les figures 3.8b, 3.8c et 3.8d, montrent que l'algorithme 1-SSCA fournit la meilleure stabilité en termes de nombre de changements de cluster-head, de la durée moyenne de vie des cluster-heads et de nombre

de ré-affiliations. Nous remarquons également que le gain en stabilité fourni par 1-SSCA par rapport aux autres algorithmes s'accroît lorsque la densité des nœuds augmente.

3.4 Impact de la stabilité sur le routage

Dans cette section, nous étudions l'impact de la stabilité des clusters sur les protocoles de routage. Dans ce cadre nous avons intégré les quatre versions d'algorithmes de clusterisation comparés dans la section précédente dans un protocole de routage basique qui se base sur l'état de liens des clusters. Ensuite, nous comparons les performances du protocole de routage pour les différents algorithmes de clusterisation.

Nous commençons par décrire le protocole de routage envisagé. C'est un protocole de routage à état de liens classique (tel que OSPF [12]) mais qui sera appliqué à la topologie de clusters et non pas à la topologie des nœuds. Pour cela chaquenœud commence par découvrir son voisinage et applique les fonctions de création et de maintenance des clusters. Par la suite, seuls les cluster-heads sont invités à envoyer périodiquement et suite aux changements leurs état de liens dans des messages LSA (Link State Advertisement). Chaque cluster détecte alors les clusters voisins à travers son voisinage à 2-sauts. Il construit un message LSA qui contient ses membres et la liste de ses clusters voisins et le diffuse dans tout le réseau.

Chaque nœud du réseau collecte tous les LSA reçus des différents cluster-heads. À partir des LSA collectés, il construit la topologie du réseau formé par l'interconnexion de tous les clusters du réseau. L'algorithme de DIJKSTRA est alors employé pour calculer les distances vers tous les nœuds du voisinage à 2-sauts et vers tous les cluster-heads du réseau. Les paquets LSA sont aussi utilisés par chaque nœud pour déterminer le cluster auquel appartient chaque nœud du réseau. Cette localisation sera employée lors de l'envoi et de l'acheminement des paquets de données. En effet, lorsqu'un nœud désire envoyer un paquet de données vers une destination, il commence par chercher son cluster d'appartenance. Par la suite, il suit le même chemin calculé pour atteindre son cluster-head.

Dans la suite nous allons présenter les résultats des simulations effectuées afin de monter l'impact des algorithmes de clusterisation et de la stabilité sur ce protocole de routage basique. Les simulations sont effectuées en utilisant le simulateur NS-2. Nous avons choisi de représenter les résultats enregistrés pour le cas d'une densité moyenne des nœuds fixée à 20 nœuds. Les nœuds se déplacent avec une vitesse moyenne dont la valeur moyenne est de 2m/s. Dans cette partie nous avons introduit un considérable trafic de données. En effet, ce trafic correspond à 30 connections de type CBR (constant bit rate) de débit 2 Ko/s. Tous les paquets sont de taille 512 octets. Les sources et les destinations sont choisies aléatoirement parmi tous les nœuds du réseau. Notons aussi que toutes les connexions commencent et se terminent entre 30 et 350 secondes et que la durée de la simulation est fixée à 400 secondes.

Dans cette expérimentation, nous allons évaluer les versions du protocole de routage en

utilisant les métriques suivantes :

- ***Ratio des paquets délivrés :*** il représente le rapport entre le nombre de paquets reçus par les destinations finales et le nombre de paquets envoyés par les sources. Ce rapport est la plus importante métrique pour la comparaison des algorithmes de routage. En effet, elle exprime le degré de complétude et d'efficacité des protocoles de routage.
- ***Trafic de contrôle (Overhead) :*** représente la taille des informations explicites du processus de routage. Cette métrique est mesurée en comptant le nombre total des paquets de contrôle diffusés dans le réseau.
- ***Délai moyen de bout en bout :*** c'est le temps moyen passé par les paquets de données envoyés par les sources et qui ont atteint les destinations finales. Ce temps inclus le temps de transmission entre les différents nœuds intermédiaires ainsi que le temps passé dans les files d'attente de ces nœuds.

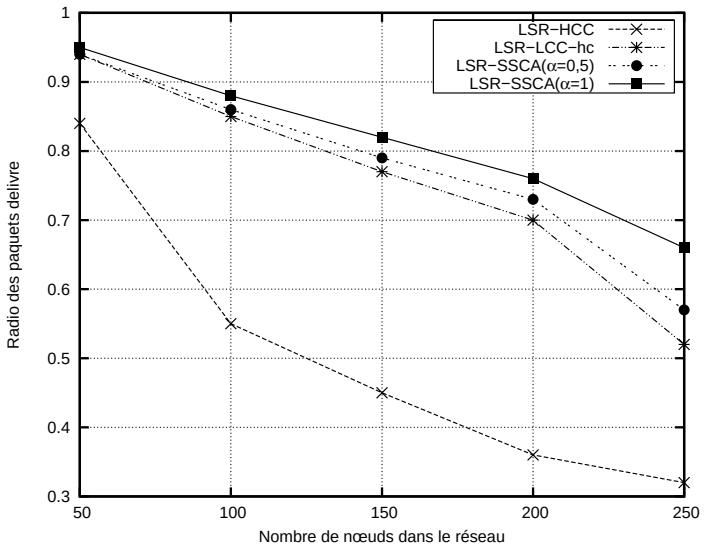


Figure 3.9 – Ratio des paquets délivrés.

La figure 3.9 représente le ratio des paquets délivrés par les quatre versions développées du protocole de routage. Nous remarquons que ce ratio diminue lorsque le nombre de nœuds augmente. Nous remarquons également que ce rapport dépend de l'algorithme de clusterisation. Nous observons que ce ratio augmente lorsque la stabilité de la structure des clusters augmente. En effet, l'utilisation de l'algorithme 1-SSCA, qui donne la structure la plus stable, permet de délivrer le plus grand nombre de paquets de données. Les courbes représentant les autres versions du protocole de routage suivent le même ordre de performance de la stabilité de la structure des clusters. Nous trouvons alors la version utilisant $\frac{1}{2}$-SSCA suivie de celle qui utilise LCC-hc et en dernière position la version utilisant l'algorithme HCC. Ce résultat est très lié à la validité des routes. En effet, les chemins que suivent les paquets de données sont calculés à partir de la structure des clusters. Chaque changement dans cette structure a comme effet de fausser temporairement certains che-

mins. La fréquence des changements fait augmenter le nombre de chemins non valides, et par la suite la perte des paquets augmente ce qui réduit la performance du protocole de routage en terme de ratio des paquets délivrés.

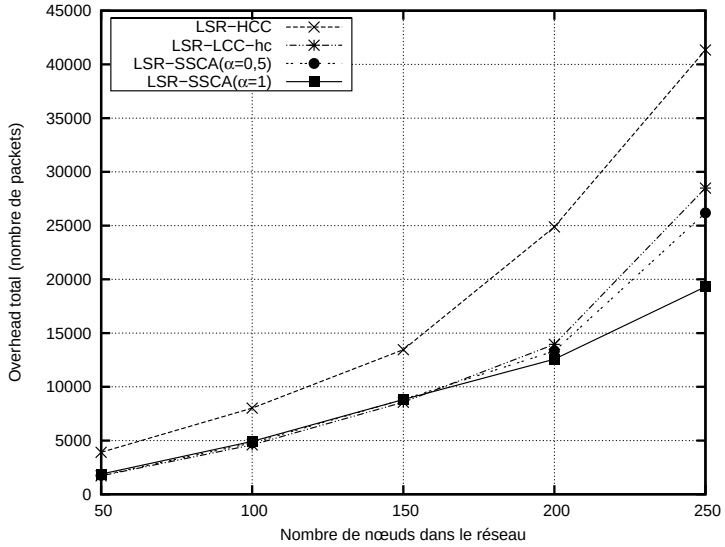


Figure 3.10 – Trafic de contrôle total (nombre de paquets).

La figure 3.10 illustre l'*overhead* généré par chaque version du protocole de routage. Cet *overhead* est comptabilisé comme le nombre de paquets de contrôle envoyés par le protocole de routage. Nous remarquons qu'à l'exception de la version utilisant HCC qui génère un important *overhead*, les autres versions génèrent des valeurs assez proches. Puisque toutes les versions utilisent les mêmes règles de routage, la différence entre ces versions réside dans le comportement des algorithmes de clusterisation. En effet, deux facteurs interviennent dans la taille de l'*overhead* généré. Puisque seuls les cluster-heads envoient périodiquement les messages d'état de liens, les algorithmes qui créent le nombre le plus réduit de cluster-heads vont générer un *overhead* plus réduit. Le deuxième facteur correspond aux messages de mise à jour envoyés lors des changements de la topologie des clusters. Par la suite, l'*overhead* sera plus réduit si l'algorithme de clusterisation offre une meilleure stabilité de la topologie des clusters. Les courbes de la figure 3.10 montrent que c'est le facteur de stabilité qui l'emporte surtout lorsque le nombre de nœuds devient important. Malgré que l'algorithme 1-SSCA crée un nombre plus important de cluster-heads, il arrive à générer l'*overhead* le plus réduit grâce à sa plus stable topologie.

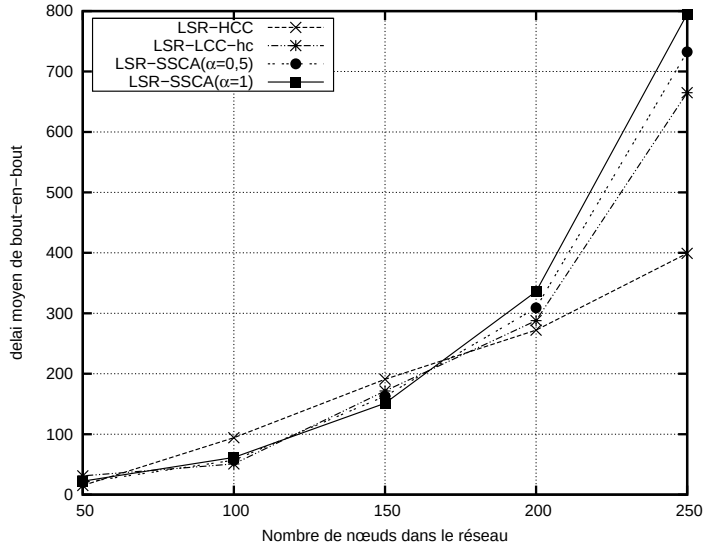


Figure 3.11 – Délai moyen de bout-en-bout.

La figure 3.11 illustre le délai moyen de bout-en-bout des paquets de données reçus par les destinations finales. Nous remarquons que l'algorithme 1-SSCA donne la valeur la plus importante du délai moyen. Sachant que les routes sont calculées en utilisant les mêmes algorithmes pour toutes les versions, et que 1-SSCA produit l'*overhead* le plus réduit et le plus important ratio de paquets délivrés, nous pouvons conclure que la différence ne réside pas dans le temps passé dans les files d'attente, mais dans le nombre de nœuds intermédiaires traversés. Ce résultat prouve que la version du routage utilisant α-SSCA est capable d'acheminé des paquets de données vers des destinations plus éloignées. Enfin, nous remarquons qu'une meilleure stabilité a comme effet d'augmenter les performances du protocole de routage à acheminer les paquets de données vers des destinations plus éloignées.

Nous concluons que l'amélioration de la stabilité de la topologie des clusters a pour effet d'améliorer les performances du protocole de routage à état de liens des clusters en termes de ratio de paquets délivrés, en taille d'*overhead* généré et de délai de bout-en-bout.

3.5 Conclusion

Dans ce chapitre, nous avons présenté notre proposition d'un algorithme de clusterisation α-SSCA qui peut être intégré dans un protocole de routage à état de liens. L'objectif de notre algorithme est de maintenir la structure des clusters aussi stable que possible au cours du temps. La stabilité de la structure des clusters permet de garantir la stabilité des routes et donc réduire le coût de leur maintenance. Notre algorithme α-SSCA est implémenté sous le simulateur des réseaux NS-2. Nous avons comparé les performances de notre algorithme α-SSCA avec HCC, LCC-hc. Les résultats des simulations montrent que notre algorithme α-SSCA a de meilleures performances comparé à HCC et LCC en termes de nombre de

changements de cluster-head et de durée moyenne de vie des cluster-heads. Nous avons également étudié l'impact de la stabilité des clusters sur les protocoles de routage. Pour cette fin, nous avons évalué les performances d'un protocole de routage à état de liens générique pour les différents algorithmes de clusterisation. Les résultats des simulations montrent que l'amélioration de la stabilité de la topologie des clusters a pour effet d'améliorer les performances du protocole de routage en termes de ratio de paquets délivrés, de taille d'*overhead* généré et de délai de bout-en-bout.

CHAPITRE 4

PROPOSITION D'UN ALGORITHME DE CLUSTERISATION GÉNÉRIQUE À K-SAUTS : SKCA

DANS LE CHAPITRE 3 nous avons proposé un nouveau algorithme de clusterisation à un saut. Dans le cadre des réseaux larges, l'utilisation d'un seul saut comme diamètre des clusters peut générer un nombre très important de clusters. Dans ce contexte, ce chapitre est consacré à la présentation d'une généralisation générique à K-sauts de l'algorithme SSCA présenté dans le chapitre 3.

Sommaire

4.1 Introduction 66
4.2 Présentation générale de l'algorithme SKCA 66
4.3 Les métriques utilisées par α-SKCA 68
4.4 La formation des clusters 68
4.4.1 Phase 1 : Premier tour d'élection des cluster-heads 70
4.4.2 Phase 2 : Deuxième tour d'élection des cluster-heads 71
4.5 La maintenance des clusters 73
4.6 Algorithmes couverts par SKCA 74
4.7 Analyse expérimentale 75
4.7.1 Impact du rayon de la zone de présélection (r) 76
4.7.2 Impact du rayon des clusters (K) 78
4.7.3 Impact de la taille la zone d'extension (ϵ) 79
4.7.4 Impact du seuil du facteur de stabilité (α) 81
4.7.5 Impact de la taille de la zone critique (ρ) 82
4.8 Conclusion 83

4.1 Introduction

Dans ce chapitre, nous allons proposer un nouvel algorithme de clusterisation générique nommé SKCA (Stability K-hops Clustering Algorithm). Ce nouvel algorithme est une généralisation combinée des deux algorithmes SSCA et 3HBAC à K-sauts. La contribution principale de l'algorithme proposé est d'améliorer la stabilité de la topologie des clusters quand la topologie du réseau change fréquemment tout en gardant assez réduit le nombre de clusters dans le réseau. SKCA reproduit l'idée de 3HBAC pour créer des clusters tout en respectant une distance minimale entre les cluster-heads. En effet, nous introduisons le statut de nœud *membre-invité* qui représente un nœud qui n'est pas à K-sauts d'aucun cluster-head mais il est à une distance de ϵ-sauts à un nœud membre d'un autre cluster. Cet état va garantir, lors de la formation des clusters, que tous les cluster-heads adjacents sont distants d'au moins $(K + \epsilon)$-sauts.

Pour améliorer encore la stabilité de la structure des clusters, nous employons aussi le facteur α-stabilité introduit dans notre proposition SSCA. Ce facteur diminue le renoncement des cluster-heads dans la phase de maintenance des clusters. En effet, quand deux cluster-heads sont à une distance entre $(K + \epsilon - \rho)$ et $(K + \epsilon)$ sauts, un parmi ces deux cluster-heads abandonne son rôle uniquement si la condition de stabilité n'est pas vérifiée.

SKCA est aussi indépendant de la métrique utilisée pour la sélection des cluster-heads. Il prévoit l'utilisation de n'importe quelle métrique proposée dans la littérature. Ainsi, une phase de déclaration et d'échange des valeurs de la métrique employée est requise. En effet, chaque nœud doit connaître les métriques des nœuds de son voisinage avant de pouvoir se décider de son rôle dans la structure des clusters à former. Étant donné que le coût de la diffusion de la métrique dans le voisinage à K-sauts est très important, SKCA prévoit deux tours d'élection des cluster-heads. Dans le premier tour, tous les nœuds participent à l'élection mais dans un voisinage réduit à r-sauts. Cette phase consiste à une présélection des nœuds éligibles à devenir cluster-heads. Seuls les nœuds vainqueurs du premier tour participent au second tour d'élection qui se déroule dans tout le voisinage à K-sauts. Dans ce deuxième tour, se fait la sélection des cluster-heads effectifs.

4.2 Présentation générale de l'algorithme SKCA

SKCA opère en deux phases. La première phase est la formation des clusters. Dans cette phase, chaque nœud collecte les informations requises pour élire les cluster-heads et se décide de son statut dans la structure des clusters (cluster-head ou membre). Dans la deuxième phase, il maintient la structure des clusters quand les nœuds se déplacent et la topologie du réseau change. Au cours de ces phases, l'algorithme SKCA utilise plusieurs paramètres. Les valeurs de ces paramètres permettent de répondre à plusieurs objectifs tels que réduire le nombre de cluster-heads ou encore améliorer la stabilité de la structure des

clusters. Il est aussi possible, de trouver un bon compromis entre plusieurs objectifs. Les paramètres globaux de l'algorithme SKCA sont les suivants :

- K : Il définit le rayon effectif d'un cluster. Il exprime la distance maximale qui peut séparer un cluster-head de l'ensemble de ses membres.
- r : Il définit le rayon du voisinage pour la présélection des cluster-heads.
- ϵ : Il détermine avec le paramètre K la distance minimale $(K + \epsilon)$ qui sépare deux cluster-heads adjacents lors de la formation des clusters. Ainsi, il précise, autour d'un cluster-head, une zone interdite pour la création de nouveaux clusters. Cette zone est considérée comme une zone d'extension du cluster.
- ρ : Il intervient lors de la phase de maintenance, et il délimite la distance entre deux cluster-heads adjacents pour laquelle la condition d'abandon sera appliquée.
- α : Il s'agit d'une valeur seuil du facteur de stabilité après laquelle un cluster-head doit abandonner son rôle.

Les paramètres K, r et ϵ interviennent dans la phase de formation des clusters tandis que les paramètres ρ et α interviennent pendant le processus de maintenance des clusters. Ces paramètres définissent quatre zones différentes autour de chaque cluster-head qui sont les suivantes :

- **Zone d'éligibilité** : Cette zone est formée par le voisinage à K-sauts de chaque cluster-head. Elle représente le rayon effectif du cluster. Par la suite, tous les nœuds distants à moins de K-sauts à un cluster-head u peuvent devenir des nœuds membres du cluster de u.
- **Zone d'extension** : Elle contient tous les nœuds distants plus que K-sauts et moins que $(K + \epsilon)$ du cluster-head. Ces nœuds ne sont pas autorisés à créer de nouveaux clusters afin de garantir la distance minimale $(K+\epsilon)$ entre les cluster-heads adjacents. S'ils ne sont pas voisins à aucun cluster-head à moins de K-sauts, ils sont autorisés à devenir des membres invités de ce cluster.
- **Zone interdite** : Elle représente le voisinage à $(K + \epsilon - \rho)$ sauts de chaque cluster-head. Dans cette zone, la présence de tout autre cluster-head n'est pas autorisée. Ainsi, si deux cluster-heads se trouvent à une distance inférieure à $(K + \epsilon - \rho)$ sauts, un de ces deux cluster-heads doit abandonner son rôle.
- **Zone critique** : Elle désigne le voisinage délimité par le nombre de sauts $(K + \epsilon - \rho)$ et $(K + \epsilon)$. Dans cette zone, seuls les cluster-heads qui ont un facteur de stabilité supérieur au seuil prédéfini α peuvent garder leur état cluster-head.

La figure 4.1 illustre les quatre zones définies autour des cluster-heads dans le cas où le paramètre K est fixé à trois sauts, le paramètre ϵ est fixé à un saut et le paramètre ρ est fixé à deux sauts. Pour plus de clarté, nous avons représenté deux zones autour de chaque cluster-head. Autour du cluster-head u, nous avons dessiné la zone d'éligibilité (zone A) et la zone d'extension (zone B). Tandis qu'autour du cluster-head v, nous avons dessiné la zone interdite (zone C) et la zone critique (zone D).

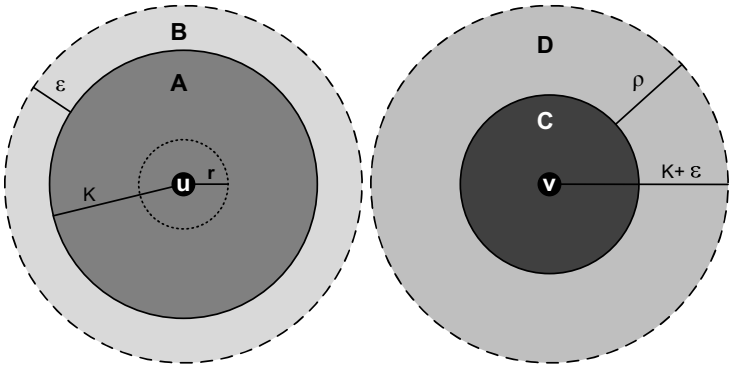


Figure 4.1 – Exemple des zones prédéfinies autour des cluster-heads.

4.3 Les métriques utilisées par α-SKCA

Dans la littérature, plusieurs métriques ont été proposées pour élire les cluster-heads. Comme la clusterisation en elle même ne représente pas un objectif, chaque métrique employée répond à un besoin ou à des contraintes liées au cadre de l'utilisation de l'algorithme. Dans notre proposition, nous allons présenter l'algorithme d'une façon indépendante de la métrique. SKCA suppose que chaque nœud connaît la valeur de sa métrique ainsi que celle des nœuds dans son voisinage. Cette connaissance est établie à travers un échange de messages entre les nœuds voisins.

Dans SKCA, les règles de la formation et de la maintenance des clusters utilisent deux fonctions :

- La fonction *score* est utilisée comme métrique pour élire les cluster-heads. Ainsi, les nœuds qui ont le meilleur *score* sont choisis comme cluster-heads. Dans la section 2.3.2 du chapitre 2 plusieurs métriques ont été présentées telles que l'identifiant du nœud, son degré de connectivité etc.
- Le fonction *affinité* est utilisée pour sélectionner le cluster. Quand un nœud est à K-sauts de plusieurs cluster-heads, il choisit de joindre le cluster-head ayant la meilleure fonction *affinité*. La cardinalité des cluster-heads ou la distance qui les séparent aux autres nds peuvent être utilisés comme fonction *affinité*.

4.4 La formation des clusters

Dans SKCA, la structure des clusters est formée par des transitions d'état de tous les nœuds du réseau. La figure 4.2 illustre les différentes transitions d'état d'un nœud. Chaque nœud peut être dans un des états suivants :

- ***non-décidé*** : Il s'agit de l'état initial des nœuds. Il indique qu'un nœud n'appartient à aucun cluster. Il est utilisé quand le nœud vient d'arriver au réseau ou vient de quitter son cluster.

- ***pré-cluster-head*** : Cet état indique que le nœud a été élu comme cluster-head potentiel après le premier tour d'élection. Il est choisi alors pour participer au deuxième tour d'élection.
- ***pré-membre*** : Cet état indique que le nœud a été battu lors du premier tour d'élection. Il est très probable qu'il deviendra un nœud membre ou invité.
- ***cluster-head*** : Il note que le nœud a créé un nouveau cluster et est devenu le chef de ce cluster.
- ***membre*** : Cet état dénote que le nœud a joint un cluster-head et qu'il appartient à son cluster.
- ***membre-invité*** : Le nœud s'invite à joindre le cluster d'un membre de son voisinage réduit.

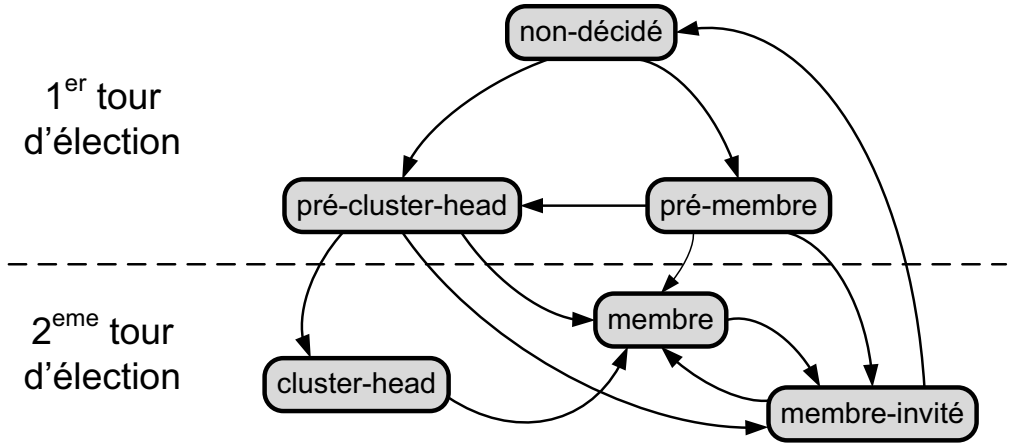


Figure 4.2 – Diagramme de transitions d'état.

Comme que nous l'avons déjà présenté, chaque nœud a besoin, pour se décider, de connaître les scores de ses voisins à K-sauts. Un échange de messages entre les voisins à K-sauts est alors requit avant l'activation du processus de clusterisation. Afin de minimiser le nombre de messages échangés entre les voisins, la sélection des cluster-heads se fait en deux tours d'élection. En effet, les nœuds ayant dans leurs voisinage réduit à r-sauts ($r <= K$) un ou plusieurs nœuds avec un meilleur score n'ont pas de chance à devenir des cluster-heads. Pour cette raison, ces nœuds ne participent pas au deuxième tour d'élection. Par la suite, la réduction du nombre de participants dans l'élection des cluster-heads à K-sauts réduit considérablement le nombre de messages échangés.

Le premier tour correspond à une pré-sélection des cluster-heads potentiels dans un voisinage réduit à r-sauts. Tous les nœuds participent à ce premier tour. Ils diffusent alors leur score seulement dans à leur voisinage réduit à r-sauts. Les nœuds ayant le meilleur *score* dans ce voisinage réduit passent à l'état pré-cluster-head, les autres nœuds passent à l'état pré-membre.

Le deuxième tour correspond à l'élection effective des cluster-heads. Seuls les nœuds pré-cluster-heads vainqueurs du premier tour participent à ce deuxième tour. Ils diffusent leur score à tous leurs voisins à K-sauts. Les nœuds ayant les meilleurs *scores* dans leur voisinage à K-sauts sont élus comme cluster-heads.

Nous remarquons, que dans le cas où le voisinage réduit coïncide avec le rayon des clusters ($r = K$), les nœuds choisis pendant le premier tour d'élection correspondent aux cluster-heads effectifs qui seront sélectionnés pendant le deuxième tour. Dans ce cas, un seul tour est suffisant pour élire l'ensemble des cluster-heads. Par la suite, SKCA court-circuite le premier tour d'élection dans le cas où $r = K$.

4.4.1 Phase 1 : Premier tour d'élection des cluster-heads

Ce tour correspond à une pré-sélection des futurs cluster-heads. Tous les nœuds prédisent leur rôle dans la structure des clusters en tenant compte des informations de leur voisinage à r-sauts. Avant de se décider de son rôle dans la structure des clusters, chaque nœud dans le réseau doit collecter des informations à propos de son voisinage à r-sauts. Des messages HELLO contenant les informations relatives au processus de clusterisation seront diffusées périodiquement à tous les voisins à r-sauts (avec $TTL = r$). Ensuite, chaque nœud entre en concurrence avec tous ses voisins à r-sauts pour devenir cluster-head. Après ce tour, un nœud peut être un cluster-head potentiel s'il a la meilleure fonction *score*. Autrement, il devient un nœud membre potentiel. L'ensemble des règles suivantes est appliqué par chaque nœud pour se décider de son futur rôle :

Règle 4.1 Atteinte d'éligibilité locale

Un nœud ayant la meilleure fonction *score* dans son voisinage à r-sauts passe à l'état pré-cluster-head.

Règle 4.2 Abandon de la concurrence

Un nœud distant à moins de r-sauts d'un ou plusieurs pré-cluster-heads passe à l'état pré-membre.

Règle 4.3 Attente de priorité

S'il n'y a pas de pré-cluster-head dans le voisinage à r-sauts et s'il existe un ou plusieurs voisins à l'état initial possédant un meilleur *score*, le nœud doit attendre jusqu'à ce que ses voisins se décident de leur rôle.

Règle 4.4 Déblocage de l'attente

S'il n'y a pas de pré-cluster-head dans le voisinage à r-sauts et si tous les voisins avec un meilleur *score* sont décidés, le nœud devient pré-cluster-head.

Chaque changement d'état est suivi de l'émission d'un message HELLO ce qui permet d'informer les voisins à r-sauts de cette décision et d'accélérer la convergence de l'algorithme.

4.4.2 Phase 2 : Deuxième tour d'élection des cluster-heads

Tous les pré-cluster-heads élus au premier tour entrent dans une compétition pour être choisis comme étant des cluster-heads effectifs. En effet, chaque nœud pré-cluster-head diffuse sa déclaration avec un message HELLO à tous ses voisins à K-sauts (avec $TTL = K$). Il maintient une liste des pré-cluster-heads concurrents dans son K-sauts voisinage. La formation des clusters est réalisée à travers les règles suivantes :

Règle 4.5 Création d'un nouveau cluster

Un nœud pré-cluster-head qui a le meilleur *score* dans son voisinage à K-sauts, devient un cluster-head.

Règle 4.6 Affiliation à un cluster

Tous les nœuds qui sont à l'état pré-cluster-head ou pré-membre et qui sont présents dans le voisinage à K-sauts d'un cluster-head u deviennent des nœuds membres du cluster de u. Dans le cas où il existe plusieurs cluster-heads dans le voisinage à K-sauts, le nœud s'attache au cluster-head ayant la meilleure *affinité*.

Règle 4.7 Adhésion à un cluster en mode invité

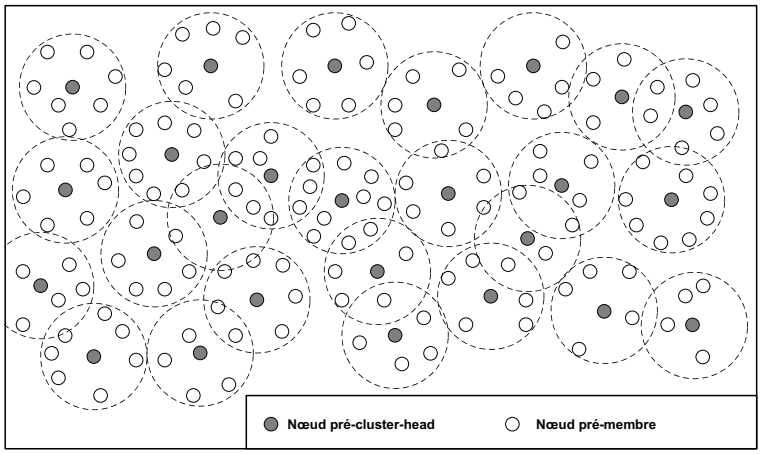
Si aucun cluster-head n'est présent dans son voisinage à K-sauts et s'il existe dans son voisinage réduit à r-sauts un nœud u à l'état membre, le nœud choisi ce nœud u comme point d'accès au cluster et il se déclare un membre-invité du cluster de u. Si plusieurs clusters sont accessibles, le nœud choisit son point d'accès avec la meilleure fonction *affinité*.

Règle 4.8 Création contrainte d'un nouveau cluster

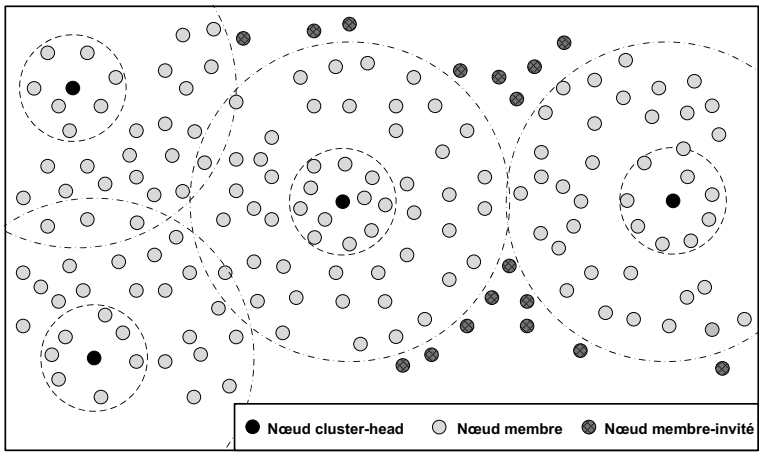
Si tous les nœuds pré-cluster-heads ayant un meilleur *score* dans le voisinage à K-sauts d'un nœud pré-cluster-head sont décidés et aucun d'eux n'est un cluster-head, ce nœud devient un cluster-head.

Règle 4.9 Préparation d'une création contrainte d'un nouveau cluster

S'il n'existe aucun nœud cluster-head dans son voisinage à K-sauts et aucun nœud membre dans son voisinage à r-sauts, et si tous les nœuds avec un meilleur *score* dans le voisinage sont décidés, un nœud pré-membre passe à l'état pré-cluster-head.



(a) Sélection des pré-cluster-heads au premier tour



(b) Structure finale des clusters après le deuxième tour

Figure 4.3 – Exemple de formation des clusters ($K = 3$, $r = 1$, $\epsilon = 1$).

Quand un nœud pré-cluster-head devient un nœud cluster-head, il doit informer ses voisins à K-sauts de sa décision. Il envoie alors sa déclaration de cluster-head par un message

HELLO. Quand un nœud pré-cluster-head devient un nœud membre, il doit aussi informer ses voisins à K-sauts de sa décision à travers l'envoi d'un message HELLO. Tandis qu'un nœud pré-membre informe seulement ses voisins à r-sauts s'il devient un nœud membre.

La figure 4.3 illustre les deux tours d'élection des cluster-heads. La figure 4.3a présente le résultat après le premier tour d'élection. Les nœuds en blanc sont battus et abandonnent la concurrence pour devenir cluster-head. Ceux en gris sont élus en tant que pré-cluster-heads et ils sont les seuls à concourir le rôle cluster-head dans le deuxième tour. La figure 4.3b expose le résultat de cette concurrence ainsi que la structure finale des clusters obtenue après le deuxième tour.

4.5 La maintenance des clusters

Cette section décrit les situations qui apportent des changements à la structure des clusters. Elle présente les différents changements d'état possibles lors des mouvements des nœuds du réseau. Elle inclut le renoncement du rôle cluster-head, la création de nouveau cluster, ou la ré-affiliation et le changement des états membre et membre-invité dans un même cluster. Les mécanismes de maintenance utilisés sont simples et ils ont comme objectif de réduire les changements dans la structure des clusters.

Règle 4.10 Perte de lien : nœud membre

Quand un nœud membre devient à une distance supérieure à K-sauts de son cluster-head :

1. Si aucun cluster-head n'est présent dans son voisinage à K-sauts et s'il existe parmi ses voisins à r-sauts un ou plusieurs membres dans le même cluster, il devient un membre-invité dans son cluster.
2. Autrement, il quitte son cluster et tente de rejoindre un autre cluster en appliquant les mêmes règles de la formation des clusters.

Règle 4.11 Détection d'un nouveau lien

Quand un nœud membre-invité détecte la présence d'un cluster-head dans son voisinages à K-sauts, il devient membre de son cluster.

Règle 4.12 Perte de lien : nœud membre-invité

Quand la distance entre un nœud membre-invité et son point d'accès dépasse le ϵ-sauts :

1. S'il existe parmi ses voisins à ϵ-sauts un ou plusieurs membres de son cluster, il reste membre-invité du même cluster et change son point d'accès.
2. Sinon et s'il existe parmi ses voisins à ϵ-sauts un ou plusieurs membres d'un autre cluster, il reste à l'état membre-invité mais il change de cluster et de point d'accès.
3. Autrement, il crée un nouveau cluster et devient cluster-head.

Règle 4.13 Rapprochement de deux cluster-heads : zone interdite

Si deux cluster-heads se trouvent à une distance inférieure à $(K + \epsilon - \rho)$-sauts, le cluster-head ayant le *score* le plus faible renonce à son rôle de cluster-head.

Règle 4.14 Rapprochement de deux cluster-heads : zone critique

Si deux cluster-heads se trouvent à une distance entre $(K + \epsilon - \rho)$ et $(K + \epsilon)$, le cluster-head avec le plus faible *score* calcule son facteur de stabilité *Fs* avec le deuxième cluster-head. Si ce facteur est inférieur à α, il renonce son rôle.

Quand un cluster-head abandonne son rôle, il doit déclarer sa décision à travers la diffusion d'un message HELLO dans son voisinage à K-sauts. Il revient avec l'ensemble de ses membres à l'état non-décidé. Ainsi, tous les nœuds à l'état non-décidé recommencent le processus de formation des clusters.

4.6 Algorithmes couverts par SKCA

SKCA est un algorithme générique multi-paramètres. En fixant les valeurs des différents paramètres et en choisissant certaines métriques, l'algorithme SKCA coïncide avec plusieurs algorithmes déjà proposés dans la littérature. Cette coïncidence peut être parfaite pour certains algorithmes, ou elle peut être partielle et elle peut être alors complété moyennant quelques règles supplémentaires. Parmi ces algorithmes nous citons comme exemple les algorithmes LCC-hc, SSCA, 3HBAC et K-CONID. Le tableau 4.1 décrit l'ensemble des valeurs des différents paramètres pour quelques exemples d'algorithmes couverts par SKCA.

Paramètres	LCC-hc	3HBAC	SSCA	K-CONID
Score	HC	HC	HC-ND	HC
Affinité	HC	–	Card	Dist(CH)
K	1	1	1	qqc
r	1	1	1	K
ϵ	0	1	0	0
α	0	1	qqc	0
ρ	0	1	1	0

Table 4.1 – Valeurs des paramètres de quelques algorithmes couverts par SKCA

4.7 Analyse expérimentale

L'algorithme SKCA a été conçu pour être assez générique, ainsi il peut répondre à plusieurs besoin. Il permet d'émuler le comportement de plusieurs algorithmes existants. De ce fait, nous avons envisagé d'étudier l'effet de chacun de ces paramètres sur ses performances, mais sans le comparer à d'autres algorithmes. En effet, cette comparaison est implicite dans le cas ou les valeurs des paramètres coïncident avec un autre algorithme.

Afin d'analyser expérimentalement les performances de l'algorithme SKCA, il a été implémenté sous le simulateur NS-2. Dans les expériences qui seront présentées dans cette section, nous utilisons les mêmes modèles décrits dans la section 3.3. Le tableau 4.2 résume l'ensemble des paramètres lié à la génération des scénarios. Nous rappelons que la densité des nœuds exprime le nombre moyen de nœuds contenus dans une surface d'un cercle de rayon égale à la porté de transmission du support radio. Elle présente aussi le nombre moyen de voisins directs d'un nœud du réseau.

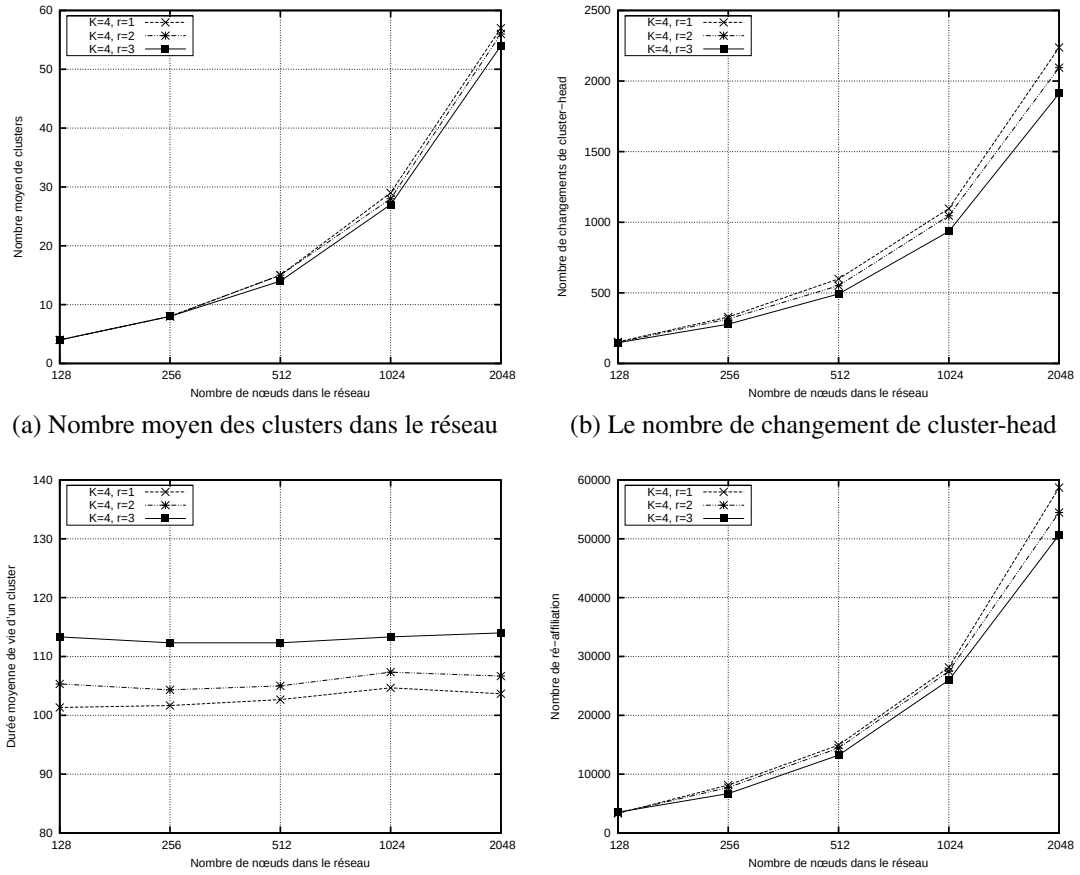
Paramètres globaux	
Couche liaison	IEEE 802.11
Durée de simulation	1000 secondes
Bande passante	2 Mb/s
Portée	250m
Densité des nœuds	10 nœuds
Mobilité	
Modèle de mobilité	RWP
Vitesse des nœuds	[0-5] m/s
Temps de pause	[0-50] s
Trafic de données	
Aucun trafic	

Table 4.2 – Paramètres de la simulation

Dans cette évaluation, nous avons choisi de mesurer les même métriques utilisées dans l'analyse de l'algorithme SSCA (section 3.3) et qui sont les suivantes :

- Le trafic de contrôle de l'algorithme (*overhead*).

- Le nombre moyen de cluster-heads dans le réseau.
- Le nombre de changements d'état cluster-head.
- La durée de vie moyenne d'un cluster.
- Le nombre ré-affiliations des nœuds membres.



(a) Nombre moyen des clusters dans le réseau

(b) Le nombre de changement de cluster-head

(c) Durée moyenne de vie du cluster

(d) Le nombre de ré-affiliation

Figure 4.4 – Résultats de simulation en fonction du paramètre r.

4.7.1 Impact du rayon de la zone de présélection (r)

La présélection des cluster-heads est un nouveau mécanisme introduit par l'algorithme SKCA. Dans cette partie, nous analysons les effets de la variation du rayon de la zone de présélection r. Nous avons fixé le rayon des clusters K à 4 sauts. Nous avons aussi écarté la zone d'extension et la zone critique en fixant les paramètres ϵ, ρ et α à zéro. Nous avons varié le nombre de nœuds dans le réseau pour les valeurs du rayon r {1, 2, 3}.

La figure 4.4 montre que les performances enregistrées en termes du nombre de clusters générés et de la stabilité de la structure des clusters sont très proches. Nous remarquons aussi que l'augmentation du rayon de la zone de présélection des cluster-heads améliore légèrement ces performances. En effet, l'augmentation du rayon de présélection réduit le nombre de pré-cluster-heads concurrent au rôle cluster-head. Cette réduction a comme effet

d'augmenter la distance moyenne qui sépare ces pré-cluster-heads. Par conséquence, ceci augmente les distances entre les clusters adjacents. Cette conduite augmente la surface dominée par chaque cluster-head ce qui réduit le nombre de clusters (figure 4.4a). Ce même facteur permet aussi de retarder le rapprochement des cluster-heads. Par la suite, il augmente la durée de vie des clusters (figure 4.4c) et réduit le nombre de changements d'état cluster-head (figure 4.4b). Enfin, la réduction du nombre de renoncement a comme effet de réduire le nombre de ré-affiliations des nœuds membres (figure).

Malgré que l'augmentation du rayon de présélection améliore les performances de l'algorithme SKCA, cette augmentation a un coût énorme en termes de nombre de messages explicites ajoutés par l'algorithme. La figure 4.5 montre le trafic de contrôle (*overhead*) produit par l'algorithme SKCA pour les trois valeurs du rayon de présélection r. L'*overhead* généré par la présélection à un seul saut est largement le plus réduit.

En conclusion, la réduction du rayon de la zone de présélection permet de réduire considérablement le nombre de messages de contrôle pour des performances comparable en termes de nombre de clusters dans le réseau et de la stabilité des clusters.

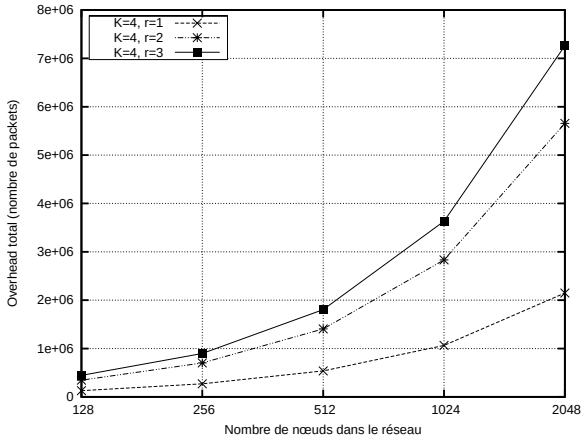


Figure 4.5 – Trafic de contrôle en fonction du paramètre r.

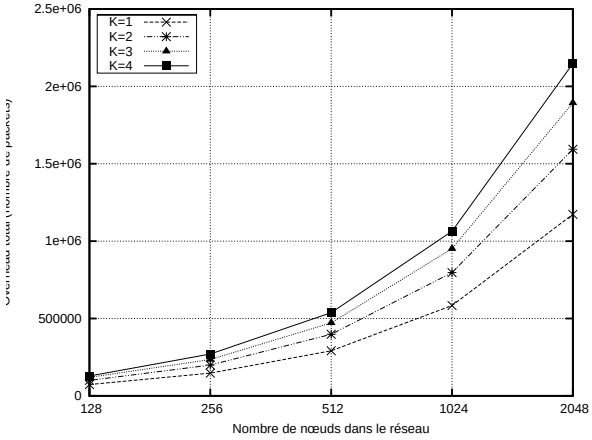
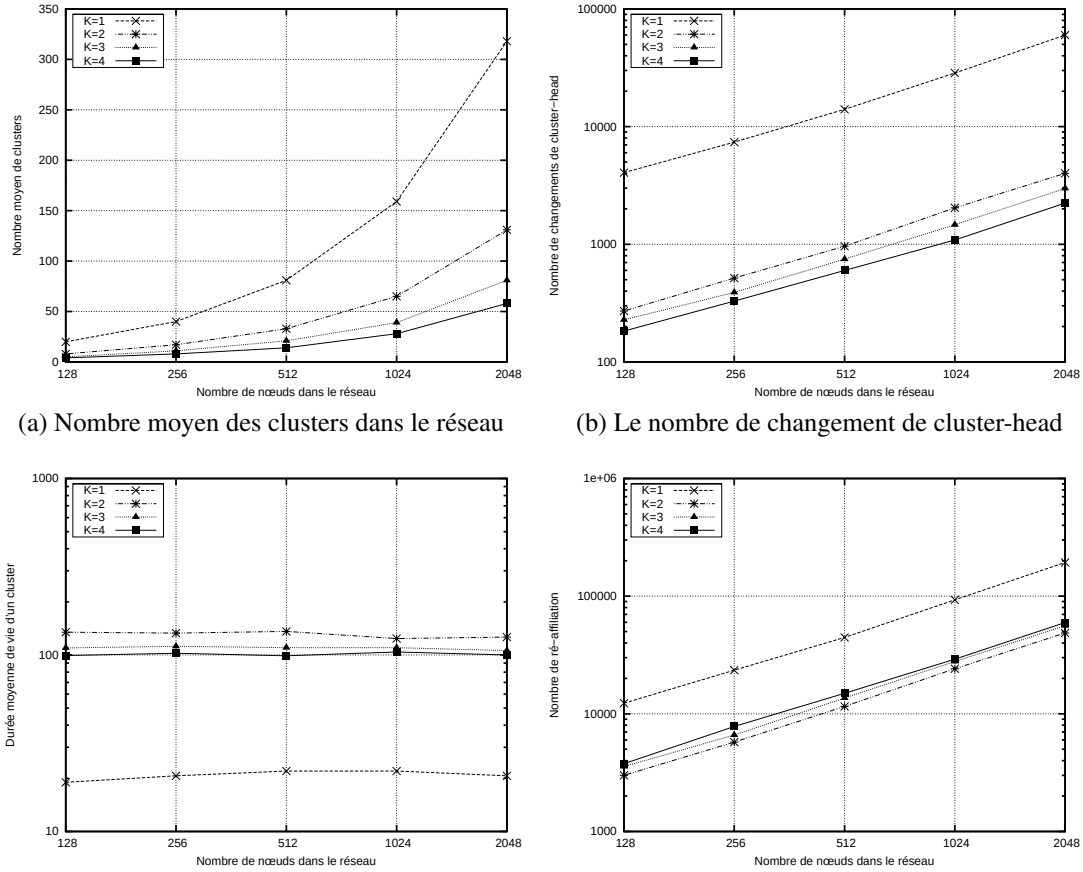


Figure 4.6 – Trafic de contrôle en fonction du paramètre K.

4.7.2 Impact du rayon des clusters (K)



(a) Nombre moyen des clusters dans le réseau

(b) Le nombre de changement de cluster-head

(c) Durée moyenne de vie du cluster

(d) Le nombre de ré-affiliation

Figure 4.7 – Résultats de simulation en fonction du paramètre K.

Dans cette section, nous allons examiner l'effet de l'élargissement de la taille des clusters. Pour cela, nous avons fixé le paramètre ϵ, α et ρ à zéro et nous utilisons une zone de présélection de rayon égale à un saut. Enfin, nous mesurons les métriques d'évaluation en fonction du nombre de nœuds pour les cas d'un rayon de cluster K égale à 1, 2, 3 et 4 sauts.

La figure 4.6 illustre le trafic total de contrôle pour chaque valeur du rayon des clusters K. Nous remarquons que lorsque le rayon des clusters augmente le trafic de contrôle augmente. Ce résultat est logique puisque les messages HELLO sont diffusés dans le voisinage à K-sauts sachant que le coût de diffusion sans optimisation est au moyenne $(K-1)^2 \times d$(d est la densité des nœuds). Par contre, nous remarquons que l'augmentation n'est pas assez importante par rapport au coût énoncé de la diffusion dans un voisinage à K-sauts. En effet, l'utilisation du mécanisme de présélection à un saut réduit considérablement le nombre de nœuds qui diffusent leurs messages HELLO à K-sauts. Par la suite, ceci réduit considérablement le trafic de contrôle lorsque la taille des clusters augmente.

L'agrandissement du rayon des clusters élargit la surface couverte par chaque cluster. Ainsi, le nombre de clusters dans le réseau sera diminué proportionnellement aux surfaces

des clusters. La figure 4.7a illustre cette observation. Elle montre que l'élargissement des clusters abaisse considérablement le nombre de clusters générés. L'élargissement de la couverture des clusters réduit aussi le nombre de création de nouveaux clusters. En plus, l'augmentation du rayon des clusters a comme effet d'éloigner les cluster-heads adjacents. Ce qui réduit le nombre de renoncement des cluster-heads. Ces raisons expliquent le résultat exposé par la figure 4.7b. Cette figure montre que le nombre de changements d'état cluster-head est réduit à chaque fois que le diamètre des clusters est agrandi.

Enfin, la figure 4.7c présente la durée de vie moyenne d'un cluster en fonction du nombre de nœuds dans le réseau. Nous remarquons que la durée de vie d'un cluster relatif à une clusterisation à un saut est assez réduite par rapport aux autres valeurs de K. Nous remarquons aussi que les valeurs de la durée de vie pour les cas où K est égal à 2, 3 et 4 se rapprochent. Pour ces trois valeurs, nous observons que la durée de vie moyenne diminue quand le rayon des clusters est plus grand. Ce résultat paraît incompatible avec le résultat obtenu du nombre de changements d'état cluster-head. Mais, il peut être concilié par le fait que l'augmentation du rayon des clusters entraine l'élargissement des frontières des clusters. Ces frontières présentent un endroit favorable pour des créations contraintes de nouveaux clusters par les nœuds qui ne peuvent pas appartenir à aucun cluster voisin. Étant donné que ces clusters sont très proches des clusters voisins, leur durée de vie est très réduite, puisque le déplacement de leur cluster-head induit fort probable à se trouver rapidement dans une zone couverte par un cluster-head voisin. Ceci peut expliquer la dégradation de la durée de vie moyenne des clusters pour les cas d'un grand rayon des clusters. Les même constations sont aussi observés dans les résultats du nombre de ré-affiliations des nœuds membres représentés dans la figure 4.7d.

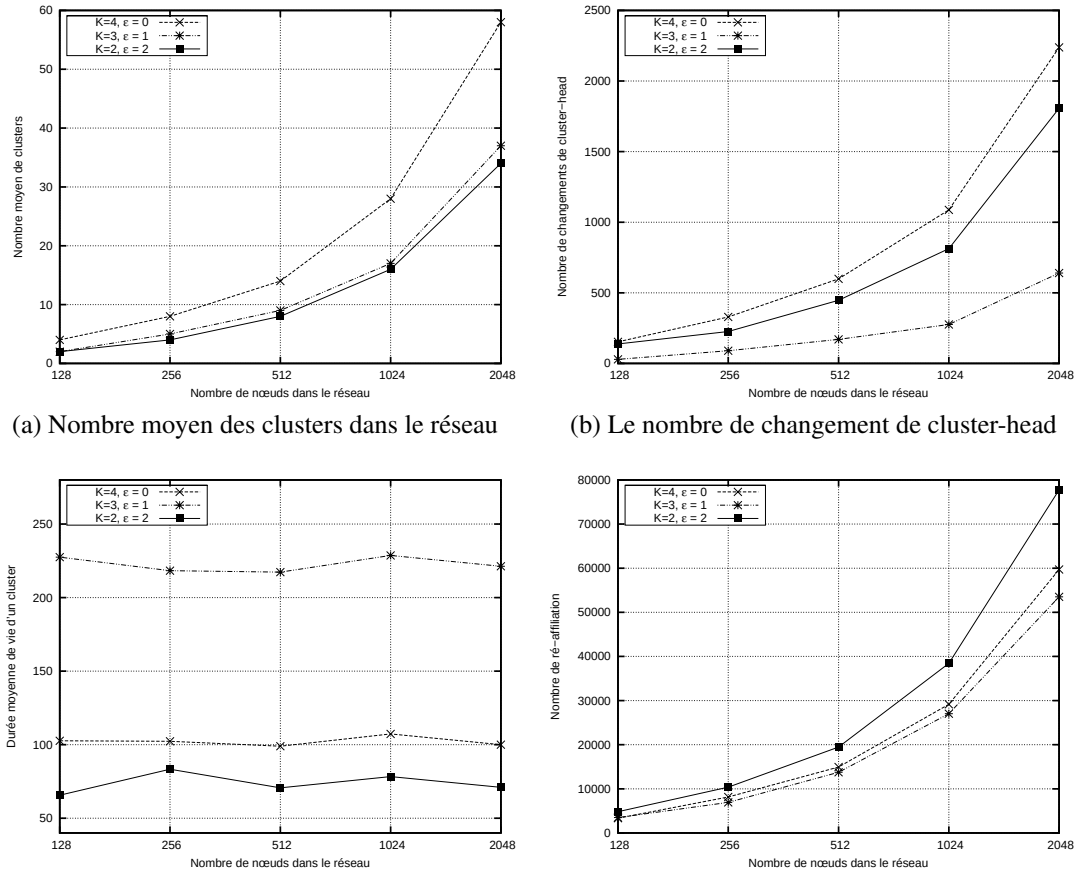
4.7.3 Impact de la taille la zone d'extension (ϵ)

Le paramètre ϵ indique la taille de la zone d'extension d'un cluster. Il intervient dans les deux phases formation et maintenance des clusters. Dans la phase de formation des clusters il garantit que tous les clusters adjacents sont à une distance supérieure à $(K + \epsilon)$-sauts. Il permet aussi de réduire le nombre de création contrainte de cluster par les nœuds aux frontières des clusters existants. En effet, ces nœuds ont la possibilité de joindre les clusters voisins en tant que membre invité. Pour analyser l'effet de cette zone d'extension, nous avons choisi de garder constante la superficie totale des clusters de rayon $K + \epsilon$. Par la suite, nous avons réalisé des simulations pour les valeurs {(4,0), (3,1), (2,2)} du couple (K, ϵ). Dans cette expérimentation, nous avons fixé le paramètre r à un saut et les paramètres ρ et α à zéro.

L'introduction du paramètre ϵ permet d'éloigner et de bien organiser les cluster-heads sur la surface du réseau. Ceci permet d'améliorer leur couverture, ce qui réduit le nombre de cluster-heads dans le réseau. La figure 4.8a illustre cette observation. Elle montre que le nombre de clusters diminue lorsque ϵ augmente. Nous observons aussi une importante

réduction du nombre de clusters pour les cas où $\epsilon = 1$ et $\epsilon = 2$ par rapport au cas où $\epsilon = 0$. Nous remarquons aussi que les valeurs obtenus pour les cas où $\epsilon = 1$ et $\epsilon = 2$ sont très proches.

La figure 4.8b expose le nombre de changements d'état cluster-head en fonction du nombre de nœuds dans le réseau. Nous remarquons que l'introduction d'état membre-invité permet d'améliorer la stabilité des clusters. Par contre, nous remarquons que le cas où $\epsilon = 1$ produit un nombre de changement plus réduit que celui où $\epsilon = 2$. En effet, l'augmentation de la taille de cette zone d'extension augmente la probabilité de création contrainte des nouveaux clusters aux frontières des clusters existants. Ces clusters sont assez proches des clusters adjacents. Leur mouvement éventuel peut rapidement les placés dans les zones interdites des cluster-heads voisins. Ceci induit à leurs disparitions rapides et cause alors un nombre important de changements d'états. Ce même reproche explique les résultats inscrits de la durée de vie moyenne d'un cluster et du nombre de ré-affiliations des nœuds membres. La 4.8d expose la durée moyenne de vie d'un cluster. Nous remarquons que le cas où $\epsilon = 1$ produit nettement la meilleure durée de vie. Par contre, le cas où $\epsilon = 2$ produit les plus faibles valeurs. Les mêmes constations sont observées dans la figure 4.8d qui présente le nombre de ré-affiliations des nœuds membres.



(a) Nombre moyen des clusters dans le réseau

(b) Le nombre de changement de cluster-head

(c) Durée moyenne de vie du cluster

(d) Le nombre de ré-affiliation

Figure 4.8 – Résultats de simulation en fonction du paramètre ϵ.

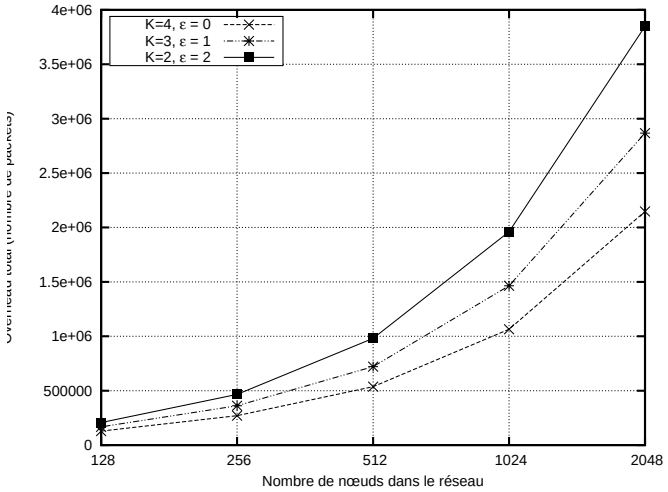


Figure 4.9 – Trafic de contrôle en fonction du paramètre ϵ.

En conclusion, l'utilisation de l'état membre-invité permet d'améliorer les performances de l'algorithme de clusterisation en termes de nombre de cluster-head générés et de la stabilité de la structure des clusters. Cependant, la taille ϵ de la zone de sélection des nœuds membre-invité ne doit pas être importante par rapport au rayon du cluster. Sinon, elle aura un effet inverse sur la stabilité des clusters.

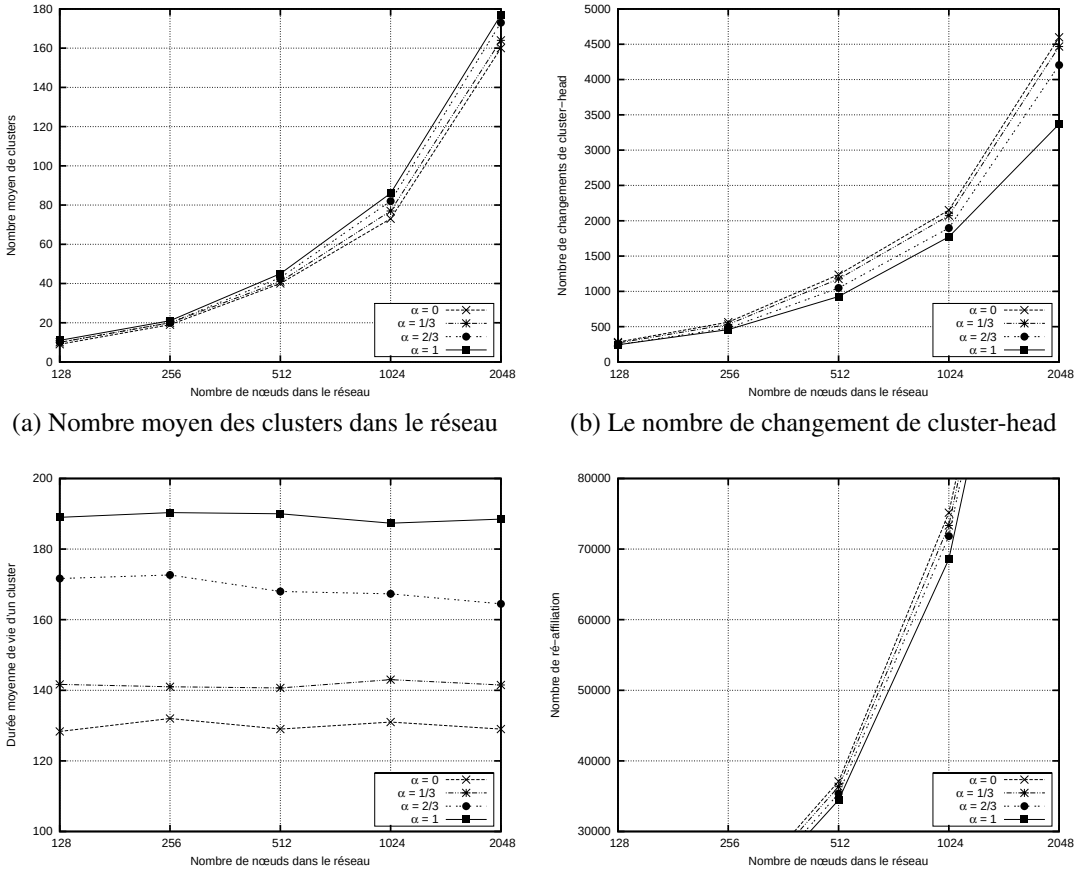
4.7.4 Impact du seuil du facteur de stabilité (α)

Dans ce cas d'expérimentation, nous avons varié le seuil α du facteur de stabilité pour lequel un cluster-head doit renoncer à son rôle. Les scénarios sont générés pour un rayon de cluster $K = 3$ et un rayon de présélection $r = 1$. Les paramètres ϵ et ρ sont fixés respectivement aux valeurs un et deux sauts.

L'élévation du seuil α a un effet direct de réduire le nombre d'abandon de cluster-head. Ceci réduit le nombre de changements d'état cluster-head. Par contre, cette conservation va augmenter le nombre de création de nouveaux clusters ce qui agrandit le nombre moyen de clusters dans le réseau. Ces deux résultats sont visibles dans les figures 4.10a et 4.10b. La figure 4.10a montre que le nombre moyen de clusters dans le réseau croît légèrement lorsque la valeur du seuil α est élevé. La figure 4.10b montre un gain plus important en termes de réduction de nombre de changements du rôle cluster-head pour les valeurs les plus grands du seuil α. La conservation des cluster-heads a aussi comme effet d'augmenter la durée de vie des clusters. Ce résultat est observé dans la figure 4.10c qui montre que la durée de vie moyenne d'un cluster augmente avec l'augmentation du seuil de renoncement α.

La figure 4.10d montre le nombre de ré-affiliation des nœuds membres en fonction du nombre de nœuds. Dans cette figure, nous remarquons que les valeurs du nombre de changements sont très proches pour les quatre valeurs envisagées du paramètre α. Pour plus de clarté, nous avons effectué un zoom sur les deux valeurs 512 et 1024 nœuds. Pour

ce cas d'expérimentation, nous remarquons aussi que le nombre de ré-affiliations est réduit lorsque le seuil α est augmenté. L'augmentation du paramètre α conduit à une plus longue conservation des clusters existants. Cette conservations permet, d'un côté, d'éviter la ré-affiliation des nœuds membres dans le cas de renoncement de leur cluster-head. D'un autre côté, lorsque les nœuds se déplacent assez rapidement, la conservation des cluster-heads va aussi augmenter la probabilité que ces nœuds s'éloignent de leur cluster-head et quittent leur cluster. L'équilibre établi entre ces deux facteurs explique les valeurs proches du nombre de ré-affiliations pour les cas testés.



(a) Nombre moyen des clusters dans le réseau

(b) Le nombre de changement de cluster-head

(c) Durée moyenne de vie du cluster

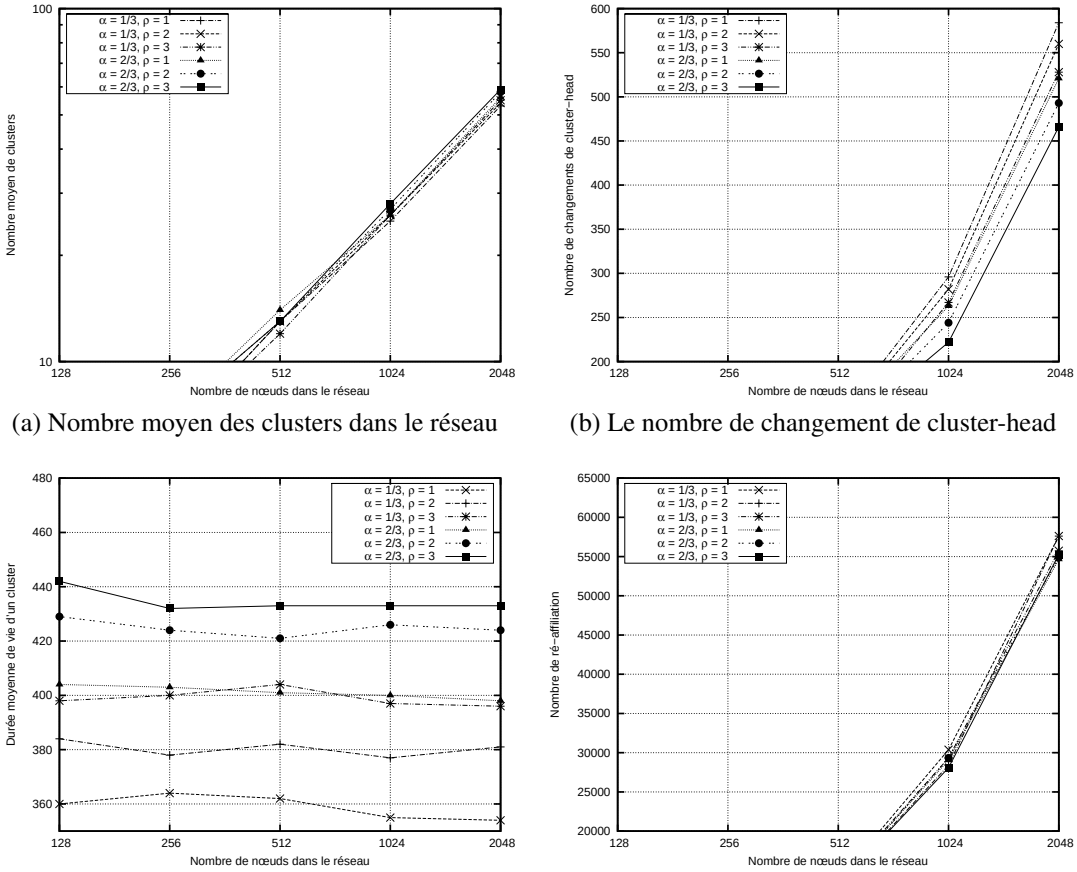
(d) Le nombre de ré-affiliation

Figure 4.10 – Résultats de simulation en fonction du paramètre α.

4.7.5 Impact de la taille de la zone critique (ρ)

Comme le seuil α du facteur de stabilité, le paramètre ρ agit au niveau de la condition d'abandon des cluster-heads. Il définit la taille de la zone critique dans laquelle cette condition est appliquée. Les courbes de la figure 4.11 illustrent les résultats obtenus lors de la variation de la taille de la zone critique pour deux valeurs du seuil α. Ces résultats sont obtenus pour un rayon de cluster $K = 3$, pour un rayon de présélection $r = 1$ et pour une zone d'extension de taille un saut(ϵ).

L'augmentation du paramètre ρ élargie de la zone critique. En même temps elle réduit, autour d'un cluster-head, la zone interdite aux autres cluster-heads. Cette réduction augmente la probabilité de conserver plus longtemps les cluster-heads existants. D'où, elle permet de réduire le nombre d'abandon de cluster-heads. Par la suite, l'augmentation de la taille de la zone critique a les mêmes effets que l'augmentation du seuil α du facteur de stabilité. Ces effets sont l'augmentation du nombre moyen de clusters (figure 4.11a), la réduction de nombre de changements de l'état cluster-head (figure 4.11b), l'augmentation de la durée de vie moyenne d'un cluster (figure 4.11c) ainsi que l'abaissement du nombre de ré-affiliations (figure 4.11d).



(a) Nombre moyen des clusters dans le réseau

(b) Le nombre de changement de cluster-head

(c) Durée moyenne de vie du cluster

(d) Le nombre de ré-affiliation

Figure 4.11 – Résultats de simulation en fonction du paramètre ρ.

4.8 Conclusion

Notre proposition SKCA d'un algorithme générique de clusterisation a été détaillée dans ce chapitre. SKCA est un algorithme à K-sauts qui a comme objectif de former et maintenir une structure stable des clusters. L'originalité de cet algorithme est révélée à travers plusieurs aspects. Tout d'abord, SKCA introduit une nouvelle notion, la pré-sélection, qui a comme objectif de réduire le nombre de messages échangés entre les voisins. En se-

cond lieu, SKCA utilise plusieurs paramètres dans son fonctionnement. Les valeurs de ces paramètres permettent de répondre à plusieurs objectifs tels que la réduction du nombre de cluster-heads ou l'amélioration de la stabilité de la structure des clusters. Il est alors possible de trouver un compromis entre ces objectifs en fixant les valeurs des paramètres. Enfin, l'algorithme SKCA est indépendant du choix de la métrique d'élection du cluster-head ou d'affiliation à un cluster. L'analyse expérimentale de SKCA, faite par simulation sous NS-2, a étudié le comportement de cet algorithme à partir de l'impact de ses différents paramètres. Cette étude a montré que l'augmentation du rayon de pré-sélection r améliore la stabilité des clusters mais engendre un trafic croissant d'overhead. Pour le rayon des clusters K, plus il augmente, plus le nombre de clusters et les changements de cluster-head diminuent. Cependant, l'augmentation de K améliore la durée de vie des cluster-heads puis elle le dégrade. Lorsque la taille la zone d'extension ϵ augmente, le nombre de cluster-heads diminue et l'overhead augmente. Pour le paramètre seuil du facteur de stabilité α, l'effet de son augmentation prolonge la durée de vie des clusters mais en contre partie fait augmenter le nombre de clusters. Ce même effet est observé pour la paramètre de taille de la zone critique ρ.

DEUXIÈME PARTIE

Le routage dans les réseaux mobiles sans-fil

CHAPITRE 5

ÉTAT DE L'ART DES PROTOCOLES DE ROUTAGE

LE ROUTAGE est une fonction fondamentale dans les réseaux. Elle a été largement visée par les travaux de recherche ce qui a produit un nombre important de protocoles divers. À la base d'une étude des protocoles proposés, nous avons distingué un ensemble de critères de classification. Nous avons présenté également les détails de certains protocoles de routage et nous avons particulièrement ciblé les protocoles qui ont visé le problème de scalabilité.

Sommaire

5.1 Introduction 89
5.2 Classification des protocoles de routage unicast 89
5.2.1 Structuration du réseau 90
5.2.2 Découverte de routes 91
5.2.3 Calcul des routes 91
5.2.4 Alternance des routes 92
5.2.5 Stockage des données 92
5.2.6 Circulation des informations de contrôle 93
5.2.7 Acheminement des données 93
5.2.8 Sélection des routes 94
5.2.9 Maintenance des routes 94
5.3 Protocoles de routage réactifs 95
5.3.1 Le protocole AODV 95
5.3.2 Le protocole DSR 97
5.3.3 Le protocole DYMO 99
5.3.4 Autres protocoles réactifs 100
5.4 Protocoles de routage proactifs 101
5.4.1 Le protocole DSDVR 101
5.4.2 Le protocole OLSR 101

5.4.3 Autres protocoles proactifs 103
5.5 Protocoles de routage hybrides 104
5.5.1 Le protocole CBRP 104
5.5.2 Le protocole ZRP 106
5.6 Scalabilité du routage à état de liens 107
5.6.1 Approche plate : le protocole F-OLSR 107
5.6.2 Approches basées sur une structure des clusters 107
5.6.3 Approches hiérarchiques 113
5.7 Conclusion 115

5.1 Introduction

Dans un réseau auto-organisable, un nœud ne peut communiquer directement qu'avec les nœuds dans sa portée. Lorsqu'un nœud veut communiquer avec un autre nœud se trouvant hors de sa portée, il est indispensable que ses voisins coopérent dans l'acheminement des paquets. La recherche du chemin d'accès entre deux nœuds du réseau est assurée par la fonction de routage. Cette fonction est d'autant plus complexe dans les réseaux mobiles puisque la topologie du réseau est constamment modifiée. Le routage dans les réseaux mobiles auto-organisables est un problème fondamental. Vu son importance et sa complexité, ce problème est le plus visé dans la recherche dans ce domaine. Pour le résoudre plusieurs protocoles ont été proposés.

Dans ce chapitre, nous présentons d'abord une classification des protocoles de routage unicast pour les réseaux auto-organisables. Ensuite, nous présentons les protocoles les plus connus, proposés pour effectuer le routage. Nous décrivons leurs principales caractéristiques et fonctionnalités qui permettent d'assurer l'acheminement des données entre les différentes unités mobiles. Notons que la plupart de ces protocoles sont en cours d'évaluation par des groupes de travail spécialisés dans les environnements mobiles. Nous nous intéréssons particulièrement au problème de scalabilité des protocoles de routage. Dans cet axe, nous étudions dans la dernière partie de ce chapitre des propositions récentes de protocoles de routage scalables.

5.2 Classification des protocoles de routage unicast

Plusieurs protocoles ont été proposés pour résoudre le problème de routage dans les réseaux auto-organisables. Vu le grand nombre et la diversité de ces protocoles, plusieurs travaux de recherche ont proposé des classifications de ces protocoles. Parmi ces travaux nous pouvons cité [Feeney, 1999, Jayakumar and Gopinath, 2007, Lee et al., 2006, Maqbool and Peer]. Dans cette partie, nous allons proposé une taxonomie des protocoles de routage selon divers critères. De même que les travaux de [Lee et al., 2006], la taxonomie présentée dans ce rapport s'appuie sur les fonctionnalités de base des protocoles de routage. Cette taxonomie, représentée dans la figure 5.1, classe les protocoles d'abord par le niveau de participation des nœuds dans le routage.

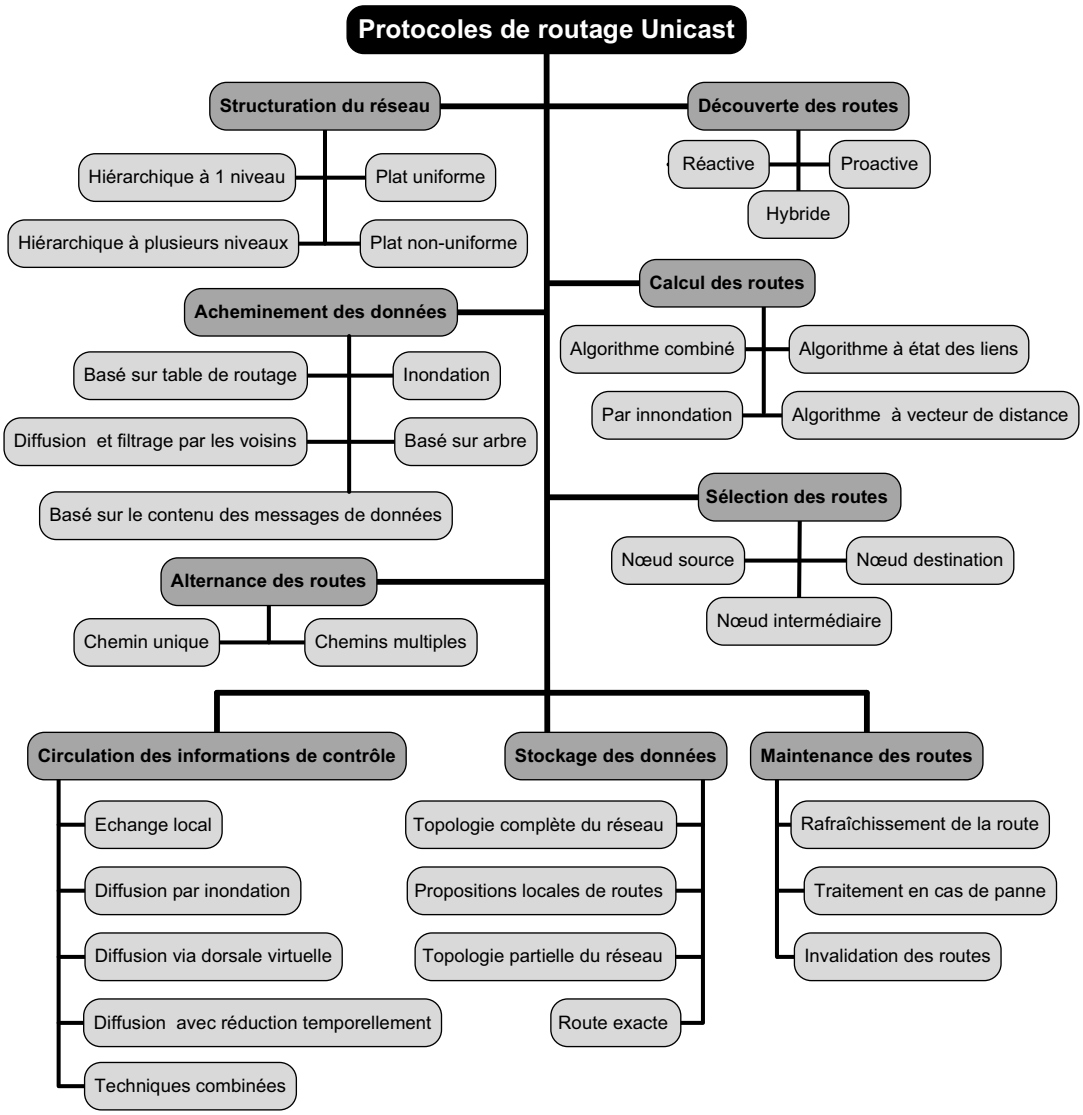


Figure 5.1 – Taxonomie des protocoles de routage unicast.

5.2.1 Structuration du réseau

Dans les réseaux auto-organisables tous les nœuds doivent coopérer pour assurer l'acheminement des données. Les protocoles de routage peuvent se distinguer par le nombre et la façon avec laquelle les nœuds participent aux différentes fonctions assurées par le protocole. Un protocole est dit uniforme si tous les nœuds participent dans les fonctions de routage et toutes les liaisons sont considérées dans le calcul des routes. Dans un protocole uniforme tous les nœuds fonctionnent de la même façon et il n'y a pas de nœuds ayant des fonctions particulières. Si les nœuds ne participent pas de la même façon dans le protocole de routage nous parlons d'un protocole non-uniforme. Un protocole est dit plat si

une route est maintenue pour atteindre une seule destination. Si une route peut être utilisée pour atteindre un groupe de nœuds le protocole est dit hiérarchique. Nous notons que les protocoles hiérarchique se basent sur une structure de groupe donc ils sont par la suite non-uniformes. Parmi les protocoles hiérarchiques ceux qui se basent sur un seul niveau de groupes de nœuds et d'autres qui prévoient un groupement des groupes. Ainsi, nous avons distingué quatre classes de protocoles :

- Plat-uniforme
- Plat-non-uniforme
- Hiérarchique à 1 seul niveau
- Hiérarchique à plusieurs niveaux

5.2.2 Découverte de routes

Chaque nœud d'un réseau auto-organisable joue le rôle d'un routeur et participe à l'acheminement des données. De nombreux débats et travaux ont discuté quand et combien de routes chaque nœud doit maintenir dans sa table de routage. En effet, un nœud doit-il mémoriser les routes pour atteindre toutes les nœuds destinations, ou seulement chercher et garder la trace des routes lorsqu'il a un intérêt d'atteindre les nœuds destinations. Par conséquent, trois types de protocoles de routage se sont distingués suivant la manière dont les routes sont maintenues dans les tables de routage.

- Proactive : Les nœuds maintiennent une route vers chaque nœud du réseau.
- Réactive : Les nœuds maintiennent seulement les routes actives (les routes qui sont utilisées).
- Hybride : Les protocoles hybrides utilisent un mélange des deux précédents types.

5.2.3 Calcul des routes

Certains algorithmes de routage ont été défini très tôt mais restent des méthodes solides et éprouvées. Ils sont encore utilisés aujourd'hui et gardent une influence très importante sur les travaux réalisés dans ce domaine.

- **Par inondation :** Le routage par inondation est probablement la méthode de routage la plus triviale. Chaque routeur recevant un paquet le réémet sur toutes les interfaces si le routeur n'est pas la destination du paquet.
- **Routage à vecteur de distance :** Le routage par vecteur de distance se base sur l'algorithme de Bellman-Ford distribué. Chaque routeur possède une table de routage qui consiste en un couple de données pour chaque destination : le routeur par lequel passer pour atteindre cette destination et le coût associé selon une métrique définie. Ces informations sont transmises périodiquement à tous les voisins, et donc chaque routeur reçoit ces informations de ses voisins. Il en résulte que, pour un routeur, toute destination existant dans la table de routage d'un voisin devient connue et donc

accessible par ce routeur.

- **Routage à état de liens :** Le principe de base du routage à état de liens est que la connaissance de la topologie complète du réseau permet de calculer aisément une route de toute source vers toutes les destinations. Pour découvrir ses voisins, un routeur envoie un message HELLO sur chacune de ses interfaces. Ensuite, chaque routeur rassemble l'ensemble des informations qu'il possède sur ses voisins et sur les coûts des liens qui le relient à ceux-ci dans un paquet d'état de liens, et transmet ce paquet en diffusion à tout le réseau. Une fois la topologie du réseau est connue, et celle-ci étant couplée avec l'ensemble des coûts de liens existants dans la topologie, un routeur dispose de toutes les informations nécessaires pour le routage. Il met en application l'algorithme de Dijkstra [Dijkstra, 1959] et détermine le plus court chemin vers tous les autres routeurs existants dans le réseau. Ce plus court chemin définit alors la route à utiliser.
- **Routage combiné :** Pour assurer la fonction de routage, certains protocoles combinent les techniques précédentes.

5.2.4 Alternance des routes

Un protocole de routage peut maintenir un seul chemin qui lui permet d'atteindre une destination particulière. Il peut également maintenir plusieurs chemins permettant d'atteindre une seule destination. Nous distinguons ainsi deux classes de protocoles :

- Chemin unique
- Chemins multiples

5.2.5 Stockage des données

Les protocoles de routage se distinguent aussi par les connaissances de la topologie maintenues par chaque nœud du réseau. Chaque nœud participant à l'acheminement des données doit stocker certaines informations qui lui permettent de déterminer les bons chemins et d'aiguiller les données vers le nœud suivant du chemin emprunté. Selon ce critère nous avons discerné quatre classes de protocoles :

- **Topologie complète du réseau :** Chaque nœud doit maintenir un graphe correspondant à la topologie complète du réseau.
- **Topologie partielle du réseau :** Dans cette classe, chaque nœud maintient une connaissance partielle de la topologie globale du réseau.
- **Propositions locales de routes :** Chaque nœud enregistre seulement des informations locales qui correspondent aux informations de la topologie locale ou des propositions de routes des nœuds appartenant au voisinage du nœud.
- **Route exacte :** Dans ces protocoles, les nœuds sont portés à enregistrer seulement les routes exactes découvertes. Ils enregistrent tous les nœuds appartenant aux chemins

qui permet d'atteindre une destination donnée.

5.2.6 Circulation des informations de contrôle

Pour permettre de découvrir les chemins assurant le transport de données dans le réseau, les nœuds participant au processus de routage doivent échanger certaines informations. Ces messages circulés entre les nœuds peuvent être des messages d'état de liens, des propositions de routes, des informations de structuration ou bien des requêtes de découverte de routes. Ces messages sont envoyés entre des nœuds séparés par différentes distances et en utilisant différents mécanismes. Nous avons repéré cinq classes d'algorithmes.

- **Échange local d'information :** Les informations de routage sont diffusées localement entre voisins.
- **Diffusion par inondation à l'aveugle :** Les informations de routage sont diffusées dans tout le réseau par inondation. Chaque nœud retransmet les messages de routage reçus vers ses voisins.
- **Diffusion via une dorsale virtuelle :** Les informations de routage sont aussi diffusées dans tout le réseau mais en utilisant une méthode de diffusion optimisée. Une structure composée d'un sous ensemble de nœuds couvrant le réseau entier est construite. Seuls les nœuds appartenant à cette structure sont autorisés à retransmettre les messages de routage reçus.
- **Diffusion avec réduction temporellement :** Les messages de routage sont aussi diffusés dans tout le réseau mais avec une fréquence variable. En effet, ces messages sont envoyés plus fréquemment aux nœuds proches qu' à ceux éloignées.
- **Technique combinée :** Plusieurs propositions combinent deux ou plusieurs des techniques précédentes.

5.2.7 Acheminement des données

Les messages de données sont envoyés d'un nœud à un autre jusqu'à atteindre la destination finale du message. Chaque nœud du chemin doit choisir un nœud de son voisinage pour lui délivrer la tâche d'acheminer ces données. Cette décision est prise en utilisant différentes techniques :

- **Basé sur table de routage :** Lorsqu'il y a des données à transmettre, un nœud regarde les entrées de sa table de routage pour connaître le nœud suivant dans le chemin à suivre.
- **Basé sur un arbre :** Le réseau est organisé sous forme d'arbre. Si la destination est présente dans l'une des branches d'un fils, le nœud intermédiaire retransmet les données vers ce fils. Sinon, il retransmet ces données vers son parant dans l'arbre.
- **Basé sur le contenu des messages de données :** Les nœuds participant à l'acheminement des données se basent sur le contenu du message pour faire aiguiller le

message. En effet, le chemin complet est stocké dans le message de données.

- **Diffusion et filtrage par les voisins :** Le message de données est diffusé vers tous les voisins directs du nœud. Seul un sous ensemble de ces nœuds retransmet le message jusqu'à atteindre la destination finale.
- **Par inondation :** Le message est inondé dans tout le réseau ce qui permet à la destination finale de recevoir au moins une copie de ce message.

5.2.8 Sélection des routes

La sélection de la route est le processus qui permet de choisir la ou les meilleures ou optimales routes à une destination parmi les routes potentielles. Bien que la plupart des protocoles choisissent seulement une meilleure route (routage à chemin unique), les protocoles multi-chemins sélectionnent et sauvegardent plus qu'un chemin pour des objectifs différents, tels que l'énergie, l'équilibre de trafic, la fiabilité et la robustesse. Les entités impliquées dans le processus de sélection peuvent inclure le nœud source, les nœuds intermédiaires ou le nœud destination.

- Le nœud source : Le nœud destination répond généralement à la première demande de route reçue. Cependant, par souci de qualité de chemin, le nœud destination peut répondre à toutes les demandes de route reçues. Par conséquent, c'est le nœud source qui est responsable de sélectionner finalement le meilleur chemin parmi les réponses multiples reçues de route.
- Le nœud destination : La destination sélectionne le meilleur chemin parmi les multiples demandes de routes reçues en se basant sur une métrique du coût prédéfinie et renvoie seulement une réponse de route à la source le long du chemin sélectionné. Le nœud source utilisera simplement le chemin envoyé par le nœud destination. Dans le cas du routage à chemins multiples, le nœud destination pourrait répondre à plus qu'une demande de route et le nœud source les acceptera toutes.
- Le nœud intermédiaire : Les nœuds intermédiaires jouent un rôle très important dans la sélection du chemin, bien qu'ils ne prennent pas la dernière décision de choisir les meilleurs chemins. Leur tâche majeure dans la sélection du chemin est de filtrer les demandes de route afin d'éviter les émissions excessives des demandes de route.

5.2.9 Maintenance des routes

La maintenance de la route est le processus qui permet de maintenir la route en cours d'utilisation valide et de la réparer si elle n'est plus disponible. Contrairement aux réseaux filaires où la cause principale de la maintenance est la panne des nœuds ou de liens, les réseaux sans fil exigent une maintenance plus complexe à cause de la mobilité des nœuds et les caractéristiques spécifiques du support de transmission. La maintenance dans le routage comprend en général le rafraîchissement de la route, le traitement en cas de panne et

l'invalidation des routes.

- Rafraîchissement de la route : Le rafraîchissement de la route essaye de confirmer et de maintenir valide les routes actuellement utilisées moyennant plusieurs approches. Par exemple, dans les protocoles réactifs, la source peut maintenir un chemin par l'envoi de paquet de contrôle ou par mise à jour automatique suite à l'expiration de la durée de vie du chemin.
- Traitement en cas de panne : La panne des routes peut être traitée de façon réactive, proactive ou hybride. Dans l'approche réactive, quand le prochain saut n'est pas accessible, la source ou un nœud intermédiaire essaie de trouver un nouveau chemin ou un chemin alternatif à la destination. L'approche proactive compte essentiellement sur le processus de rafraîchissement de route. L'information du chemin est mise à jour et est échangée à travers le réseau entier. L'approche hybride combine les deux approches réactive et proactive. Par exemple, chaque nœud met à jour de façon proactive les routes vers ses voisins et de façon réactive pour découvrir tous les chemins à une destination via un voisin.
- Invalidation des routes : L'invalidation de la route consiste à retirer toutes les routes à la suite d'une panne. Ce processus permet d'éliminer l'entrée associée à la destination dans les tables de routage.

5.3 Protocoles de routage réactifs

5.3.1 Le protocole AODV

AODV (Ad hoc On-demand Distance Vector)[Perkins and Belding-Royer, 1999] est un protocole décrit dans la RFC 3561 [Perkins et al., 2003]. Ce protocole crée les routes au besoin et utilise le principe de numéro de séquence afin d'utiliser les routes les plus nouvelles, dites encore les plus fraîches. En plus, il utilise le nombre de sauts comme métrique pour choisir entre plusieurs routes disponibles. Trois types de paquets sont utilisés par AODV : les paquets de requête de route RREQ (Route Request Message), les paquets de réponse de route RREP (Route Reply Message) et les paquets d'erreur de route RERR (Route Error Message). En plus de ces paquets, AODV invoque des paquets de contrôle HELLO qui permettent de vérifier la connectivité des routes. AODV repose sur deux mécanismes : découverte de route et maintenance de route. La découverte de route permet de trouver une route pour atteindre une destination et la maintenance de route permet de détecter et signaler les coupures de routes provoquées éventuellement par la mobilité des nœuds.

5.3.1.1 Découverte de voisinage

Lorsqu'un nœud veut émettre un message à une destination pour laquelle il ne possède pas de route valide dans sa table de routage, il diffuse une requête de route RREQ. Cette

requête RREQ contient un identifiant de la requête, le nombre de sauts parcourus, ainsi que le numéro de séquence de la source et le dernier numéro de séquence connu de la destination. Le numéro de séquence de la destination est recopié de la table de routage, s'il n'est pas connu, la valeur nulle sera prise par défaut.

Quand un nœud reçoit la requête RREQ, il vérifie s'il a déjà traité cette requête. Si c'est le cas, le nœud doit l'ignorer et arrêter le traitement. Dans le cas contraire, il enregistre la requête dans sa table historique pour rejeter les futurs doublons. Si le nœud ne connaît pas la route vers la destination : il incrémente le nombre de sauts et rediffuse la requête de route. Le nœud sauvegarde l'adresse du nœud précédent et celle du nœud source. Ces informations seront utilisées pour construire le chemin inverse, qui sera traversé par le paquet de réponse de la route (cela veut dire qu'AODV supporte seulement les liens symétriques).

Quand la requête atteint la destination ou un nœud qui possède une route récente (avec un numéro de séquence de la destination dans sa table supérieur ou égal au numéro de séquence dans le paquet RREQ), un paquet de réponse RREP est envoyé à la source. Cette réponse contient l'adresse du nœud initiateur de la requête, l'adresse de la destination de la requête, le numéro de séquence de la destination de la requête, le nombre de sauts parcourus par la requête. La réponse RREP est émise à destination de l'initiateur de la requête sur la route inverse de celle parcourue par la requête.

Quand un nœud reçoit une réponse de route RREP, le paquet est examiné, et une entrée pour la route vers la destination est inscrite dans la table de routage si au moins une des ces conditions est satisfaite :

- aucune route vers la destination n'est connue.
- le numéro de séquence pour la destination dans le paquet de réponse est supérieur à la valeur présente dans la table de routage.
- les numéros de séquences sont égaux mais la nouvelle route est plus courte.

Afin de limiter le coût dans le réseau, AODV propose d'étendre la recherche progressivement, initialement, la requête RREQ est diffusée à un nombre de sauts limité. Si la source ne reçoit aucune réponse après un délai d'attente déterminé, elle retransmet un autre message de recherche en augmentant le nombre maximum de sauts. Cette procédure est répétée un nombre maximum de fois avant de déclarer que cette destination est injoignable.

5.3.1.2 Maintenance de route

La maintenance des routes est réalisée en deux étapes. La première étape consiste en la détection de la perte d'une route. Quand un nœud sur une route établie se déplace, les routes qui passent par ce nœud peuvent être rompues. Les nœuds en amont, détectant la perte de connectivité grâce aux paquets HELLO, préviennent les sources affectées en émettant une requête d'erreur RERR. A la réception de ce paquet, le nœud source engage la deuxième étape de la maintenance des routes. Il entame une nouvelle phase de découverte des routes, si un chemin est toujours nécessaire.

AODV utilise un mécanisme d'échange périodique de messages HELLO pour détecter une rupture de lien. Les messages HELLO sont envoyés à tous les voisins par un nœud pour signaler son existence. Un message HELLO contient la liste de tous les nœuds connus par l'émetteur ; en vérifiant qu'il est présent dans un message HELLO reçu de son voisin, un nœud peut ainsi vérifier que le lien entre lui et le voisin est bidirectionnel. Si un nœud ne reçoit plus de messages HELLO d'un voisin, il peut déduire une rupture de lien.

Quand un nœud détecte une rupture de lien, il envoie une requête d'erreur RRER à toutes les sources affectées. Si un nœud source reçoit une requête d'erreur, il peut décider de lancer à nouveau une découverte de route dans l'éventualité où il a toujours du trafic à envoyer.

5.3.1.3 Gestion des numéros de séquence

Chaque nœud possède son propre numéro de séquence afin de dater les informations qui proviennent de lui et de lui seul. Un numéro de séquence est incrémenté dans les cas suivants :

- avant de commencer une découverte de route, un nœud incrémente son numéro de séquence ;
- avant d'envoyer une réponse RREP, le nœud met à jour son numéro de séquence en utilisant le maximum du numéro de séquence actuel et de celui indiqué comme numéro de séquence destination dans la requête RREQ reçue ;
- en cas de rupture d'un lien, pour chaque route passant par le lien, le numéro de séquence associé à la destination de la route est incrémenté avant d'envoyer la requête d'erreur RRER informant de la rupture du lien.

Le protocole AODV utilise l'inondation pour découvrir les routes, ce qui peut générer un trafic de contrôle énorme. Comme tout protocole réactif, AODV ajoute un délai initial lors de l'envoi des premiers paquets vers une destination non connue. Un inconvénient d'AODV est qu'il n'existe pas de format générique des messages. Chaque message a son propre format : RREQ, RREP, RERR. L'utilisation des numéros de séquence crée aussi une certaine complexité, mais a l'avantage de permettre de fortement limiter les retransmissions inutiles.

5.3.2 Le protocole DSR

DSR (Dynamic Source Routing) [Johnson and Maltz, 1996] est un protocole qui est normalisé dans la RFC 4728 [Johnson et al., 2007]. Ce protocole crée les routes à la demande comme le protocole AODV. Il utilise la technique "routage à la source" dans laquelle la source inclut dans l'entête du paquet la route complète par laquelle un paquet doit passer pour atteindre sa destination. Les nœuds intermédiaires entre la source et la destination n'ont pas besoin de maintenir à jour les informations sur la route traversée puisque la route

complète est insérée dans l'entête du paquet. DSR est composé de deux mécanismes : la découverte de route et la maintenance de route. Le premier permet de chercher les routes nécessaires à la demande, tandis que le second permet de s'assurer de la maintenance des routes tout au long de leur utilisation.

5.3.2.1 Découverte de route

La découverte de route est initiée par un nœud qui veut communiquer avec une destination pour laquelle il ne possède pas de route. Il diffuse dans le réseau un paquet de requête RREQ similaire à celui d'AODV. Ce message contient l'adresse du nœud initiateur, l'adresse de la destination, un identifiant unique de la requête et la liste de tous les nœuds parcourus par le message.

Chaque nœud qui reçoit la requête et qui n'est pas la destination vérifie s'il peut la retransmettre. La requête ne sera retransmise que si aucune requête du même nœud initiateur avec le même identifiant n'a été reçue et si le nœud n'apparaît pas dans la liste des nœuds parcourus par le message. Avant la retransmission, le nœud insère son adresse dans le paquet RREQ.

La réponse RREP à la requête est retournée par la destination ou par un autre nœud qui possède une route vers la destination. Le paquet réponse de route contient la liste des nœuds parcourus par la requête RREP. Si la destination possède une route vers le nœud initiateur, alors cette route est utilisée pour l'envoi de la réponse RRER. Dans le cas contraire, deux cas sont possibles. Le premier consiste à lancer le mécanisme de découverte de route pour obtenir une route de la destination au nœud initiateur de la première requête. Dans ce cas, la réponse doit être incluse dans le message de requête. Le second cas est invoqué si les liens sont symétriques, alors la liste des nœuds parcourus est inversée et est utilisée comme route pour la réponse RREP.

Tout nœud intermédiaire acheminant une réponse RREP peut disposer gratuitement d'une route vers la destination ayant émis cette réponse, en plus des routes vers les nœuds entre lui-même et la destination.

5.3.2.2 Maintenance de route

Si un nœud détecte, à l'aide de sa couche liaison, qu'un lien vers un nœud suivant est coupé, il considère invalide toute route passant par ce nœud suivant. Il envoie aussi un message d'erreur de route RRER à tout nœud utilisant le nœud suivant dans une de leurs routes pour les prévenir que celles-ci ne sont plus valides. Le message d'erreur contient l'adresse du nœud qui a détecté l'erreur et celle du nœud qui le suit dans le chemin. Lors de la réception du paquet erreur de route par la source, le nœud concerné par l'erreur est supprimé du chemin sauvegardé, et tous les chemins qui contiennent ce nœud sont tronqués à ce point là. Par la suite, une nouvelle opération de découverte de routes vers la destination est initiée par la source. Si le nœud détectant un lien coupé dispose d'une route alternative

vers la destination, il peut l'utiliser en remplacement de la route spécifiée par l'émetteur du paquet.

Le protocole DSR comme le protocole AODV utilise l'inondation pour découvrir les routes ce qui génère un trafic de contrôle énorme. En contre partie, DSR ne génère pas de messages périodiques. En plus, la taille des paquets de données dans DSR devient très grande quand le nombre de nœuds dans réseau est grand, puisque les paquets doivent porter les adresses de chaque nœud dans la route traversée. DSR a aussi un délai avant de commencer la transmission des paquets provoqués par la procédure de découverte de route.

Parmi les avantages de l'utilisation de la technique "routage à la source" du protocole DSR est que les nœuds de transit n'aient pas besoin de maintenir les informations de mise à jour pour envoyer les paquets de données, puisque ces derniers contiennent toutes les décisions de routage. En outre, dans ce protocole, il y a une absence totale de boucle de routage, car le chemin source-destination fait partie des paquets de données envoyés.

5.3.3 Le protocole DYMO

DYMO (DYnamic Manet On-demand) [Chakeres and Perkins, 2010] est un protocole qui détermine les routes de manière dynamique à la demande. Il est considéré comme une évolution d'AODV en proposant un format de message extensible. Ce protocole est encore en cours de développement et n'a pas été très étudié. Le fonctionnement de base de DYMO est similaire à AODV notamment dans les phases de découverte de route et de maintenance de route qui ont été déjà présentées dans la partie 5.3.1. Nous notons les principales différences de DYMO par rapport à AODV :

- dans la découverte de route, la destination de la requête RREQ est la seule qui y répond.
- la liste des nœuds parcourus par les requêtes RREQ peut être incluse dans les requêtes, comme c'est le cas pour DSR. Cette information peut être utilisée afin de disposer gratuitement de certaines routes.
- pour préserver la route en utilisation, les nœuds étendent la durée de vie de la route en transmettant un paquet.
- pour tenir en compte les changements dans la topologie du réseau, les nœuds surveillent les liens sur lesquels le trafic reste fluide.

DYMO est intéressant pour les dispositifs de mémoire limitée puisque l'état de routage maintenu dans chaque nœud est minimal. Ce nouveau protocole essaye de tirer profits du meilleur des protocoles de la génération précédente mais il n'apporte néanmoins aucun changement radical dans l'approche du problème de routage dans les réseaux ad hoc.

5.3.4 Autres protocoles réactifs

Plusieurs autres protocoles réactifs ont été proposés ces dernières années. Nous pouvons cités à titre d'exemple LMR, TORA, AOMDV, BSR et ABR. Ces protocoles visent à diminuer le coût de la maintenance des routes.

Le protocole LMR (Light-weight Mobile Routing) [Corson and Ephremides, 1995] construit des routes seulement à la demande en inondant le réseau avec un paquet de requête. Le protocole LMR permet de maintenir au niveau de chaque nœud un ensemble de multiples routes vers la destination. Les routes supplémentaires augmentent la fiabilité. En outre, si l'information du nombre de sauts est ajoutée dans le paquet de réponse, chaque nœud peut améliorer sa décision de routage en choisissant le chemin le plus court. Par contre, LMR peut produire temporairement des routes invalides.

Le protocole TORA (Temporally Ordered Routing Algorithm) [Park and Corson, 1997], comme LMR, détermine plusieurs chemins vers une même destination, ce qui diminue l'effet négatif des changements de topologie sur le routage des données. La différence entre TORA et LMR réside dans la manière de maintenir ces différents chemins. L'algorithme TORA utilise la propriété "orienté destination" des graphes acycliques orientés. Un graphe acyclique orienté (DAG) est orienté destination s'il y a toujours un chemin possible vers une destination spécifiée. Le graphe devient non orienté destination, si un lien (ou plus) devient défaillant. Dans ce cas, le protocole TORA utilise le concept d'inversement de liens. Ce concept assure la transformation du graphe précédent, en un graphe orienté destination durant un temps fini.

Le protocole AOMDV (On-Demand Multipath Vector Routing in Ad Hoc Networks) [Marina and Das :, 2001] est une extension multi-chemin du protocole AODV qui permet de déterminer plusieurs chemins à liens disjoints entre une paire de nœuds. Deux chemins sont à liens disjoints s'ils n'ont pas de liens en commun. Ce protocole cherche plusieurs chemins pendant une même phase de découverte de routes. Mais uniquement la meilleure route en terme de nombre de saut est utilisée pour la transmission de données. Les autres routes calculées ne seront utilisées que lorsque la route principale est rompue. L'utilisation de chemins disjoints garantit que la rupture d'un lien sera répercutée sur un seul chemin. Cependant, le calcul et le maintien de plusieurs routes entre une source et une destination exigent une occupation plus importante de la table de routage et consomme plus de mémoire du nœud.

Le protocole BSR (Backup Source Routing) [GUO and YANG, 2002] est protocole de routage à la demande qui établit et maintient des routes de secours qui peuvent être utilisées après la rupture du chemin principal. L'avantage de BSR est de réduire la fréquence d'invoquer la phase de découverte qui est reconnue comme une source importante de trafic de contrôle dans les protocoles à la demande. Une nouvelle métrique de routage, appelée fiabilité, est introduite pour choisir le chemin de secours. Comparé à DSR, le protocole BSR améliore les performances du réseau en cas de haute mobilité.

Le protocole ABR (Associativity-Based Routing) [Toh, 1996] cherche à trouver des chemins ayant une grande durée de vie. Pour ce faire, il introduit la notion de stabilité de route comme nouvelle métrique de routage. Le choix d'une route tient compte de la stabilité du lien entre deux nœuds du réseau. Pour déterminer cette stabilité, chaque nœud génère périodiquement des messages de contrôle afin de faire connaître sa présence à ses voisins. Un nœud calcule la stabilité d'un voisin en fonction du temps que ce voisin passe dans son voisinage.

5.4 Protocoles de routage proactifs

5.4.1 Le protocole DSDVR

DSDV (Destination-Sequenced Distance-Vector) [Charles E.Perkins and Bhagwat, 1994] est un protocole de routage de type vecteur de distance. Chaque nœud maintient une table de routage contenant des informations sur les destinations accessibles dans le réseau. Ces informations comprennent le nœud suivant utilisé pour atteindre la destination, le nombre de sauts qui sépare le nœud de la destination et le numéro de séquence estampillé par la destinataire. Ce numéro de séquence permet de distinguer les nouvelles routes des anciennes.

Chaque nœud envoie périodiquement à ses voisins la totalité de sa table de routage. D'autres paquets de mise à jour sont aussi envoyés à la suite d'un changement dans la topologie du réseau. Ces paquets n'incluent que les entrées de la table affectées par le changement et ont pour objectif de propager les informations de routage aussi rapidement que possible.

Quand un nœud reçoit un paquet de mise à jour, il le compare avec les informations existantes dans sa table de routage. Toute entrée dans la table est mise à jour si l'information reçue est plus récente (ayant un numéro de séquence plus grand), ou si elles ont le même numéro de séquence mais avec une distance plus courte.

Dans le protocole DSDV, une unité mobile doit attendre jusqu'à ce qu'elle reçoive la prochaine mise à jour initiée par la destination afin de mettre à jour l'entrée associée à cette destination dans la table de distance. De ce fait, la réaction de DSDV aux changements de la topologie est considérée lente. D'autre part, ce protocole cause une charge de contrôle importante dans le réseau à cause des paquets de mise à jour envoyés périodiquement ou à la suite des événements.

5.4.2 Le protocole OLSR

OLSR (Optimized Link State Routing) [Jacquet et al., 2001] est un protocole de routage à état de liens qui a fait l'objet de la RFC 3626 [Clausen and Jacquet, 2003]. OLSR

introduit le concept des relais multi-points MPR (MultiPoint Relays) pour optimiser la diffusion des messages topologiques sur le réseau. Dans un routage à état de liens, tous les nœuds du réseau diffusent les messages topologiques. Dans OLSR, chaque nœud possède un ensemble de nœuds particuliers qui sont les seuls autorisés à retransmettre ces messages topologiques. Ces nœuds particuliers sont les nœuds MPR qui sont sélectionnés parmi les voisins à un saut de telle sorte que l'ensemble des MPR permette au nœud l'accès à tous les nœuds voisins à deux sauts. Ainsi, chaque nœud, pour savoir s'il peut retransmettre un paquet de contrôle reçu, doit disposer de la liste des nœuds qui l'on choisit comme MPR et qui sont dits ses sélectionneurs multi-points.

5.4.2.1 Sélection des relais multi-points

Pour sélectionner un ensemble de voisins qui forment ses MPRs, un nœud doit disposer, en plus de la liste de ses voisins directs, la liste de ses voisins à deux sauts. Ces informations sont obtenues grâce à l'échange périodique entre voisins directs du paquet HELLO. Ce paquet contient les informations suivantes :

- la liste des voisins pour lesquelles le nœud a reçu un paquet HELLO ;
- la liste des voisins accessibles par un lien bidirectionnel ;
- la liste des voisins que le nœud a choisis comme MPRs.

A l'aide des messages HELLO de ses voisins, un nœud peut donc connaître les voisins de ses voisins et dons ses voisins à deux sauts et être par la suite capable de choisir ses MPRs.

Quand un nœud reçoit un paquet HELLO d'un voisin, il insère ce voisin dans la première liste de son propre paquet HELLO. Si le nœud remarque qu'il se trouve dans la première liste du paquet reçu, alors il conclut que le lien entre lui et le voisin est bidirectionnel ; il enregistre donc le voisin dans la seconde liste. Enfin, si le nœud est dans la troisième liste, alors il est un nœud MPR pour ce voisin et doit donc retransmettre les paquets en diffusion de ce voisin.

Lorsque l'ensemble des voisins ou l'ensemble des voisins à deux sauts change, l'ensemble des relais multi-points doit être recalculés.

5.4.2.2 Calcul des routes

Les messages topologiques, appelés TC (Topology Control) sont envoyés périodiquement et en diffusion sur le réseau entier uniquement par les nœuds choisis comme MPR. Ces messages contiennent la liste des sélectionneurs multi-points du nœud. Grâce à ces messages, chaque nœud peut construire une topologie du réseau basée sur les MPR. Chaque nœud applique par la suite l'algorithme de Dijiksra pour calculer une route à chaque nœud. Les routes construites sont donc une suite de MPRs.

La table de routage est recalculée dans le cas où un lien vers un voisin direct ou à deux sauts est apparu ou perdu. Cette mise à jour consiste à appliquer l'algorithme du plus court

chemin au niveau de chaque nœud et ne génère pas de trafic supplémentaire sur le réseau.

La technique des MPRs employée dans OLSR est intéressante car elle améliore de manière significative le trafic de contrôle comparée à la diffusion par inondation classique. En outre, les informations sur l'état de liens ne sont générées que par les nœuds élus comme MPR ce qui minimiser aussi le nombre de messages de contrôle diffusés dans le réseau. Ces messages de contrôle ne contiennent que les liens qui relient un nœud avec les nœuds qui l'ont élu MPR. Malgré que ces informations sur les états de liens du réseau soient partielles, elles sont suffisantes pour qu'un nœud calcule des routes optimales vers chaque nœud du réseau. Comme tout protocole proactif, OLSR nécessite des ressources plus importantes qu'un protocole réactif : la connaissance de la topologie du réseau a un coût, notamment en terme d'énergie. De plus, le calcul des routes avec l'algorithme de Dijkstra implique la nécessité d'une puissance de calcul un peu plus importante. Ces aspects rendent OLSR peu utilisable dans les réseaux de capteurs, mais ne posent pas réellement de problèmes pour des appareils un peu plus évolués. Enfin, le fait que les routes utilisées ne sont formées que par des MPRs est une limitation. Il peut y avoir un engorgement au niveau des liens entre des relais multi-points si la sélection globale des relais multi-points dans le réseau n'est pas assez variée. Cet aspect rend impossible l'utilisation de toute la capacité disponible dans le réseau.

5.4.3 Autres protocoles proactifs

Le protocole GSR (Global State Routing)[Chen and Gerla, 1998] est un protocole proactif à état de liens où chaque nœud connaît la topologie globale du réseau ce qui lui permet de calculer les routes pour atteindre chaque destination. GSR diffère des protocoles à état de liens dans le fait que les nœuds ne diffusent pas leurs états de liens à tout le réseau, mais ils se limitent à l'envoyer aux voisins uniquement. Ainsi, GSR réduit le trafic des paquets de contrôle. Le problème de GSR est la taille des ses paquets de mise à jour (table de topologie) qui peuvent devenir considérable si le réseau contient un grand nombre de nœuds. En plus, il a une lenteur dans la détection des changements de la topologie.

Le protocole FSR (Fisheye State Routing) [Pei et al., 2000] est un protocole à état de liens qui utilise la technique "oeil de poisson" (fisheye) dans le but de réduire le nombre de messages de contrôle. Cette technique est utilisée dans la représentation des données graphiques où la capture se fait avec précision pour les points proches du point focal et diminue quand la distance séparant le point vu et le point focal augmente. Dans le routage qui utilise la technique fisheye, un nœud maintient des informations de routage plus précises sur ses voisins directs en termes de distance et de qualité du chemin. Cette précision diminue progressivement quand la distance augmente. La diminution de la précision est assurée en changeant les fréquences de mise à jour, et cela en utilisant des périodes d'échanges différentes pour les différentes entrées de la table de routage. Pour les entrées qui correspondent

aux nœuds les plus proches, la période d'échange est relativement petite. Elle augmente progressivement quand les nœuds deviennent de plus en plus lointains.

Le protocole WRP (Wireless Routing Protocol) [Murthy and Garcia-Luna-Aceves, 1995] est un protocole de routage à vecteur de distance. Les nœuds envoient les vecteurs de distance uniquement à la suite des changements dans la topologie du réseau. Pour être conscient de ces changements, les nœuds échangent périodiquement des paquets pour mettre en évidence leur présence dans leur voisinage.

Le protocole STAR (Source Tree Adaptive Routing) [Garcia-Luna-Aceves and Spohn, 1999] est un protocole proactif à état de liens qui se base sur le principe LORA (Least Overhead Routing Approach) habituellement utilisé dans les protocoles réactifs. Cette stratégie garde les chemins aussi longtemps que possible pour minimiser le trafic de contrôle. En plus, STAR, dans le même objectif de réduire les messages de contrôle, ne cherche pas le plus court chemin. Après une procédure initiale de découverte de voisins, chaque nœud maintient un arbre contenant l'ensemble des routes préférées pour joindre ses voisins. Dans l'étape suivante, correspondante à la première mise à jour, chaque nœud envoie son arbre à tous les autres nœuds. Ainsi, chaque nœud peut connaître une topologie de tout le réseau et construire un arbre qui contient des routes vers toutes les destinations. Ce protocole réduit la quantité d'informations de contrôle échangées en éliminant les mises à jour périodiques du protocole à état de liens. L'envoi de l'arbre n'est pas fait périodiquement, il est réalisé uniquement lors de changements importants sur le réseau (détection d'une nouvelle destination, rupture d'un lien,etc.). Ce protocole reste performant dans les grands réseaux puisqu' il réduit le nombre de messages de contrôle transmis.

5.5 Protocoles de routage hybrides

5.5.1 Le protocole CBRP

Le protocole de routage CBRP (Cluster Based Routing Protocol) [Jiang et al., 1999] est un protocole hybride se basant sur la clusterisation. Les nœuds sont rassemblés en des clusters de diamètre égal à deux sauts. Ce protocole se compose principalement des entités suivantes :

- La formation des clusters
- La découverte des clusters adjacents
- Le routage

5.5.1.1 La formation des clusters

Le nœud ayant le plus petit identifiant dans son voisinage sera élu comme cluster-head. Initialement, les nœuds sont tous dans un état indéterminé. Un nœud en état indéterminé active un temporisateur et envoie en diffusion un message HELLO. Deux cas se présentent :

- Un cluster-head lui renvoie un message HELLO en lui indiquant une liaison bidirectionnelle. Ainsi, le nœud devient membre de ce cluster et construit sa table de voisinage à deux sauts.
- Si le temporisateur expire sans recevoir un message HELLO d'un cluster-head et dans le cas où le nœud possède au moins un lien bidirectionnel vers au moins un nœud voisin, il considère lui-même un cluster-head. Dans le cas contraire, il reste dans l'état indéterminé et il répète la même procédure.

La mise à jour des clusters suit les deux règles suivantes. Un nœud membre rattaché à un cluster-head par une liaison bidirectionnelle ne deviendra pas cluster-head même s'il a un identifiant plus petit que son cluster-head. Si deux clusters-heads sont reliés par une liaison bidirectionnelle pendant un certain temps celui ayant le plus grand identificateur abandonnera sa fonction de cluster-head. Ses membres essayent de joindre des clusters voisins.

5.5.1.2 La découverte des clusters adjacents

Avec l'échange des messages HELLO, chaque nœud établit et maintient une table de son voisinage à deux sauts avec des informations sur l'état de liens. En plus de cette table, chaque nœud possède aussi une table des clusters adjacents appelée CAT (Cluster Adjency Table). Une entrée dans cette table est associée à un cluster voisin : elle contient l'identificateur du cluster-head et le nœud de liaison à travers lequel le cluster peut être atteint. Cette table est aussi construite grâce à l'échange des messages HELLO. En effet, les cluster-heads se trouvant dans le voisinage d'un nœud membre sont au plus distants de deux sauts. Il est donc possible à chaque nœud membre de les connaître grâce aux messages HELLO. Il en est de même pour les clusters-heads qui sont séparés les uns des autres par deux ou trois sauts.

5.5.1.3 Le routage

CBRP est basé sur le principe de "routage à la source" et se compose de deux phases : la découverte de la route et le routage des paquets. La structure en clusters permet de réduire le phénomène d'inodation pendant la phase de découverte de la route. Quand un nœud source veut envoyer des données à un nœud destination il diffuse alors par inondation une requête de demande de chemin uniquement aux cluster-heads des clusters voisins. Un cluster-head qui reçoit la requête de demande vérifie, en utilisant sa table des membres du cluster, l'existence du nœud destination dans son cluster. Si la destination existe, le cluster-head y envoie directement la réponse, si non la requête est diffusée aux cluster-heads des clusters voisins. L'adresse des cluster-heads est incluse dans la requête de demande de chemin. Un cluster-head ignore toute requête déjà traitée. Quand la destination reçoit le paquet contenant la requête, elle répond par l'envoi du chemin qui a été sauvegardé dans le paquet de la requête. Dans le cas où le nœud source ne reçoit pas de réponse après une certaine période,

il envoie de nouveau une requête de demande de chemin. Une fois que la source reçoit une réponse de chemin, les données sont alors routées à l'aide de la méthode "routage à la source" où le paquet de données contient la liste des nœuds à traverser de la source à la destination. Si un nœud détecte qu'un lien est coupé, il envoie un message d'erreur à la source et il applique un mécanisme de réparation locale. Dans ce mécanisme, si un nœud trouve qu'un nœud n suivant ne peut pas être atteint, il essaie de vérifier si le nœud n ou le nœud qui vient après n peut être atteint à travers un autre nœud voisin. Si l'un des deux cas est vérifié, les données sont envoyées en utilisant le chemin réparé. Les nœuds étant mobiles dans le réseau, il est possible qu'une route devienne de moins en moins optimale. Il existe donc un mécanisme appelé "shortening mechanism" qui permet à chaque nœud de recalculer un chemin optimal en se servant de sa connaissance de la topologie du réseau à deux sauts pour la remise de paquets. A leur réception, le nœud de destination renvoie alors un message RREP pour valider cette nouvelle route.

5.5.2 Le protocole ZRP

ZRP (Zone Routing Protocol) [Haas et al., 2002a] est un protocole hybride qui combine les deux approches proactive et réactive. Le protocole ZRP divise le réseau en différentes zones. Pour chaque nœud, il définit une zone de routage exprimée en nombre de sauts maximal σ . Ainsi, la zone de routage d'un nœud inclut tous les nœuds qui sont à une distance au maximum de σ sauts. Les nœuds qui sont exactement à σ sauts sont appelés nœuds périphériques. À l'intérieur de cette zone, ZRP utilise un protocole proactif et à l'extérieur de cette zone de routage, il fait appel à un protocole réactif. Le protocole proactif est IARP (IntrAzone Routing Protocol) [Haas et al., 2002d] et celui réactif est IERP (IntErzone Routing Protocol) [Haas et al., 2002c].

Chaque nœud doit tout d'abord connaître ses voisins. Pour cela, ZRP utilise soit le protocole de contrôle d'accès au support (MAC) pour connaître les voisins immédiats ou le protocole NDP (NeighbourDiscovery Protocol) pour la transmission et la gestion des échanges de messages HELLO. Par la suite, chaque nœud invoque le protocole IARP pour découvrir les routes vers tous les autres nœuds qui se trouvent dans sa zone de routage. Cependant, le protocole IERP est utilisé à la demande pour chercher les routes entre un nœud et une destination qui se trouvent à l'extérieur de sa zone de routage. Un troisième protocole BRP (Bordercast Resolution Protocol) [Haas et al., 2002b] est inclus avec IERP pour guider la propagation des requêtes de recherche de route dans le réseau. BRP utilise les données de la topologie fournies par le protocole IARP afin de construire sa liste des nœuds de périphérie et la façon de les atteindre.

5.6 Scalabilité du routage à état de liens

Le problème de passage à l'échelle des protocoles à état de liens a déjà été étudié dans des travaux précédents. L'introduction de la technique connue de clusterisation dans un protocole de routage à état de liens est une solution pour réduire la complexité du protocole et limiter l'overhead généré. Nous étudions quelques travaux qui ont été proposés pour améliorer la scalabilité des protocoles de routage à état de liens.

5.6.1 Approche plate : le protocole F-OLSR

Fisheye-OLSR [Clausen, 2003] est un protocole de routage qui intègre la technique "oeil de poisson" (fisheye)[Pei et al., 2000] dans le protocole OLSR. Le principe du routage fisheye consiste à rafraîchir l'information de topologie plus fréquemment pour les nœuds proches que pour les nœuds plus lointains. Donc, la fréquence de la mise à jour de l'information de topologie diminue quand la distance augmente. Cette optimisation est justifiée par le fait qu'une idée vague de l'emplacement du nœud est suffisante pour acheminer les paquets aux destinations lointaines. Quand les paquets des données s'approchent de la destination, l'information de routage devient de plus en plus précise et les nœuds les plus proches peuvent acheminer les paquets des données plus précisément. L'introduction de la technique fisheye est faite à travers le champ TTL (Time-To-Live) dans les messages TC (Topology Control) du contrôle de la topologie. Le nœud initiateur du message TC met la valeur du champ TTL de ce message selon la distance, en nombre de sauts, que ce message doit traverser. Des études analytiques [Adjih et al., 2004] montrent que OLSR avec la technique fisheye peut atteindre les limites théoriques de scalabilité calculées par Gupta et Kumar [Gupta and Kumar, 2000]. Mais, les auteurs de [Clausen, 2003] n'indiquent pas l'algorithme nécessaire pour adapter le protocole au passage à l'échelle. D'autre part, même si la périodicité de l'information de la topologie est réduite, il est encore nécessaire de la diffuser sur le réseau entier et donc le nombre de messages envoyés augmentent quand le nombre de nœuds du réseau augmente. De plus, chaque nœud conserve et calcule encore les routes à tous les nœuds destinations potentiels dans le réseau et donc le problème de stockage et d'overhead de OLSR n'est pas entièrement résolu.

5.6.2 Approches basées sur une structure des clusters

5.6.2.1 Le protocole OLSR-Trees

OLSR-Trees [Baccelli, 2006] présente une technique de clusterisation basée sur la structure d'arbre comme celle dans [Nikaein et al., 2000]. Ce protocole introduit un routage hiérarchique dans OLSR sans supposer l'existence d'une hétérogénéité des nœuds du réseau. Le principe d'OLSR-Trees est d'appliquer OLSR régulier à l'intérieur de la structure d'arbre. OLSR est aussi appliqué au niveau supérieur de la topologie des arbres formées.

Algorithme de clusterisation

L'algorithme de clusterisation fait correspondre à chaque cluster un arbre et chaque cluster-head sera la racine de cet arbre. La clusterisation repose sur la connectivité locale des nœuds pour élire le cluster-head (la racine). Chaque nœud choisit comme parent son voisin qui a le plus grand nombre de voisins. Un nœud ayant le maximum nombre de voisins dans son voisinage devient la racine de l'arbre. Pour rompre l'égalité, le nœud ayant le plus petit identifiant devient la racine de l'arbre. Ainsi, le réseau est vu comme une collection d'arbres. La figure 5.2 montre un exemple de topologie de clusters qui sont formés par la clusterisation avec les arbres.

Pour former et maintenir les arbres, les nœuds échangent périodiquement un message Branch (en plus des messages de contrôle d'OLSR). Dans ce message, le nœud spécifie son identifiant, l'arbre auquel il appartient et son parent dans l'arbre, ainsi que la profondeur qui mesure la distance qui le sépare de la racine. Le message Branch est envoyé groupé avec le message HELLO et il est transmis à un seul saut. La structure d'arbre crée peut être utilisée dans un routage hiérarchique avec OLSR.

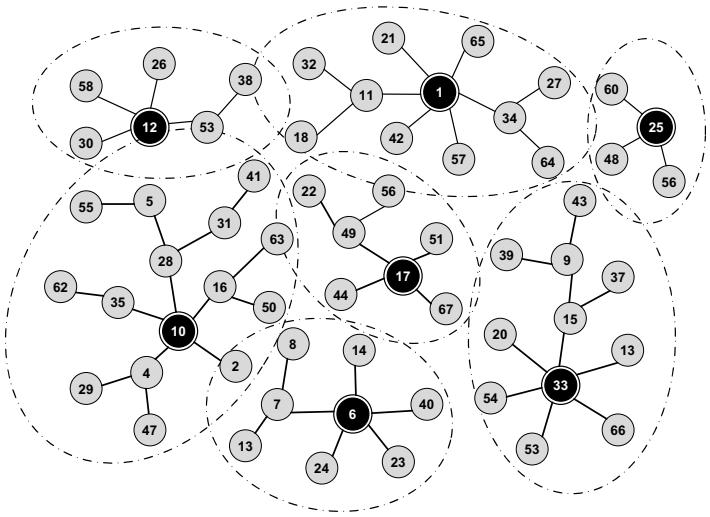


Figure 5.2 – Exemple d'une structure des clusters créé par OLSR-Trees.

Routage à l'intérieur de l'arbre

OLSR régulier est appliqué à l'intérieur de la structure d'arbre à l'exception des points suivants :

1. Les messages de contrôle parvenus d'un voisin qui n'est pas dans le même arbre, ne sont pas retransmis.
2. La racine de l'arbre est responsable de la communication avec les nœuds se trouvant à l'extérieur de l'arbre.
3. Un nœud en contact avec un autre arbre doit informer la racine.

La retransmission des messages TC : La sélection des MPRs reste inchangée, mais, la diffusion des messages TC est limitée à l'intérieur de l'arbre. Les messages TC envoyés depuis l'arbre seront retransmis uniquement à l'intérieur de l'arbre. La retransmission se fait via les MPRs. Mais, un nœud ne retransmet un message TC qui vient d'un voisin n'appartenant pas au même arbre que si :

1. Il est sélectionné comme MPR par ce voisin ET
2. Le message lui est parvenu pour la première fois ET
3. Il a un autre voisin appartenant au même arbre que ce voisin.

Nœud feuille : Un nœud ayant un ou plusieurs voisins appartenant à d'autres arbres doit informer sa racine. Pour chaque arbre qu'il peut atteindre, il peut informer sa racine par un message appelé message Leaf en spécifiant son identifiant, la racine de l'arbre qu'il peut atteindre et une estimation de la distance entre les deux racines. Le nœud doit diffuser ce message à travers l'arbre à moins qu'il reçoit un autre message Leaf récent qui annonce le même arbre avec une distance plus courte. De cette façon, la racine et les autres nœuds de l'arbre sont informés des chemins qui mènent à tout arbre voisin. Le message Leaf est envoyé groupé avec un message TC à l'intérieur de l'arbre.

Routage à l'extérieur de l'arbre

La racine de l'arbre est responsable de diffuser l'information de routage d'un arbre à un autre. OLSR est opéré à un niveau supérieur : sur la super-topologie formée par les racines des arbres. A ce niveau, chaque arbre, concrétisé par sa racine, se comporte comme s'il était un nœud OLSR (un super-nœud). Comme dans OLSR régulier, ces super-nœuds (les racines) envoient périodiquement les messages super-HELLO et super-TC.

Super-messages : Ces messages sont identiques à un message régulier sauf qu'ils incluent l'adresse du super-saut suivant (racine suivante). Ces super-messages sont acheminés via les routes établies à l'intérieur de chaque arbre, au lieu d'être diffusés. Ils sont les seuls messages à être retransmis à l'extérieur de l'arbre.

Super-HELLO et Super-TC : La racine envoie périodiquement un message super-HELLO à toutes les autres racines qu'il connaît grâce aux messages Leaf. Les super-HELLO sont acheminés en unicast et utilisent les chemins les plus courts qui sont annoncés dans les messages Leaf. De la même manière qu'OLSR, les racines sont informées de leurs super-voisins et peuvent exécuter la sélection des super-MPRs. Les super-HELLO sont retransmis à un seul super-saut (ne sont pas retransmis plus loin que les racines voisines).

En plus au super-HELLO, les racines envoient périodiquement un message super-TC qui est super-diffusé (via des chemins unicast en utilisant les super-MPRs et les chemins les plus courts entre les racines) à toutes les racines du réseau.

Les racines sont ainsi informées de toute la super-topologie formée par les racines. Grâce aux messages super-TC, les racines sont capables de trouver les chemins aux autres

clusters. Le trafic à l'extérieur passe par la racine : quand un nœud a besoin d'envoyer des données à un nœud à l'extérieur de son arbre, il envoie le trafic à la racine qui elle-même l'envoie au nœud destination en suivant le chemin au cluster auquel la destination appartient. Cependant, les auteurs dans [Baccelli, 2006] ne spécifient pas comment les racines déterminent le cluster de la destination. Néanmoins, cette information de localisation peut être ajoutée dans les messages super-TC qui sont diffusés à toutes les racines.

OLSR-trees propose une approche intéressante pour améliorer la scalabilité de OLSR. Cependant, appliquer OLSR sur la topologie des clusters peut produire un trafic de contrôle supplémentaire. D'autre part, les chemins via les arbres ne sont pas optimiaux. En plus, comme toutes les routes vers l'extérieur passent par les cluster-heads (les racines), ces nœuds peuvent être surchargés et présenter des goulots d'étranglement.

5.6.2.2 Le protocole C-OLSR

C-OLSR (Clustered OLSR) [Ros and Ruiz, 2007] est un protocole qui se base sur OLSR et qui suit le même principe qu'OLSR-Trees avec quelques modifications. Comme dans OLSR-Trees, C-OLSR partitionne le réseau en un ensemble de clusters, applique OLSR régulier à l'intérieur de chaque cluster et applique les mêmes mécanismes d'OLSR au niveau des clusters. De même qu'OLSR-Trees, il ne propage les messages TC qu'à l'intérieur du cluster et prévoit deux nouveaux messages C-HELLO et C-TC (semblales aux messages Super-HELLO et Super-TC d'OLSR-Trees) pour émuler le comportement d'un nœud OLSR par un cluster. Similaires aux nœuds Super-MPRs, des nœuds appelés C-MPRs sont élus grâce aux messages C-HELLO. Les C-MPRs sont utilisés pour optimiser la diffusion des messages de contrôle.

Les différences entre OLSR-Trees et C-OLSR sont les suivantes :

- La technique de clusterisation utilisée dans OLSR-Trees se base sur la construction d'arbres tandis que C-OLSR ne dépend pas d'un algorithme de clusterisation spécifique mais il suppose qu'un mécanisme de clusterisation est exécuté dans le réseau ad hoc.
- Les messages de contrôle (Super-HELLO et Super-TC pour OLSR-Trees et C-HELLO et C-TC pour C-OLSR) sont générés par les racines dans le cas d'OLSR-Trees. C-OLSR distingue trois différents modes pour désigner les nœuds qui génèrent ces messages : soit les cluster-heads, soit les nœuds frontières, soit un mode hybride où les nœuds frontières génèrent les messages C-HELLO et les cluster-heads les messages C-TC.
- Les messages Super-HELLO et Super-TC sont transmis en unicast entre les racines. Tandis que les messages C-HELLO et C-TC sont transmis en diffusion vers tous les nœuds. La diffusion des messages C-HELLO et C-TC vers tous les nœuds peut produire un overhead important.

- Pour le routage inter-clusters, un nœud dans OLSR-Trees transmet le trafic à la racine qui est responsable de l'acheminer à la destination. Dans C-OLSR, un nœud établit des routes vers chaque cluster et non pas vers chaque nœud. Cependant, les auteurs dans [Ros and Ruiz, 2007] ne spécifient pas comment les nœuds déterminent le cluster de la destination. Ce qui explique pourquoi les auteurs dans l'évaluation des performances de C-OLSR ont donné des emplacements fixes à un nombre de cluster-heads.

5.6.2.3 Le protocole SA-OLSR

Les auteurs dans [Canourgues et al., 2008] proposent une adaptation scalable du protocole OLSR basée sur la clusterisation. Dans le reste du rapport, nous nommons ce protocole SA-OLSR. Le protocole de routage SA-OLSR est indépendant de l'algorithme de clusterisation utilisé. Mais, il suppose qu'un mécanisme de clusterisation est exécuté dans le réseau ad hoc et que chaque nœud est informé de l'adresse de son cluster-head. En plus, SA-OLSR recommande d'utiliser un algorithme de clusterisation à K-sauts qui forme des clusters de diamètre plus large que deux sauts. Pour les communications intra-cluster, SA-OLSR utilise le protocole OLSR régulier et la propagation de l'information du contrôle de la topologie est limitée à l'intérieur du cluster. Pour le routage à l'extérieur du cluster, contrairement à d'autres approches basées sur OLSR qui utilisent la clusterisation comme OLSR-Trees et C-OLSR, SA-OLSR n'applique pas une version d'OLSR au niveau de la topologie des clusters. En effet, au lieu d'échanger les messages complexes d'OLSR, les cluster-heads sont amenés à envoyer un nouveau message appelé "TC Cluster" pour assurer le routage en dehors du cluster. Grâce à l'information contenue dans ces messages, un nœud qui veut envoyer des données à une destination n'appartenant pas à son cluster est capable de connaître le prochain nœud vers le clusterhead de la destination. C'est la seule information que chaque nœud doit connaître pour pouvoir acheminer les paquets des données.

Le message HELLO

La première modifi cation du protocole OLSR porte sur le message HELLO. Chaque nœud doit inclure l'adresse de son clusterhead dans son message HELLO afin que ses voisins puissent savoir s'ils appartiennent au même cluster. Dans le but d'être conforme avec le protocole OLSR régulier, le format du message HELLO n'est pas modifié. Un message HELLO est composé de plusieurs blocs, un bloc par valeur du code du lien. Le code du lien spécifie des informations sur le lien entre l'interface de l'émetteur et la liste des interfaces de ses voisins. Il spécifie aussi des informations sur le statut des voisins. Le protocole SA-OLSR introduit un nouveau code du lien pour indiquer l'adresse du clusterhead. Le protocole SA-OLSR sauvegarde cette adresse dans la liste des voisins du nœud et lui fait correspondre un nouveau champ.

Le messages TC

La version régulière du protocole OLSR est utilisée à l'intérieur de chaque cluster. Pour limiter la propagation des messages TC à l'intérieur du cluster, ces messages ne sont jamais retransmis par un nœud qui n'appartient pas au même cluster que le nœud qui est à l'origine du message TC. L'algorithme de sélection des MPRs est exécuté sans prendre en considération les clusters du réseau. Par conséquent, chaque MPR envoie périodiquement les messages TC qui contiennent la liste des nœuds qui l'ont choisit comme MPR. A la réception d'un message TC, le nœud exécute OLSR régulier. Cependant, l'algorithme de décision de faire suivre le message TC est modifié puisqu'il dépend du clusterhead du nœud qui est à l'origine du message TC. Chaque nœud exécute l'algorithme suivant : si l'adresse de l'émetteur du message n'est pas détectée dans son voisinage à 1-saut ou si l'adresse du clusterhead de l'émetteur n'est pas la même que son propre clusterhead, alors le message TC n'est pas retransmis. Il n'y a aucun besoin que l'initiateur du message TC ajoute l'adresse de son clusterhead dans le message TC. En effet, un nœud est capable de savoir le clusterhead du nœud ayant envoyé le message TC grâce à l'information contenue dans la liste de ses voisins. En plus, le clusterhead du nœud qui a retransmis le message (l'émetteur) est nécessairement le même que celui qui lui a précédemment envoyé le message, autrement le message n'aurait pas été retransmis. De proche en proche, le clusterhead de l'initiateur du message TC est le même que le clusterhead du nœud qui vient d'envoyer le message TC reçu.

Le message TC Cluster

Un nœud est capable de calculer un chemin à tous les nœuds de son cluster. Les nœuds de frontière sont aussi capables de calculer des chemins aux nœuds appartenant aux clusters voisins. L'approche de SA-OLSR concernant les chemins vers les nœuds qui appartiennent aux différents clusters est que chacun nœud devrait savoir le prochain saut vers le clusterhead de la destination. Alors, une fois le paquet des données arrive dans le cluster de la destination, le nœud intermédiaire saura le chemin exact à la destination. Pour se faire, l'information de la topologie des clusters doit être envoyée sur le réseau entier. OLSR-Trees ou C-OLSR reproduisent le protocole OLSR au niveau de la topologie des clusters pour créer les chemins vers les différents clusters. L'approche suivie par SA-OLSR est différente. Il définit un nouveau message TC Cluster qui est envoyé par les clusterheads sur le réseau en utilisant l'algorithme d'inondation des MPRs. Ce message contient les adresses des nœuds qui appartiennent à son cluster. Comme ce message est inondé sur tout le réseau, chaque nœud peut maintenir une table de correspondance nœud/cluster et peut par conséquent déterminer auquel cluster une destination appartient. Néanmoins, l'information du clusterhead de la destination n'est pas suffisante pour acheminer le paquet à la destination. En effet, le chemin au clusterhead ou au moins le prochain saut du chemin au clusterhead est nécessaire. L'information du prochain saut est extraite à partir du message TC Cluster. Quand un

nœud reçoit un message TC cluster, il enregistre comme prochain saut au cluster-head qui a envoyé le message le nœud qui vient de lui faire suivre le message. SA-OLSR considère seulement la première copie reçue du message TC cluster en supposant que cette première copie a suivi nécessairement le chemin le plus rapide et le moins congestionné. Les autres copies sont supprimées.

Envoi et relais des paquets de données

Quand un nœud a un paquet des données à envoyer :

- S'il connaît la destination depuis sa table de routage (la destination appartient à son cluster ou la destination est un clusterhead), il envoie le paquet au prochain saut indiqué dans sa table de routage.
- Si la destination n'est pas dans sa table de routage, il consulte sa table de correspondance nœud/cluster pour connaître le cluster auquel appartient la destination.
  - ⊙ Si la destination n'est pas dans la table, le paquet est supprimé.
  - ⊙ Si la destination est dans la table, le nœud cherche dans sa table de routage le prochain saut vers le clusterhead associé à la destination.

L'adresse de destination du paquet des données n'est pas changée. Quand le prochain saut reçoit les données, le même processus est exécuté.

Bien que SA-OLSR réduit le trafic de contrôle comparé à la solution C-OLSR, en considérant la première copie reçue du message TC cluster, elle peut surcharger certains nœuds. En effet, les nœuds voisins, qui reçoivent le message TC cluster du même nœud, vont choisir le même nœud comme prochain saut. De plus, l'évaluation de performance [Canourgues et al., 2008] étudie seulement le paramètre overhead et ne considère pas une couche MAC réaliste et une réelle mobilité des nœuds.

5.6.3 Approches hiérarchiques

5.6.3.1 Le protocole HSR

Le protocole HSR (Hierarchical State Routing) [Iwata et al., 1999] utilise les notions de groupes dynamiques, niveaux hiérarchiques et la gestion de localisation. Dans HSR, la topologie du réseau est vue sous une forme hiérarchique. Le réseau est partitionné en un ensemble de groupes, dont l'union donne le réseau entier. Dans un groupe, un nœud doit être élu pour représenter le reste des membres. Les représentants des groupes dans un niveau l, deviennent des membres dans le niveau $l + 1$. Ces nouveaux membres, s'organisent en un ensemble de groupes de la même manière du niveau précédent, et ainsi de suite pour le reste des niveaux. Plusieurs algorithmes de partitionnement peuvent être utilisés, dans la création dynamique des groupes et l'élection des représentants de groupes. Le but principal du partitionnement de HSR est l'utilisation efficace des médiums de communication et la

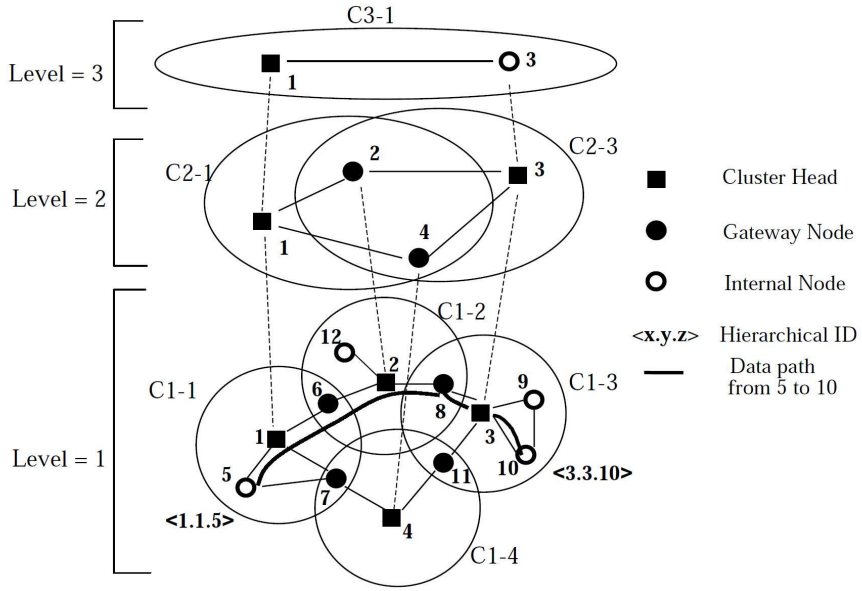


Figure 5.3 – Le partitionnement du réseau en clusters dans HSR [Iwata et al., 1999].

réduction du contrôle de routage. La figure 5.3, illustre l'application de ce mécanisme de partitionnement dans un réseau de 13 nœuds mobiles.

Dans le partitionnement en groupe, on peut avoir trois types de nœuds :

- un nœud représentant du groupe (appelé aussi, tête du groupe)
- un nœud de liaison, qui relie deux groupes ;
- un nœud interne qui est un nœud ordinaire et n'a aucun rôle spécial.

Le nœud représentant d'un groupe donné, peut être vu comme un coordinateur de transmission des données. Les identificateurs (IDs) des nœuds sont uniques pour chaque nœud. Une des méthodes qui peut être appliquée afin d'associer des adresses hiérarchiques, ou HIDs (Hierarchical IDs), aux différents nœuds ; est de prendre les numéros des groupes, dans le chemin reliant la racine et le nœud en question. Par exemple le nœud 6 de la figure 5.3 a l'adresse HID(6) = <1,1,6>, le chemin reliant la racine et le nœud 6, est composé de trois nœuds : le représentant du groupe G1-1, le représentant du groupe G0-1 et enfin le nœud 6 d'ID égal à 6. Un nœud de liaison, peut être atteint, à partir de la racine, en suivant plusieurs chemins, par conséquent, ce genre de nœud peut avoir plus qu'une adresse hiérarchique. Cela ne pose aucun problème, car le nœud peut être atteint à travers ces adresses, et ces dernières sont associées à un nœud unique. On peut toujours trouver une manière d'associer une seule adresse à ce genre de nœuds, par exemple en prenant la plus petite valeur des numéros de groupes dans lesquels appartient le nœud. L'avantage de l'adressage hiérarchique, est le fait que chaque nœud puisse dynamiquement et localement mettre à jour

son HID, lors de la réception des données de mise à jour du routage, provenant des nœuds de niveau supérieur. L'adresse hiérarchique, suffit pour délivrer les paquets de données à une destination, indépendamment de la localisation de la source.

5.6.3.2 Le protocole H-OLSR

Hierarchical OLSR [Ge et al., 2005] propose une amélioration d'OLSR basée sur la clusterisation. HOLSR organise dynamiquement les nœuds dans des niveaux de clusters hiérarchiques et suppose que certains nœuds ont de meilleures capacités de communication (nombre d'interfaces, rayon de propagation, taux de données, bandes de la fréquence, durée de vie de la batterie, etc.). Par exemple, les nœuds équipés avec une seule interface participent au niveau un. Les nœuds de niveau deux sont équipés de deux interfaces : une pour communiquer avec les nœuds de niveau un et l'autre pour communiquer avec les nœuds de niveau deux. Cette deuxième interface doit avoir un rayon de transmission plus large. Plus de niveaux de hiérarchie peuvent être construits selon les capacités de communications des nœuds du réseau. Ainsi, les nœuds avec les meilleures capacités deviennent des cluster-heads pour les nœuds de niveaux inférieurs. L'information de la topologie est échangée seulement à l'intérieur du cluster. Les cluster-heads de même niveau échangent les adresses de leurs nœuds locaux à travers des communications directes. Les nœuds membres ont assez d'information pour acheminer les données à toute destination du même-niveau même cluster. Pour la transmission des données à l'extérieur de la région locale, le cluster-head est toujours utilisé comme gateway par les nœuds de niveaux hiérarchiques inférieurs et il est responsable d'acheminer les données au cluster-head approprié du même-niveau. Cela peut mener aux chemins non optimaux quand la source et la destination sont proches mais appartiennent à des clusters différents. En plus, HOLSR suppose que certains nœuds sont équipés de meilleures capacités de communication ce qui peut ne pas être vérifié dans quelques réseaux.

5.7 Conclusion

Le routage est une fonction de base essentielle au bon fonctionnement des réseaux. Il a fait l'objet de très nombreux travaux de recherches, particulièrement pour les réseaux auto-organisables où plusieurs contraintes doivent être tenues en compte comme la mobilité, la limitations des ressources,etc. Vu la grande diversité des protocoles proposés, certains travaux se sont intéressés à faire une classification des différentes approches. En s'inspirant des taxonomies existantes, nous avons proposées des critères de classification que nous avons juger pertinents pour un protocole de routage. Seule une infime partie des protocoles de routage existants a été présentée dans ce chapitre. Particulièrement, nous avons porté un intérêt aux protocoles qui visent principalement le problème de passage à l'échelle. Deux différentes approches sont proposées : protocoles à base de structure de clusters et proto-

coles hiérarchiques. Malgré les résultats motivants de ces travaux, ils présentent certains carences soit au niveau de leur conception ou de leur expérimentation. De ce fait, nous avons focalisé nos contributions sur la proposition d'un protocole de routage scalable qui tire profit des avantages de la clusterisation. Les détails de la conception et de l'expérimentation de ce protocole se trouvent dans le chapitre suivant.

CHAPITRE 6

PROPOSITION D'UN PROTOCOLE DE ROUTAGE À ÉTAT DE LIENS DES CLUSTERS : CLSR

LA RÉDUCTION du trafic de contrôle est la clé du problème de scalabilité des protocoles de routage. Afin de réduire ce trafic, nous proposons un nouveau protocole de routage CLSR. Ce protocole est un protocole à état de liens basé sur une structure des clusters. Il tâche à supprimer la redondance des informations contenues dans les paquets de contrôle en utilisant seulement deux types de paquets dans la clusterisation et le calcul des routes.

Sommaire

6.1 Introduction 118
6.2 La découverte du voisinage 119
6.3 La structuration du réseau 119
6.3.1 Construction de la structure des clusters 120
6.3.2 Construction de la dorsale virtuelle 120
6.4 La gestion de la topologie 122
6.5 Calcul de la table de routage 124
6.5.1 Routage au voisinage 124
6.5.2 Routage inter-cluster 127
6.6 Acheminement des paquets 129
6.7 Analyse expérimentale 129
6.7.1 Paramètres de simulation 130
6.7.2 Métriques d'évaluation 131
6.7.3 Trafic de contrôle *(Overhead)* 131
6.7.4 Ratio des paquets délivrés 133
6.7.5 Délai de bout-en-bout 135
6.7.6 Nombre moyen de sauts 137
6.8 Conclusion 139

6.1 Introduction

Dans ce chapitre, nous proposons un nouveau protocole de routage à état de liens, appelé CLSR (Cluster-based Link State Routing) [Guizani et al., 2012a]. Ce protocole utilise l'approche état de liens appliquée à une topologie formée par l'interconnexion d'un ensemble de clusters. La technique de clusterisation est utilisée pour réduire les informations de contrôle circulées dans le réseau, les informations de la gestion de la topologie du réseau et la taille des tables de routage.

Comme les protocoles OLSR-Trees et C-OLSR, CLSR utilise un routage à état de liens en inter-cluster. L'apport principal de CLSR est l'utilisation de l'algorithme de clusterisation SSCA [Guizani et al., 2011a]. Cet algorithme n'ajoute pas de trafic supplémentaire car seuls les messages HELLO sont utilisés par le processus de clusterisation. En outre, le choix d'un algorithme de clusterisation à un seul saut permet de réduire les messages de routage utilisés. En effet, les messages d'état de liens en intra-cluster ne sont plus nécessaires avec cette distance car ils sont redondants avec le contenu des messages HELLO. En plus, avec cette distance les cluster-heads peuvent déterminer leurs cluster-heads voisins sans avoir recours à des messages supplémentaires tels que les messages Super-HELLO du protocole OLCR-Trees ou les messages C-HELLO du protocole C-OLSR.

Le protocole CLSR est conçu pour fonctionner efficacement dans des réseaux de grande taille. Il est décomposé en quatre entités complémentaires :

1. **La détection du voisinage :** chaque nœud est conscient de l'état de liens de ses voisins directs et ses voisins à deux sauts grâce aux messages HELLO.

2. **La structuration du réseau :** cette entité s'occupe de l'auto-organisation du réseau. Nous distinguons dans ce cadre deux ensembles de fonctions : la formation et le maintien de la structure des clusters et la formation d'une dorsale virtuelle. Cette dernière structure est composée par un ensemble dominant connecté et permet d'optimiser la diffusion des paquets de contrôle.

3. **La gestion de la topologie :** cette entité s'occupe de la collecte des informations de la topologie du réseau. Elle construit la topologie globale du réseau formée par l'interconnexion de tous les clusters. Cette entité permet aussi de maintenir les informations de localisation des nœuds et de déterminer à quel cluster appartient chaque nœud du réseau.

4. **Calcul des routes et acheminement des données** : À partir des connaissances de la topologie, chaque nœud établit une table de routage qui contient des entrées pour atteindre chaque destination dans le voisinage à deux sauts et chaque cluster du réseau. Les données seront acheminées d'un nœud à un autre selon les entrées des tables de routage.

6.2 La découverte du voisinage

La base des protocoles à état de liens est que chaque nœud du réseau doit suivre l'évolution de l'état de ses liens à travers l'échange des messages HELLO. Ces messages HELLO sont échangés périodiquement entre les voisins, mais non propagés. Ils contiennent des informations sur l'état de liens des nœuds voisins. Le protocole CLSR utilise les informations des messages HELLO pour assurer plusieurs fonctions :

- Découvrir et maintenir la topologie locale à 2-sauts ;
- Former et maintenir la structure des clusters ;
- Maintenir les chemins locaux menant vers les nœuds voisins à 2-sauts ;
- Découvrir et maintenir la liste des clusters voisins.

Pour assurer ces fonctions, le protocole CLSR ajoute certaines informations dans le message HELLO. Ainsi, un message HELLO contient les informations suivantes :

- L'adresse du nœud source ;
- L'adresse du cluster auquel appartient la source ;
- Le nombre de voisins de la source ;
- Des informations relatives à chaque voisin de la source : son adresse, son degré de connectivité et l'adresse de son cluster.

Notons aussi que les messages HELLO contiennent explicitement l'état de la source dans la structure des clusters ainsi que l'état de ses nœuds voisins. En effet, l'adresse du cluster correspond à l'adresse du cluster-head. Ainsi, si l'adresse de la source est égale à l'adresse de son cluster alors le nœud source est un cluster-head. En plus, un identifiant particulier Max-Adresse est utilisé pour exprimer que le nœud n'appartient à aucun cluster s'il n'a pas encore décidé de son rôle. Enfin, les messages HELLO permettent aussi à chaque nœud de détecter les changements de la topologie. En effet, chaque nœud associe un temporisateur à chaque voisin. Ce temporisateur est réinitialisé à chaque fois que le nœud reçoit un message HELLO de son voisin. Le temporisateur est fixé à une valeur supérieure à deux fois la période d'envoi du message HELLO. Si ce temporisateur expire avant la réception d'un message HELLO du voisin associé, le nœud considère que le lien avec ce voisin est rompu et il le supprime de sa liste des voisins.

6.3 La structuration du réseau

CLSR utilise deux structures virtuelles pour organiser le réseau. La première structure est composée d'un ensemble de clusters créé afin de réduire la taille du réseau. Elle représente la topologie du réseau sous forme d'un graphe qui relie l'ensemble des clusters formés. Cette première structure est utilisée pour calculer les routes vers toutes les destinations du réseau. La deuxième structure est utilisée pour optimiser la diffusion des messages de contrôle liés au processus de routage. Elle représente une dorsale virtuelle de diffusion

et elle est formée par un sous ensemble dominant connecté.

6.3.1 Construction de la structure des clusters

Cette entité de construction de la structure des clusters permet de partitionner le réseau en un ensemble de clusters disjoints. Ces clusters formeront la topologie globale du réseau utilisée dans la circulation des données. Intégré dans le protocole de routage CLSR, ce processus doit être simple pour minimiser le temps de calcul. Il doit aussi converger rapidement pour ne pas retarder la transmission des données. Il doit également optimiser l'utilisation de la bande passante en réduisant le nombre de clusters dans le réseau et le nombre de changements de la topologie. Ces raisons nous ont conduit à utiliser l'algorithme de clusterisation SSCA, décrit dans le chapitre 3, et qui a été crée pour ce cadre d'utilisation.

Nous rappelons brièvement le principe de l'algorithme SSCA. C'est un algorithme à un saut qui utilise le degré de connectivité comme métrique de sélection des cluster-heads. Chaque nœud peut être dans l'un des trois états suivants : non-décidé, cluster-head ou membre. Lorsqu'un nouveau nœud se connecte au réseau, il passe à l'état non-décidé. Tout d'abord, il doit attendre un temps supérieur à deux fois l'intervalle d'envoi des messages HELLO pour découvrir son voisinage à deux sauts. La formation des clusters s'appuie sur les règles suivantes :

1. Un nœud qui a le plus grand nombre de voisins non-décidés se déclare cluster-head.
2. Un nœud qui est voisin à un cluster-head devient membre de ce cluster.

La maintenace des clusters applique les règles suivantes :

1. Quand un nouveau lien est détecté entre deux cluster-heads, le cluster-head ayant la faible cardinalité cède son rôle. Il essaye alors, avec ses membres, de joindre la structure des clusters en appliquant les mêmes règles de formation des clusters.
2. À la suite de la perte d'un lien entre un membre et son cluster-head, ce membre quitte le cluster et tente de joindre un autre cluster.

6.3.2 Construction de la dorsale virtuelle

Dans le protocole CLSR, la dorsale virtuelle est utilisée pour optimiser la diffusion des paquets de contrôle dans le réseau. Cette structure est composée d'un sous-ensemble dominant connecté (CDS). Elle profite de la caractéristique du support physique de transmission qui fonctionne en mode partagé, pour réduire le nombre de transmissions nécessaire à la diffusion des paquets de routage. En effet, tous les nœuds présents dans la portée d'une source peuvent écouter le message. Alors, seuls les nœuds de la dorsale virtuelle dominante retransmettent les paquets de routage.

La structure sous ensemble dominant d'un graphe est la structure la plus adaptée pour optimiser la diffusion des informations dans un réseau mobile sans fils. Cette structure est

composée d'un sous ensemble de nœuds tel que tout nœud du réseau est voisin d'au moins un nœud de ce sous ensemble. Le sous ensemble dominant est aussi connexe ce qui permet de garantir que chaque paquet est diffusé à tous les nœuds du réseau. Plusieurs algorithmes ont été proposés pour construire un sous ensemble dominant. Parmi ces algorithmes nous pouvons citer l'algorithme de Guha et Khuller [Guha and Khuller, 1998], l'algorithme de Alzoubi et Wan [Alzoubi et al., 2002] et l'algorithme de Wu et Li [Wu and Li, 2001]. Ces algorithmes sont simples et se basent sur des métriques différentes pour colorer les nœuds (dominant, dominé, autres). Par exemple, l'algorithme Guha et Khuller sélectionne le nœud avec le nombre maximum de voisins non-dominés comme un nœud dominant. Ensuite, il refait la sélection d'un autre nœud dominé avec la même condition jusqu'à que tous les nœuds soient dominés. D'autres heuristiques distribuées utilisent seulement des informations locales pour construire un CDS. La technique d'optimisation utilisée par OLSR ainsi que ses extensions basées sur la clusterisation repose sur le même principe appliqué localement. En effet, chaque nœud sélectionne un sous-ensemble appelé MPRs parmi ses voisins à 1-saut qui couvre tout son voisinage à 2-sauts. En effet, les travaux de [Wu et al., 2006] ont prouvé qu'il est plus efficace de sélectionner les nœuds MPRs en utilisant un voisinage plus étendu à K-sauts avec K supérieur à deux. Dans le cas de CLSR, les cluster-heads élus forment déjà un ensemble dominant qui couvre tout le réseau. Etant donné un cluster-head u, l'ensemble des cluster-heads voisins de u domine au moins son voisinage à 3-sauts. Par conséquent, il suffit de connecter les cluster-heads voisins à travers des nœuds dits passerelles pour former la dorsale virtuelle de diffusion.

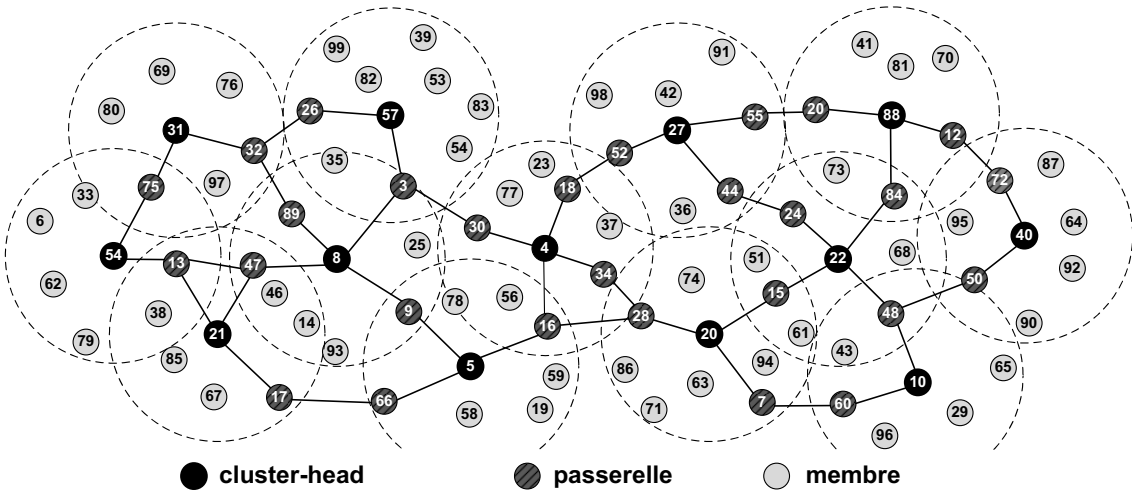


Figure 6.1 – Exemple de la Structure du réseau dans CLSR.

Dans la suite, nous présentons l'heuristique utilisée pour sélectionner les nœuds passerelles. Chaque cluster-head choisit ses propres passerelles qui lui permettent d'atteindre chaque cluster voisin. Puisque la sélection des passerelles est effectuée par les cluster-heads, chaque couple de cluster-heads voisins doit choisir la ou les mêmes passerelles pour ne pas avoir une discontinuité dans la dorsale virtuelle. Avec l'algorithme de clusterisation

SSCA, trois cas de connexion peuvent avoir lieu entre deux cluster-heads voisins. Ces cas dépendent de la distance qui sépare les deux cluster-heads voisins. En effet, ils sont soit directement liés, soit séparés par un seul nœud, soit séparés par deux nœuds. Par la suite et selon le cas, chaque cluster-head doit appliquer pour chaque cluster voisin l'une des trois actions suivantes :

---

① Dans ce cas les deux cluster-heads sont directement liés et aucune passerelle n'est nécessaire. Ces deux cluster-heads représentent des passerelles pour les autres nœuds du cluster.

② Dans ce cas les deux cluster-heads sont séparés par un seul nœud qui sera choisi comme passerelle. Si plusieurs nœuds sont candidats pour le rôle de passerelle, le nœud ayant la meilleure métrique (haut degré de connectivité et en cas d'égalité le plus faible identifiant) est sélectionné comme passerelle.

③ Dans ce cas les deux cluster-heads sont séparés par deux nœuds. Ces deux nœuds doivent être alors sélectionnés comme passerelles. Si plusieurs couples sont candidats pour ce rôle, le couple de nœuds ayant la meilleure métrique (somme du degré de connectivité la plus élevée et en cas d'égalité les deux plus faible identifiant) sera choisi.

Chaque cluster-head doit informer le reste de son cluster des différentes passerelles sélectionnées. Il diffuse alors dans son cluster un message *Gateway_Declaration* qui contient la liste des clusters voisins, ainsi que les passerelles vers chacun d'entre eux. Ce message permet aux nœuds sélectionnés de passer à l'état passerelle. Il permet aussi aux autres nœuds de connaître et mémoriser la liste de tous les clusters voisins de leur cluster et comment les atteindre. La dorsale virtuelle est formée des cluster-heads et des différentes passerelles sélectionnées. Ainsi, seuls les nœuds qui appartiennent à cette dorsale retransmettent les paquets de contrôle de topologie relatifs au protocole de routage CLSR.

La figure 6.1 montre un exemple d'une dorsale virtuelle utilisée pour la diffusion des données de contrôle. Cette dorsale est construite en utilisant l'heuristique proposée pour lier les cluster-heads. Arès avoir créer sa liste des clusters voisins, chaque cluster-head sélectionne sa liste de passerelle en appliquant les règles précédentes.

6.4 La gestion de la topologie

Chaque nœud du réseau possède deux vues différentes du réseau :

- La première vue est locale et elle contient les informations relatives à tous les nœuds voisins à 2-sauts. Cette topologie locale est représentée par un graphe G (V, E), où l'ensemble des sommets représente les nœuds du voisinage et l'ensemble des arcs représente les liens entre ces nœuds. Un lien entre deux nœuds représente le lien

physique de transmission, et par la suite deux nœuds qui sont dans la même portée de transmission sont liés.

- La topologie globale du réseau est aussi représentée par un graphe G' (V', E'). L'ensemble V' des sommets de ce graphe représente les clusters construits. Tandis que l'ensemble des arcs représente les liens entre ces clusters. D'après la définition 6.1, deux cluster-heads sont voisins si et seulement si, il existe au moins un lien physique de transmission entre un nœud du premier cluster et un nœud du deuxième.

Définition 6.1 Cluster-heads voisins

Soient deux cluster-heads U et V appartenant à G'. U est voisin de V si et seulement si :

$$\exists(x, y) \in Cl(U) \times Cl(V) \mid x \in N(y) \tag{6.1}$$

Notons que $Cl(U)$ représente l'ensemble des nœuds membres du cluster formé par le nœud cluster-head U et $N(y)$ représente l'ensemble des nœuds directement liés au nœud y (les nœuds voisins à 1-saut).
Pour maintenir ces deux vues de la topologie, chaque nœud maintient quatre structures de données qui sont :

- La liste des voisins directs ;
- La liste des voisins à 2-sauts ;
- La liste des clusters voisins ;
- La liste de tous les clusters du réseau.

À travers l'échange des messages HELLO, chaque nœud possède une connaissance complète de son voisinage à 2-sauts. Il maintient ainsi les trois premières listes citées au-dessus. Mais, ces informations ne permettent que l'acheminement des données vers des destinations au voisinage. Pour atteindre des destinations éloignées, CLSR prévoit d'appliquer un routage à état de liens entre les clusters. Ainsi, chaque nœud doit maintenir la topologie globale du réseau formée par l'interconnexion de tous les clusters. De ce fait, chaque cluster-head est invité à diffuser dans le réseau son état de liens avec ses clusters voisins. Il construit la liste de ses clusters voisins et les envoie dans un message CTC (Cluster Topology Control). Ce message contient les informations suivantes :

- L'identifiant du cluster source ;
- Un numéro de séquence ;
- La cardinalité du cluster ;
- La liste des nœuds membres du cluster ;
- Le nombre de clusters voisins ;
- La liste des clusters voisins ;

■ Le coût pour atteindre chaque cluster voisin.

Les paquets de gestion de la topologie assurent deux rôles qui sont la construction de la topologie globale des clusters et la construction d'une table de localisation des nœuds dans les différents clusters qui forment le réseau. Ces deux structures seront utilisées plus tard dans le processus de calcul de la table de routage et dans l'acheminement des paquets de données. Ces deux tâches sont réalisées à travers la diffusion des messages d'état de liens CTC.

Les messages CTC sont générés et envoyés seulement par les cluster-heads. Il sont diffusés dans tout le réseau à travers la dorsale virtuelle. Pour optimiser en plus la diffusion des messages CTC, la technique de groupage introduite dans OLSR est aussi utilisée. Ainsi, chaque nœud de la dorsale peut retarder la retransmission des messages CTC pour un temps réduit fixé à CLSR_CTC_DELAY pour permettre l'émission de plusieurs messages CTC dans le même paquet de contrôle.

À partir des messages CTC collectés, chaque nœud construit un graphe représentant la topologie globale des clusters. En plus, il construit une table de localisation à partir de la liste des membres contenue dans chaque CTC collecté. Cette table contient l'association entre tous les nœuds du réseau à leurs clusters. Elle permet de localiser le cluster auquel appartient une destination lors de l'acheminement des données.

6.5 Calcul de la table de routage

Le processus de routage permet de calculer les routes vers toutes les destinations du réseau. Il permet également l'acheminement des paquets de données tout le long de ces routes. Ces routes sont calculées à partir la connaissance de la topologie du réseau. En effet, chaque nœud du réseau maintient une topologie locale complète à 2-sauts et une topologie globale partielle du réseau.Deux types de routes sont alors gérées. Le premier type correspond à des routes où les destinations sont des nœuds du voisinage. Le deuxième type représente des routes vers des clusters-heads. Ces routes sont utilisées pour atteindre toutes les destinations qui sont en dehors du voisinage. Le calcul des routes suit les étapes suivantes :

1. Détruire toutes les entrées de la table de routage.
2. Ajouter dans la table de routage une entrée pour chaque voisin à au moins 2-sauts.
3. Ajouter une entrée pour chaque cluster de la liste des clusters voisins.
4. Ajouter une entrée pour chaque cluster du réseau.

6.5.1 Routage au voisinage

À partir de la topologie locale du voisinage à deux sauts, chaque nœud maintient une table de routage qui permet d'acheminer les paquets de données vers toutes les destina-

tions de ce voisinage. Chaque nœud calcule sa propre table de routage qui lui permet de communiquer avec les nœuds de son voisinage à deux sauts. Chaque entrée de cette table représente un chemin vers une destination de ce voisinage. Cette entrée est décrite par le tuple (Adresse_Destination, Distance, Nœud_Suivant) où Adresse_Destination représente l'identifiant du nœud destinataire, Distance représente la distance qui sépare ce nœud de la destination et enfin Nœud_Suivant représente le nœud suivant qui va permettre l'acheminement des informations.

Algorithme 1: Calcul_Routes_Voisinage

Données : N : Liste des voisins à 1-saut ;
N_2 : Liste des voisins à 2-sauts ;
$Mon\_Cluster$: L'adresse du cluster auquel appartient le nœud ;
Résultat : TR : Table de Routage ;

```
1  début
2      pour U ∈ N faire
3          Ajouter_TR(U.Adresse, 1, U.Adresse) ;
4      pour chaque nœud U ∈ N faire
5          pour chaque nœud V ∈ N(U) faire
6              si V.Adresse ∉ TR alors
7                  Ajouter_TR(V.Adresse, 2, U.Adresse) ;
8              sinon
9                  E = Rechercher_Entrée(TR, V.Adresse) ;
10                 si (E.Distance > 2) alors
11                     MAJ_TR(V.Adresse, 2, U.Adresse) ;
12                 si ((E.Distance = 2) ET (E.Nœud_Suivant est cluster-head)) alors
13                     MAJ_TR(V.Adresse, 2, U.Adresse) ;
14             si (V.CL_Adresse ≠ Mon_Cluster) alors
15                 si V.CL_Adresse ≠ V.Adresse alors
16                     si V.CL_Adresse ∉ TR alors
17                         Ajouter_TR(V.CL_Adresse, 3, U.Adresse) ;
18                     sinon
19                         E = Rechercher_Entrée(TR, V.CL_Adresse) ;
20                         si ((E.Distance = 3) ET (E.Nœud_Suivant est cluster-head))
                           alors
21                             MAJ_TR(V.CL_Adresse, 3, U.Adresse) ;
```

L'algorithme 1 *Calcul_Routes_Voisinage* décrit la procédure de calcul des routes à partir des informations de la topologie à deux sauts. Cet algorithme implémente les trois règles suivante :

---

① Ajouter une entrée pour chaque voisin direct dans la table de routage avec un coût égal à 1 (lignes 2-3).
② Ajouter une entrée pour chaque voisins à 2-sauts avec un coût égal à 2 (lignes 6-13).
③ Ajoute une entrée pour chaque cluster-head voisin qui n'est pas dans le voisinage à 2-sauts avec un coût égal à 3 (ligne 14-21).

Afin d'alléger la charge des cluster-heads lors de l'acheminement des paquets de données et si plusieurs chemins mènent vers la destination, le nœud choisit de passer par un nœud qui n'est pas un cluster-head. Cette action est inclut lors de l'application des règles 2 et 3. Elle est proclamée par les lignes 12 et 20 de l'algorithme 1 *Calcul_Routes_Voisinage*.

Afin d'illustrer le calcul des routes au voisinage, nous utilisons l'exemple de topologie présenté dans la figure 6.2 qui montre la topologie locale du nœud numéro 1. Après que le nœud applique l'algorithme 1 *Calcul_Routes_Voisinage*, il obtient la table de routage décrite par le tableau 6.1. Nous distinguons trois types d'entrées dans cette table de routage.

- Des entrées avec un coût égal à 1 ajoutées pour tous les voisins directs {2, 23, 18, 7, 21, 17, 12}.
- Des entrées avec un coût égal à 2 ajoutées pour chaque voisin à 2-sauts {3, 34, 27, 16, 11, 33, 32, 19, 8, 4}.
- Des entrées avec un coût 3 pour tous les cluster-heads qui ne figurent pas dans le voisinage à 2-sauts et qui ont un nœud membre présent dans ce voisinage {14, 20, 9, 6}.

Ces routes à 3-sauts sont déduites implicitement. Étant donné que ces cluster-heads ne sont pas présents dans le voisinage à 2-sauts, alors la distance vers ces nœuds est strictement supérieure à 2. Comme ils admettent un nœud membre à 2-sauts et sachant que chaque membre est distant d'un seul saut de son cluster-head, la route utilisant ces membres est alors optimale et elle est égale à 3 sauts.

Dans cet exemple nous remarquons aussi que le nœud 1 choisit les nœuds 2, 12 et 21 comme nœud suivant pour atteindre respectivement les nœuds 27, 16 et 8 dans le but de ne pas augmenter la charge des cluster-heads 7 et 17.

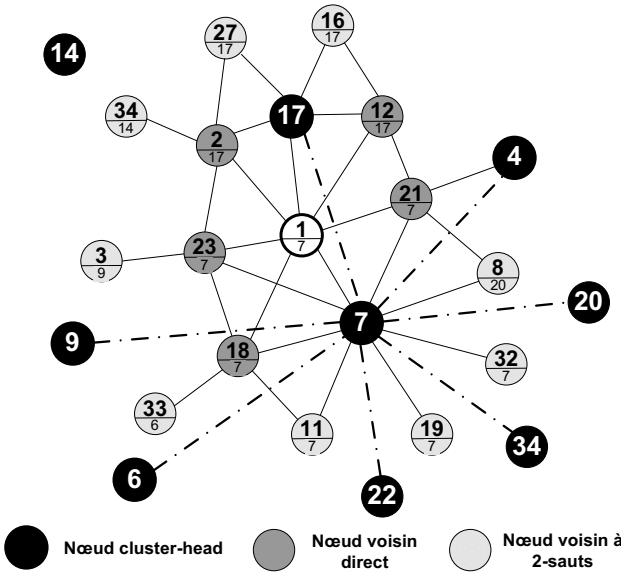


Figure 6.2 – Exemple de calcul des routes au voisinage du nœud 1

Adresse_Destination	Distance	nœud _Suivant
2	1	2
23	1	23
18	1	18
7	1	7
21	1	21
17	1	17
12	1	12
3	2	23
34	2	2
27	2	2
16	2	12
11	2	18
33	2	18
32	2	7
19	2	7
8	2	21
4	2	21
14	3	2
20	3	8
9	3	23
6	3	18

Table 6.1 – Table de routage au voisinage du nœud 1

6.5.2 Routage inter-cluster

Le routage en dehors du cluster est aussi un routage à état de liens. Chaque nœud doit calculer un chemin pour atteindre chaque cluster du réseau. Ces routes sont calculées à partir de la base des liens formée par les messages CTC reçus. Pendant le calcul des routes

dans son voisinage, chaque nœud vient d'ajouter une entrée pour chaque cluster-head détecté dans son voisinage à deux sauts. Les distances correspondantes à ces routes seront enregistrées dans le graphe global formé par tous les cluster-heads du réseau. En plus de ces entrées, chaque nœud applique l'algorithme de DIJKSTRA sur le graphe des clusters et ajoute une entrée pour chaque cluster-head du graphe.

Afin d'illustrer le calcul des routes vers l'ensemble des clusters, nous allons utiliser la topologie présentée dans la figure 6.3. Dans cette figure, nous avons représenté les voisins à 2-sauts du nœud 5 ainsi que tous les clusters du réseau. Après l'exécution de l'algorithme 1, le nœud 5 va ajouter dans sa table de routage des entrées qui correspondent à des cluster-heads. Ces cluster-heads sont détectés grâce aux informations du voisinage à 2-sauts. Ces entrées correspondent à ceux du tableau 6.2 et prennent en charge les clusters (2,15,3,12,7,9). En se basant sur ces entrées, le nœud 5 applique l'algorithme de DIJKSTRA sur le graphe des clusters (représenté par les cluster-heads et par les liens en pointillé dans la figure 6.3). Il ajoute une entrée dans la table de routage pour chaque cluster du graphe. Les entrées ajoutées après cette phase sont présentées dans le tableau 6.3.

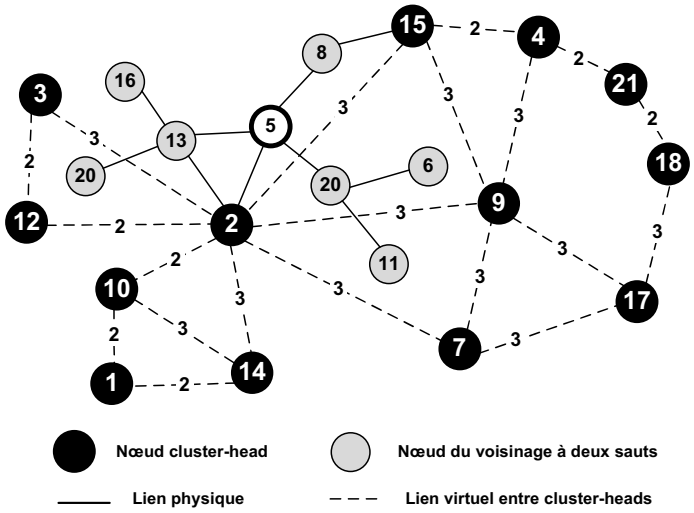


Figure 6.3 – Exemple de la vue de la topologie du réseau (par le nœud 5)

Adresse_Destination	Distance	Nœud_Suivant
2	1	2
15	2	8
3	3	13
12	3	13
7	3	20
9	3	20

Table 6.2 – Entrées vers cluster-heads calculées à partir du voisinage du nœud 5 de la figure 6.3.

Adresse_Destination	Distance	Nœud_Suivant
10	3	2
14	4	8
4	4	2
1	5	8
21	6	8
17	6	20
18	8	20

Table 6.3 – Entrées calculées à partir du graphe des clusters du nœud 5 de la figure 6.3.

6.6 Acheminement des paquets

Le processus d'acheminement des données est responsable à acheminer les paquets de données de la source à la destination. Tous les nœuds du réseau ont assez d'informations sur la topologie pour acheminer les paquets vers toutes les destinations grâce aux messages CTC. Chaque nœud, en plus de sa table de routage ordinaire, maintient une table de localisation qui contient une association entre le nœud et son cluster. Quand un nœud intermédiaire reçoit un paquet de données, il commence par chercher dans sa table de routage une entrée correspondante à un nœud de son voisinage. S'il trouve une entrée correspondante à l'adresse de destination, il envoie le paquet au prochain saut. Autrement, il cherche l'adresse du cluster de la destination dans sa table de localisation. Si la destination est localisée, le nœud intermédiaire cherche une entrée qui correspond au cluster auquel appartient la destination parmi les entrées des routes inter-cluster. Il envoie alors le paquet au prochain saut. Quand la destination n'est pas accessible, le paquet de données est supprimé.

6.7 Analyse expérimentale

Dans cette partie nous évaluons par simulation les performances de notre proposition CLSR en utilisant le simulateur des réseaux NS-2 (Network Simulator) [Doe]. Dans ce cadre, nous comparons les résultats de notre proposition à des protocoles appartenant à la même famille des protocoles à état de liens et qui ont visé le problème de scalabilité. Nous avons choisi les protocoles F-OLSR [Clausen, 2003], OLSR-Trees [Baccelli, 2006] et SA-OLSR [Canourgues et al., 2008]. Pour toutes les expériences réalisées, nous utilisons le standard IEEE 802.11 comme modèle de la couche liaison. Nous supposons également que le support de transmission radio possède une portée de transmission de 250 m et il a un débit de 2 Mb/s. Nous supposons aussi que tous les nœuds du réseau ont les mêmes capacités en termes de mémoire tampon et d'autres ressources.

6.7.1 Paramètres de simulation

Dans cette évaluation nous définissons la densité des nœuds comme étant le nombre de nœuds dans une surface unitaire. Cette surface correspond à celle d'un cercle de rayon égal à la portée de transmission du support radio 250 m. La densité des nœuds correspond aussi au nombre moyen de voisins directs à un seul saut pour chaque nœud du réseau.

Pour évaluer les différents protocoles de routage nous avons choisi d'utiliser le modèle de mobilité RWP (Random WayPoint) [Johnson and Maltz, 1996] et un trafic CBR (Constant Bit Rate) comme couche application. Ces deux modèles sont les plus utilisés dans les travaux similaires d'évaluation des protocoles de routage tels que [Boukerche, 2004, Broch et al., 1998, Das et al., 2000].

Les nœuds sont uniformément distribués dans une surface carrée. Ils se déplacent conformément au modèle RWP dans cette surface. Tel que dans [Das et al., 2000], la vitesse des nœuds est uniformément répartie dans l'intervalle [0.1-0.3] m/s pour le cas d'une faible mobilité et dans l'intervalle [1-4] m/s pour le cas d'une moyenne mobilité.

Comme couche applicative, nous utilisons un trafic de données constant CBR. Comme le débit du réseau est réduit, il ne peut pas supporter un nombre important de connexions CBR. Etant donné que le nombre de nœuds est assez important et que tous les nœuds dans le réseau ont un besoin de transmission, nous avons choisi de limiter la durée de chaque connexion. Elle est alors fixée à 60 secondes. Les débuts des connexions CBR sont uniformément répartis dans l'intervalle [50-300] secondes sachant que la durée de la simulation est fixée à 400 secondes. Ce choix permet aux différents protocoles de routage de s'organiser et de calculer les routes. Il permet également à toutes les connexions de se terminer avant la fin de la simulation. Les couples source-destination des connexions de données sont sélectionnés d'une façon aléatoire parmi les nœuds du réseau. Tous les paquets de données ont une taille fixe égale à 512 octets. Les connexions CBR sont générées avec deux différentes valeurs de débit. Dans le cas d'un nombre réduit de connexions le débit est fixé à 4 paquets/secondes. Par contre, le débit est fixé à 2 paquets/secondes pour le cas d'un nombre important de connexions.

Nous avons distingué deux cas d'expérimentation :

- Cas 1 : Dans ce cas d'expérimentation, nous fixons la densité des nœuds à une valeur moyenne égale à 15 nœuds. Ces nœuds se déplacent dans une surface carrée avec une faible mobilité. Le trafic de données applicatif est fixé à 100 connexions CBR avec un débit réduit. Pour ces valeurs fixes, nous générons plusieurs cas de réseaux en variant le nombre de nœuds dans chaque réseau. Les métriques de comparaison seront représentés en fonction du nombre de nœuds.
- Cas 2 : Dans ce cas nous utilisons des réseaux de taille fixe égale à 200 nœuds. Pour une moyenne valeur de la mobilité, nous varions la surface de la zone de simulation ce qui permet d'obtenir différentes valeurs de la densité des nœuds. Le trafic de données est composé de 50 connexions CBR avec un débit de 4 paquets/secondes.

6.7.2 Métriques d'évaluation

Pour comparer les différents protocoles de routage, nous avons choisi les métriques les plus couramment utilisées dans ce type de comparaison [Boukerche, 2004, Broch et al., 1998, Das et al., 2000]. Les métriques choisies sont les suivantes :

1. ***Le ratio de livraison des paquets :*** cette métrique correspond au taux de livraison des paquets de données. Elle est calculée par la division du nombre de paquets de données reçus par les destinations finales par le nombre de paquets envoyés par l'ensemble des sources.
2. ***La taille du trafic de contrôle (Overhead) :*** L'*overhead* dénombre les paquets de routage envoyés par le protocole pour le calcul et la maintenance des routes. Cette métrique est mesurée par le nombre de paquets de routage envoyés tout au long de la simulation par tous les nœuds du réseau.
3. ***Le délai moyen de bout-en-bout :*** c'est la moyenne du temps passé par chaque paquet pour être acheminé de sa source jusqu'à sa destination finale. Ce délai inclut le temps de traitement des paquets, le temps passé dans les files d'attente et le temps de propagation entre tous les nœuds intermédiaires.
4. ***Le nombre moyen de retransmissions des paquets :*** Cette métrique correspond au nombre moyen de sauts traversés par chaque paquet de données. Elle est mesurée par la division du nombre total de retransmissions des paquets de données par le nombre de paquets envoyés.

6.7.3 Trafic de contrôle *(Overhead)*

L'*overhead* est le trafic de contrôle ajouté par le protocole de routage afin de calculer et maintenir les routes d'acheminement des données. Il est représenté par le nombre de paquets de routage envoyés par le protocole. L'*overhead* est une métrique importante dans la comparaison des protocoles de routage. Il est la clé de la scalabilité des protocoles de routage et indique la capacité d'un protocole de fonctionner dans des réseaux larges, congestionnés ou avec une bande passante limitée. En effet, les protocoles qui envoient un grand nombre de paquets de routage utilisent plus la bande passante. Ainsi, ils augmentent la probabilité de collision des paquets et augmentent par la suite la perte des paquets de données.

La figure 6.4 représente les résultats de l'expérimentation 1 décrite dans la section 6.7.1. Elle trace l'*overhead* généré par les protocoles de routage en fonction du nombre de nœuds dans le réseau. A première vue, nous remarquons que l'*overhead* croît d'une façon monotone quand le nombre de nœuds augmente pour les quatre protocoles de routage. Malgré son optimisation temporelle de l'envoi des messages TC, le protocole F-OLSR génère le plus d'*overhead*. En effet, sa structure plate et le coût énorme de diffusion de chaque message TC pour chaque nœud du réseau demeure son handicape. L'utilisation de la technique

de clusterisation dans les trois autres protocoles permet de réduire considérablement le nombre de messages d'état de liens diffusés dans tout le réseau. L'utilisation des messages Super-Hello pour la découverte des clusters voisins par OLSR-Trees ajoute une quantité importante d'*overhead*. Ceci le désavantage par rapport aux deux autres protocoles SA-OLSR et CLSR. L'utilisation des messages HELLO pour la clusterisation ainsi que la suppression des messages d'état de liens à l'intérieur des clusters permettent à notre proposition CLSR de générer la plus faible quantité d'*overhead*.

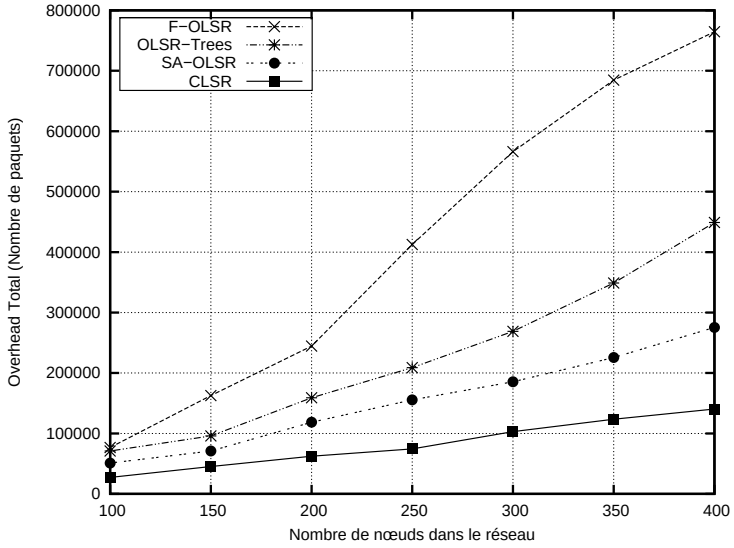


Figure 6.4 – Trafic de contrôle du routage en fonction du nombre de nœuds.

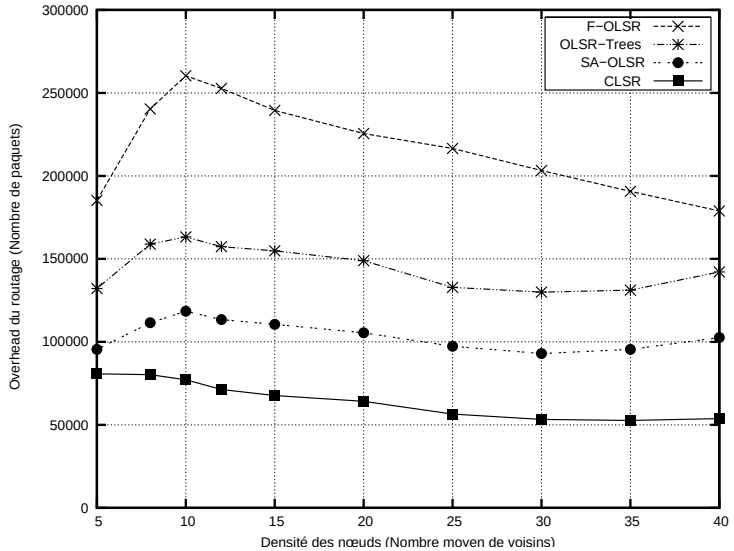


Figure 6.5 – Trafic de contrôle du routage en fonction de la densité des nœuds.

La figure 6.5 illustre le trafic de contrôle résultant de l'expérimentation 2. Elle représente l'*overhead* généré par les quatre protocoles de routage en fonction de la densité des nœuds. Le même ordre constaté dans l'expérimentation 1 est observé. Le protocole F-OLSR

avec sa nature plate génère l'*overhead* le plus élevé. Les optimisations introduites par le protocole CLSR lui permettent de générer l'*overhead* le plus réduit. Quand la densité est faible (5 nœuds) le réseau n'est pas assez connexe, par conséquent il est possible de trouver des parties isolées dans le réseau. L'augmentation de la densité des nœuds renforce la connectivité du réseau qui devient de plus en plus connexe. Ces raisons expliquent l'augmentation observée de l'*overhead* des quatre protocoles de routage entre les valeurs 5 et 10 nœuds de la densité. Au-delà de la densité 10 nœuds, le réseau étant connexe, nous remarquons que l'*overhead* diminue quand la densité augmente. En effet, l'élévation de la valeur de la densité pour un nombre fixe de nœuds implique la réduction de l'étendue du réseau. Cette réduction conjointement avec l'optimisation de la diffusion au niveau des quatre protocoles diminuent le coût de la diffusion des paquets de contrôle et ainsi abaissent la taille totale de ce trafic. Enfin, nous remarquons une augmentation de l'*overhead* pour les protocoles OLSR-Trees et SA-OLSR dans le cas d'une densité très élevée. Cette montée de l'*overhead* est expliquée par l'augmentation du nombre de paquets de contrôle diffusés à l'intérieur du cluster, vu que le nombre de nœuds dans chaque cluster a augmenté. En effet, à une très haute densité, le réseau sera composé d'un seul cluster et par la suite OLSR-Trees et SA-OLSR deviennent identiques à OLSR et F-OLSR. Cette augmentation de l'*overhead* à haute densité n'est pas observée chez le protocole CLSR puisqu'il élimine les messages de contrôle à l'intérieur des clusters.

6.7.4 Ratio des paquets délivrés

L'acheminement des paquets de données est la tâche principale d'un protocole de routage. De ce fait, le taux de livraison des paquets est le paramètre le plus important dans la comparaison des protocoles de routage. Cette métrique caractérise combien un protocole de routage est complet, correct et efficace. Elle décrit la quantité de perte de paquets ainsi que la quantité maximale de paquets que peut supporter le réseau.

Dans la figure 6.6 nous représentons le ratio de livraison de paquets pour chaque protocole en fonction du nombre de nœuds dans le réseau (expérimentation 1). Nous remarquons que les performances des quatre protocoles en terme de livraison des paquets de données diminuent lorsque le nombre de nœuds augmente. Ce résultat est lié à l'augmentation de l'*overhead* généré par ces protocoles. En effet, l'augmentation de l'*overhead* augmente la probabilité de collision entre les paquets. Par la suite, le risque de perte des paquets de données augmente ce qui diminue le taux de livraison de paquets. Le protocole F-OLSR qui génère le plus grand *overhead* est le plus affecté. Son taux de livraison diminue brusquement lorsque le nombre de nœuds dépasse les 200 nœuds. La réduction de la fréquence d'envoi des messages d'état de liens aux destinations les plus éloignées permet aussi de réduire les performances de F-OLSR. En effet, le risque de perte des messages d'état de liens augmente aussi avec l'augmentation de l'*overhead* et l'augmentation des distances. Ainsi, les sources qui désirent communiquer avec des destinations éloignées risquent aussi

de ne pas pouvoir déterminer des routes en absence des messages d'état de liens. Grâce à son faible *overhead*, CLSR permet de livrer le plus grand nombre de paquets de données. L'utilisation des arbres créées par OLSR-Trees dans l'acheminement des données, sollicite plus les nœuds appartenant à cette structure. Par conséquent, des goulets d'étranglement peuvent apparaître. Ceci explique donc son faible taux de livraison de paquets par rapport aux protocoles SA-OLSR et CLSR. Cette sollicitation est aussi remarquée dans le cas du protocole SA-OLSR pour les nœuds MPRs mais elle a un moindre effet. Ceci peut être expliquer par le nombre et la dynamique de ces nœuds MPRs ainsi que leur choix individuel par chaque nœud.

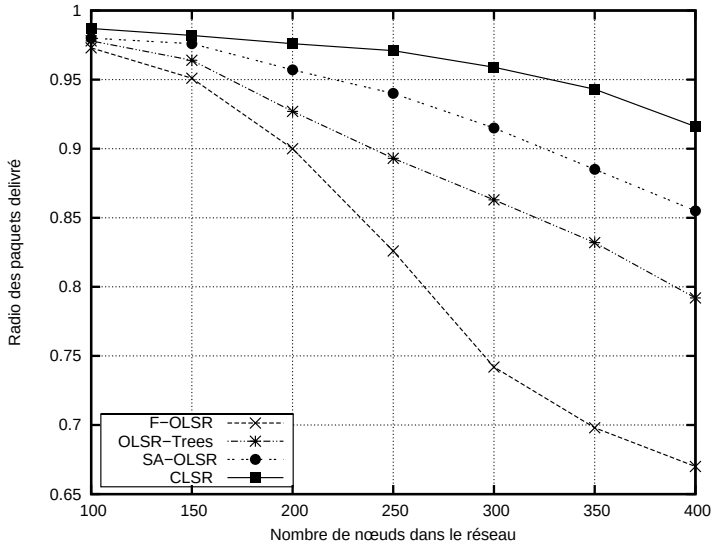


Figure 6.6 – Taux de livraison des paquets en fonction du nombre de nœuds.

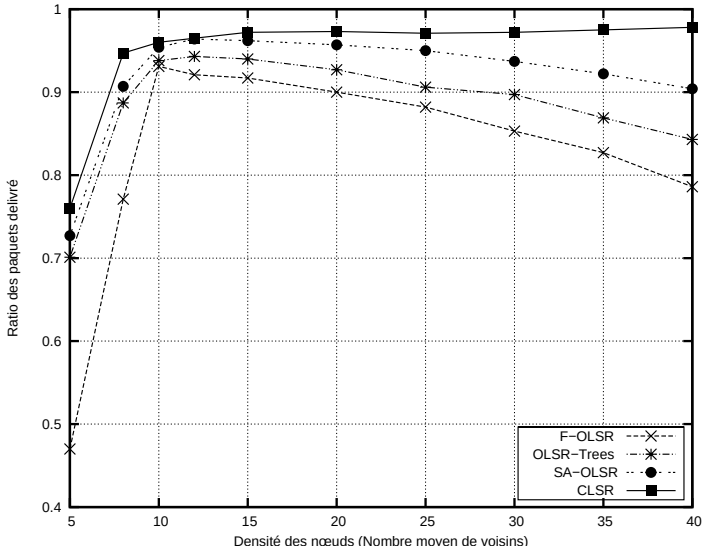


Figure 6.7 – Taux de livraison des paquets en fonction de la densité des nœuds.

Dans la figure 6.7, nous présentons le ratio des paquets délivrés en fonction de la densité

des nœuds dans le cas des scénarios de l'expérimentation 2. Nous remarquons que pour une faible densité (égale à 5 nœuds) les quatre protocoles donnent un faible ratio des paquets délivrés. En effet, comme la connectivité du réseau est faible étant donné que la densité est faible, les protocoles ne trouvent pas des routes pour atteindre certaines destinations et spécialement ceux qui sont isolés. Le protocole F-OLSR est le protocole le plus affecté avec un ratio inférieur à 50%. Comme ce protocole utilise la réduction temporelle de la fréquence des mises à jour des messages de contrôle avec la distance, la découverte de nouvelles routes sera assez retardée surtout pour les nœuds éloignés. Pour des valeurs moyennes de la densité (entre 10 et 15 nœuds), les quatre protocoles produisent des valeurs assez proches du ratio des paquets délivrés. Ce ratio est très intéressant et dépasse largement 90% des paquets envoyés. Cependant, l'augmentation du ratio des paquets délivrés est suivie par une baisse observée avec l'augmentation de la densité. Cette baisse est remarquée chez tous les protocoles sauf CLSR qui garde un ratio quasiment constant pour les valeurs de densité considérées par les expérimentations (jusqu'à 40 nœuds). En effet, pour les protocoles F-OLSR, OLSR-Trees et SA-OLSR, à haute densité chaque nœud est tenu à faire suivre plus de trafic de contrôle. Ce trafic de contrôle fait augmenter le taux de collision et gène le trafic de données et par conséquent la perte des paquets de données augmente. Grâce à son faible trafic de contrôle à haute densité, le protocole CLSR réussit à garder un ratio de paquets délivrés quasiment constant.

6.7.5 Délai de bout-en-bout

Le délai de bout en bout caractérise le fonctionnement d'un protocole de routage. Il représente le temps nécessaire pour acheminer un paquet de sa source jusqu'à atteindre sa destination finale. Il est important pour certaines applications telles que les applications multimédia et en temps réel. Dans notre expérimentation, ce temps est calculé comme la moyenne des temps de réponse de chaque connexion de données CBR.

Nous représentons les valeurs mesurées du délai en fonction du nombre de nœuds dans le réseau dans la figure 6.8 (expérimentation 1). Nous remarquons que le délai augmente en fonction du nombre de nœuds. En effet, puisque les scénarios sont générés avec une même densité des nœuds la surface de simulation augmente avec le nombre de nœuds. Par la suite, les distances qui séparent les nœuds augmentent ce qui influe directement sur l'augmentation du temps de réponse. Ce facteur n'est pas le seul qui provoque l'augmentation du temps de réponse. La priorité du trafic de routage par rapport au trafic de données est aussi un facteur majeur dans la dégradation du temps de réponse. Lorsque le trafic de routage augmente, les paquets de données sont de plus en plus retardés dans les files d'attente des nœuds intermédiaires ce qui se répercute directement par une augmentation énorme du temps de réponse. Ce facteur est la cause principale du grand délai donné par le protocole F-OLSR. Étant donné que F-OLSR génère le plus grand *overhead*, ce protocole produit le plus grand délai comparé aux autres protocoles. D'autre part, la sollicitation de certains nœuds par

rapport à d'autres dans l'acheminement des paquets de données peut engendrer des goulets d'étranglement dans les chemins de données. Ainsi, le délai de livraison des paquets augmente considérablement. Cet effet explique l'important temps de réponse engendré par le protocole OLSR-Trees. Les protocoles SA-OLSR et CLSR génèrent des délais proches et nettement inférieurs à ceux enregistrés par les protocoles F-OLSR et OLSR-Trees. Grâce à son plus faible *overhead* et la diversité des chemins utilisés, CLSR produit le plus faible délai de bout-en-bout.

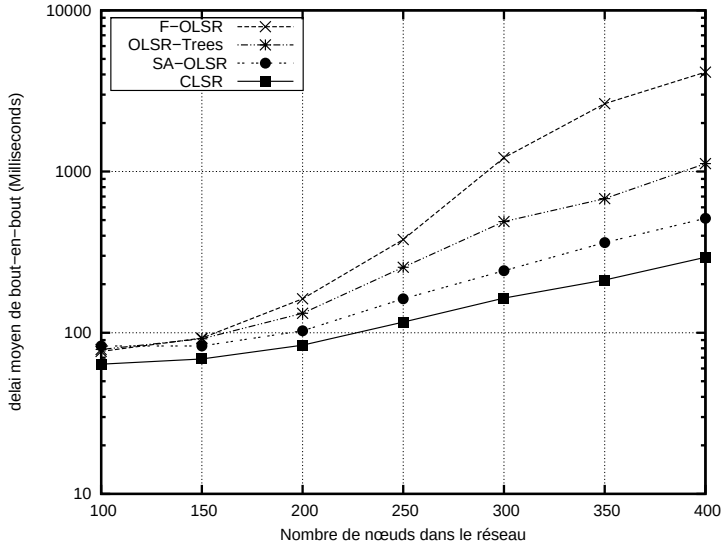


Figure 6.8 – Délai de bout en bout en fonction du nombre de nœuds.

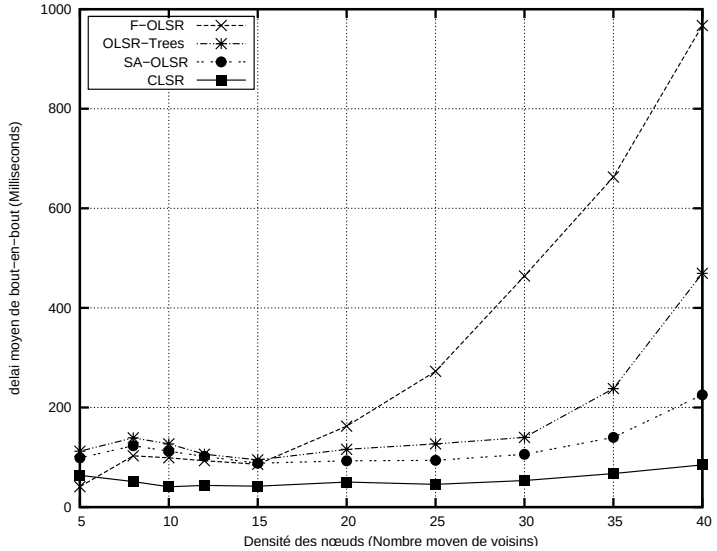


Figure 6.9 – Délai de bout en bout en fonction de la densité des nœuds.

La figure 6.9 présente le délai moyen de bout-en-bout mesuré dans le cadre de l'expérimentation 2 en fonction de la densité des nœuds. Comme nous l'avons remarqué deux facteurs majeurs interviennent dans les mesures du délai qui sont les longueurs des chemins

et le temps passé dans les files d'attente. L'augmentation de la densité réduit la moyenne des longueurs de chemins empruntés et par conséquent le délai devrait se baisser. Cette perception est observée dans le cas d'une moyenne densité (comprise entre 8 et 15 nœuds). Par contre, le deuxième facteur commence à dominer à partir de la densité 20 nœuds. Son impact est nettement observé pour les hautes valeurs de la densité des nœuds. En effet, à haute densité chaque nœud est tenu à faire suivre plus de trafic de contrôle. Étant donné que le trafic de contrôle est plus prioritaire, il va gêner le trafic de données et causer des importants retards passés dans les files d'attente. Ceci explique la prolifération du délai moyen de bout-en-bout observé dans la deuxième partie des courbes de la figure 6.9. Le protocole F-OLSR est le plus affecté par le trafic de contrôle et engendre le délai le plus élevé. L'*overhead* et la sollicitation des nœuds des arbres sont la cause de l'important délai enregistré par OLSR-Trees. Malgré qu'il prévoit l'utilisation des chemins les moins congestionnés, SA-OLSR est aussi affecté par les attentes dans les files des nœuds composant le chemin. En effet, les chemins choisis restent valides pendant toute la durée de l'intervalle séparant deux messages de contrôle de la topologie (TC-Cluster) ce qui ne garantit pas que ces chemins restent les moins congestionnés et les plus rapides. Enfin, le protocole CLSR est le plus résistant aux effets de l'augmentation de la densité grâce à son faible *overhead* et il propose les meilleures performances en terme de délai moyen de bout-en-bout.

6.7.6 Nombre moyen de sauts

Le nombre moyen de sauts traversés par chaque paquet de données exprime l'efficacité de l'utilisation de la bande passante du réseau par le protocole de routage. Cette métrique donne aussi une indication sur la longueur des routes établies par chaque protocole dans l'acheminement des données. Elle montre la capacité du protocole de sélectionner les plus courts chemins dans le réseau. En plus, couplée avec le taux de livraison de paquets elle peut donner une indication sur la capacité d'un protocole d'acheminer les paquets de données vers les destinations les plus éloignées.

La figure 6.10 illustre le nombre moyen de retransmissions des paquets de données enregistré par les quatre protocoles dans le cas de l'expérimentation 1. Nous remarquons que le protocole OLSR-Trees engendre le plus grand nombre moyen de retransmissions des paquets de données. Ce résultat est attendu puisque OLSR-Trees ne cherche pas les chemins optimaux, mais il utilise la structure des arbres créée dans l'acheminement des paquets. SA-OLSR s'appuie sur la première copie reçue des messages de topologie TC-Cluster pour calculer les routes vers les clusters. Ces routes sont supposées être les plus rapides et les moins congestionnées. Ce choix induit à des routes qui ne sont pas nécessairement optimales en terme de nombre de sauts traversés. Ceci explique le nombre de retransmissions le plus élevé produit par SA-OLSR par rapport aux résultats fournis par F-OLSR et CLSR. Les protocoles F-OLSR et CLSR calculent les routes optimales en terme du nombre de sauts. Cette propriété permet à ces deux protocoles d'optimiser l'utilisation de la bande

passante du réseau en réduisant le nombre de retransmissions de paquets. Cependant, c'est le protocole F-OLSR qui produit le nombre de retransmissions de paquets le plus réduit. Ce résultat n'est pas seulement lié à l'optimalité des routes calculées. En effet, l'association de ce résultat avec le taux assez faible de livraison de paquets de données de F-OLSR nous permet de déduire que ce protocole a des difficultés d'acheminer les paquets de données vers des destinations très éloignées des sources.

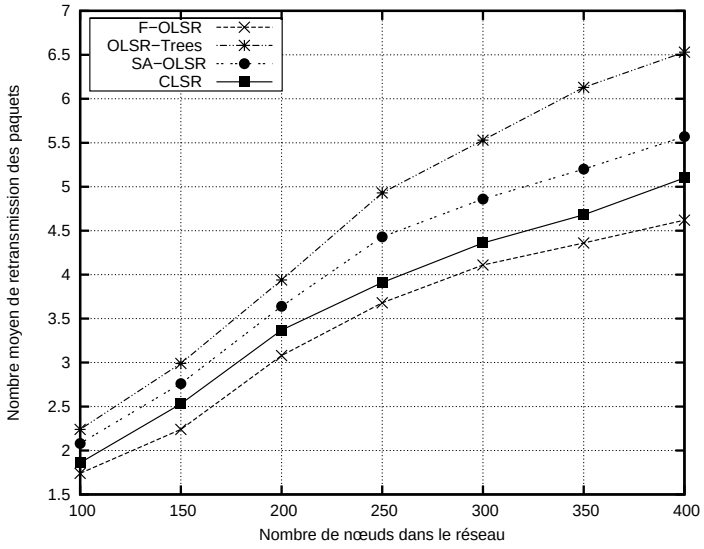


Figure 6.10 – Nombre moyen de retransmissions des paquets en fonction du nombre de nœuds.

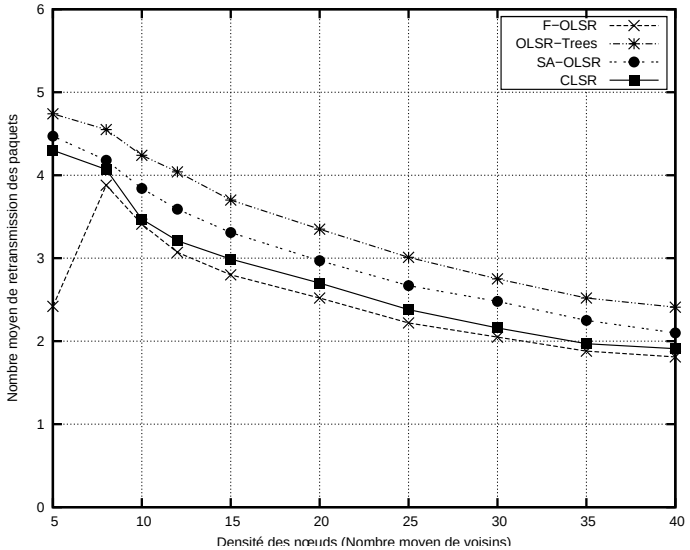


Figure 6.11 – Nombre moyen de retransmissions des paquets en fonction de la densité des nœuds.

La figure 6.11 représente le nombre moyen de retransmissions des paquets de données en fonction de la densité (expérimentation 2). En premier, nous remarquons que le nombre

moyen de retransmission décroit avec l'élévation de la densité des nœuds. Cette observation est valide pour les quatre protocoles évalués. Ce résultat ne ressort pas du comportement des protocoles, mais il est lié à la caractéristique du réseau. En effet, pour un nombre fixe de nœuds l'étendue du réseau se réduit avec l'augmentation de la densité des nœuds. Par la suite, la moyenne des longueurs des chemins entre sources et destinations est diminuée. En plus, les mêmes constatations retenues dans le cas de l'expérimentation 1 restent valide. Ainsi, nous observons le même ordre des performances en terme du nombre moyen de retransmissions pour les quatre protocoles de routage. Enfin, nous distinguons un faible nombre de retransmissions enregistré par le protocole F-OLSR pour le cas de la faible densité égale à 5 nœuds. Ce résultat confirme notre constatation concernant le ratio de paquets délivrés pour cette même valeur de la densité. En effet, à faible densité F-OLSR trouve des difficultés pour acheminer les paquets de données aux destinations éloignées. Ces difficultés sont causées par la réduction des fréquences de l'envoi des messages de contrôle de la topologie envoyés aux nœuds éloignés.

6.8 Conclusion

Dans ce chapitre, nous avons présenté une proposition d'un protocole de routage à état de liens basé sur une structure de clusters appelé CLSR. Grâce à la suppression des informations redondantes dans les différents types de paquets de contrôle, le protocole CLSR réduit la taille du trafic de contrôle. Ainsi il améliore les performances du routage dans le cas des réseaux de grande taille. En effet, CLSR utilise seulement deux types de paquets. Le paquet HELLO est utilisé localement dans la structuration du réseau en un ensemble de clusters et dans le calcul des routes dans le voisinage local réduit à deux sauts. Un deuxième type de paquets (CTC) est diffusé dans tout le réseau par les cluster-heads. Il est utilisé pour établir les routes inter-cluster. Pour évaluer les performances de notre proposition, nous l'avons implémenté sous le simulateur NS-2 et nous avons comparé ses performances à ceux des protocoles F-OLSR, OLSR-Trees et SA-OLSR. Les résultats montrent que CLSR offre de meilleures performances en termes de livraison de paquets, de taille du trafic de contrôle, du nombre moyen de retransmissions des paquets et du délai moyen de bout en bout.

CHAPITRE 7

PROPOSITION D'UN PROTOCOLE DE ROUTAGE HIÉRARCHIQUE : HCLSR

INTRODUIRE une hiérarchie dans la structure des clusters utilisée par le protocole CLSR peut améliorer la scalabilité de ce protocole. Dans ce chapitre, nous proposons une version hiérarchique du protocole CLSR qui profite des avantages de ce dernier. Cette nouvelle version intègre aussi la technique d'espacement temporelle des messages de mise à jour de la topologie afin de réduire plus le trafic de contrôle.

Sommaire

7.1 Introduction 142
7.2 Structuration du réseau 142
7.2.1 Structuration du premier niveau 142
7.2.2 Clusterisation hiérarchique 142
7.3 Le routage dans HCLSR 144
7.3.1 La gestion de la topologie 144
7.3.2 Calcul des routes 146
7.3.3 Acheminement des données 146
7.4 Analyse théorique 148
7.4.1 Borne supérieure du protocole F-OLSR 148
7.4.2 Borne supérieure du protocole SA-OLSR 149
7.4.3 Borne supérieure du protocole L-HCLSR 149
7.4.4 Comparaison des bornes supérieures 150
7.5 Analyse expérimentale 153
7.6 Conclusion 157

7.1 Introduction

Dans des réseaux très larges, malgré que CLSR génère un nombre réduit de cluster-heads, ce nombre reste relativement grand et devient un handicap pour l'algorithme de routage CLSR. En effet, pour le cas d'une densité égale à 10 nœuds, le nombre moyen de cluster-heads générés est supérieur à $\frac{1}{10}$ du nombre de nœuds du réseau. Par la suite, l'application du protocole CLSR à 1000 nœuds a des performances équivalentes aux performances de l'application d'un protocole plat à 100 nœuds. Afin de réduire le trafic de contrôle et d'augmenter la scalabilité du protocole CLSR, il sera intéressant de réduire le nombre de clusters par groupement des clusters proches dans un même cluster. Dans ce cadre, nous proposons une version hiérarchique du protocole CLSR nommé L-HCLSR (L-Hierarchical Cluster-based Link State Routing) [Guizani et al., 2011b, 2012b]. Dans cette proposition, les cluster-heads sont organisés dans L niveaux logiques de clusters. Le premier niveau est structuré de la même façon proposé par le protocole CLSR. Les cluster-heads élus dans ce premier niveau participent comme étant des cluster-heads ordinaires dans le second niveau et parmi eux seront sélectionnés des cluster-heads de niveau 2. Suivant le même principe, les cluster-heads de niveau K participent à l'élection des cluster-heads de niveau $K + 1$ jusqu'à construire le dernier niveau d'ordre L.

7.2 Structuration du réseau

7.2.1 Structuration du premier niveau

La structuration du premier niveau se fait de la même façon décrite dans le protocole CLSR. Chaque nœud procède à une découverte de voisinage à travers l'échange des messages HELLO. Il décide de son état en appliquant l'algorithme de clusterisation α-SSCA. Enfin, chaque cluster-head sélectionne ses passerelles et les informe à travers le message *Gatway_Declaration*.

7.2.2 Clusterisation hiérarchique

Pour créer la structure hiérarchique, les cluster-heads du niveau K deviennent des nœuds ordinaires de niveau $K + 1$. Parmi ces nœuds sera élu un ensemble des cluster-heads de niveau $K + 1$ en utilisant le même algorithme de clusterisation α-SSCA décrit dans le paragraphe 3.2. Par conséquent, tous les cluster-heads de ce niveau sont invités à découvrir leur voisinage à 2-sauts. Ainsi, chaque cluster-head diffuse à travers la dorsale virtuelle un message spécifique de contrôle appelé CTC (Cluster Topology Control). Les messages CTC sont utilisés pour assurer deux rôles. Ils permettent aux cluster-heads de découvrir et suivre l'état de leurs liens avec les cluster-heads voisins ce qui permet d'élire les cluster-heads du niveau suivant. Ces messages sont aussi utilisés pour décrire les liens entre

les cluster-heads de même niveau. Ainsi, ils permettent aux nœuds du réseau de construire la topologie globale du réseau et calculer les différentes entrées de leurs tables de routage.

Pour chaque niveau K correspond un message CTC d'ordre K. Ces messages $CTC(K)$ sont générés et envoyés seulement par les cluster-heads de niveau K. Ils contiennent les informations suivantes :

- K : L'ordre du message ;
- L'adresse du cluster-head source ;
- La liste des membres du cluster de niveau K ;
- L'adresse du cluster de niveau $K + 1$ auquel appartient la source ;
- La liste des cluster-heads voisins dans le niveau K ;
- L'adresse du cluster de chaque cluster-head voisin dans le niveau K.

Nous remarquons que la structure du message CTC est très proche du message HELLO envoyé par tous les nœuds dans le premier niveau. Pour appliquer l'algorithme de clusterisation au niveau K, il suffit que les cluster-heads de ce niveau effectuent un échange de message $CTC(K)$. Nous savons déjà que deux cluster-heads du premier niveau sont séparés au maximum par trois sauts. Par la suite, les messages $CTC(1)$ doivent être envoyés vers tous les nœuds qui sont à trois sauts pour garantir que tous les cluster-heads voisins dans ce premier niveau reçoivent ces informations. D'où, chaque message $CTC(1)$ sera envoyé avec un $TTL = 3$.

Dans le cas général, deux cluster-heads de niveau K sont à une distance maximale égale à 3^K. Par conséquent, étant donné un niveau K, les messages $CTC(K)$ seront envoyés avec $TTL = 3^K$ pour atteindre tous les cluster-heads voisins dans ce niveau. Après l'échange des messages $CTC(K)$, le cluster-head du niveau K ayant le plus grand nombre de voisins non-décidés devient cluster-head dans le niveau $K+1$. Les voisins de ce cluster-head deviennent des nœuds membre de son cluster dans le niveau $K + 1$.

La figure 7.2 présente la topologie du niveau 2 formée par les cluster-heads du niveau 1 qui sont déjà présentés dans l'exemple de la figure 7.1. Elle illustre l'élection des cluster-heads de niveau 2. Le nœud 5 est un cluster-head du niveau 1 et il a la plus grande connectivité dans son voisinage. Ainsi, il devient cluster-head de niveau 2 et les cluster-heads de niveau 1 qui lui sont voisins comme les nœuds 2, 20 et 8 deviennent membres de son cluster dans le niveau 2.

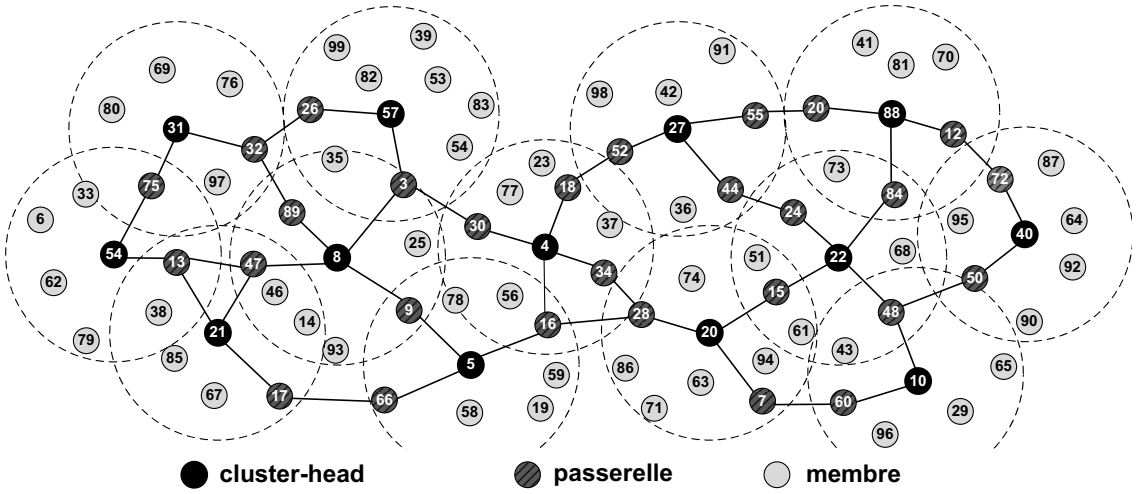


Figure 7.1 – Exemple de structuration du premier niveau.

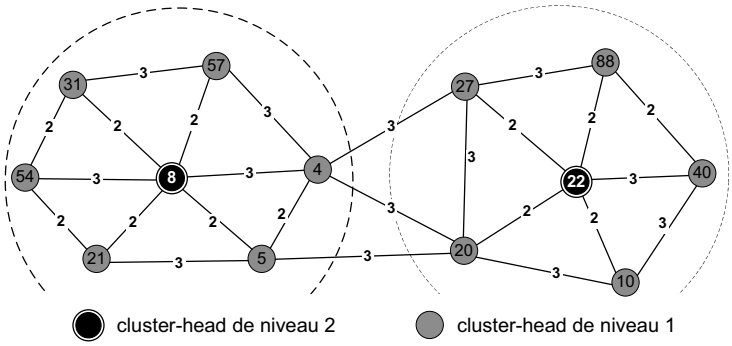


Figure 7.2 – Exemple d'élection de cluster-heads de niveau 2.

7.3 Le routage dans HCLSR

7.3.1 La gestion de la topologie

Dans L-HCLSR, les nœuds sont organisés dans L niveaux hiérarchiques. Tous les nœuds envoient périodiquement le message HELLO vers tous leurs voisins. Chaque cluster-head de niveau $K \in [1-L]$ envoie périodiquement K messages CTC d'ordre $K \in [1-L]$. Enfin, tous les messages CTC d'ordre $K \in [1-(L-1)]$ sont diffusés à tous les voisins à 3^K et seuls les messages d'ordre L sont diffusés dans tout le réseau.

Pour réduire l'overhead du routage, le protocole L-HCLSR intègre aussi la technique Fisheye. Par la suite, les destinations les plus proches reçoivent plus fréquemment les messages CTC que ceux les plus éloignées. En effet, un message CTC d'ordre K est diffusé plus fréquemment qu'un message CTC d'ordre $K+1$. Nous proposons la fonction $F(K) = 1/2K$ comme la fonction de la fréquence du message CTC d'ordre K. Par exemple, un cluster-head de niveau 3 doit envoyer trois messages CTC qui sont

$CTC(1)$, $CTC(2)$ et $CTC(3)$. L'envoi des messages $CTC(1)$ est séparé par l'intervalle $CTC\_Intervalle$. Tandis que les messages $CTC(2)$ et $CTC(3)$ seront séparés respectivement par les intervalles $2 \times CTC\_Intervalle$ et $3 \times CTC\_Intervalle$.

Nous savons déjà que dans un paquet IP la valeur maximale que peut prendre le champ TTL est égale à 255. Par conséquent, le nombre maximum de niveaux que nous pouvons définir est égal à 5 ($3^5 = 243$). Par la suite dans L-HCLSR, le paramètre L qui définit le nombre de niveaux considérés prend ses valeurs dans l'ensemble $\{1, 2, 3, 4, 5\}$. Le tableau 7.1 décrit les différentes caractéristiques des clusters de différents niveaux appartenant à $\{1, 2, 3, 4, 5\}$.

Niveau	1	2	3	4	5
TTL du message CTC	3	9	27	81	243
Distance entre deux cluster-heads voisins	2-3	4-9	10-27	28-81	82-234
Rayon maximal d'un cluster (sauts)	1	4	13	40	121
Nombre moyen des nœuds dans un cluster	M	16.M	160.M	1600.M	14641.M

Table 7.1 – Caractérisation des clusters dans les différents niveaux

Chaque nœud du réseau maintient un graphe représentant son voisinage local à 2-sauts construit à partir des messages HELLO collectés. L'information contenue dans les messages HELLO reçus permet aussi de créer la liste des clusters présents dans son voisinage. Chaque nœuds est aussi porté à maintenir un graphe composé des tous les clusters connus par ce nœud. Ce graphe est construit à partir de tous les messages CTC collectés par le nœud. En effet, ce graphe contient tous les cluster-heads de niveau K et qui sont à une distance inférieur à (2×3^K)-sauts du nœud. Il comporte aussi tous les cluster-heads de niveau L dans le réseau.

La figure 7.3 montre la connaissance de la topologie du réseau au niveau du nœud 1 pour le cas d'un nombre de niveau L égal à 2. Cette topologie contient une vue locale composée par tous les nœuds de son voisinage à 2-sauts, d'une vue partielle du niveau 1 composée par tous les cluster-heads de niveau 1 qui se trouvent à une distance inférieure à 6-sauts et composée par tous les cluster-heads du dernier niveau 2.

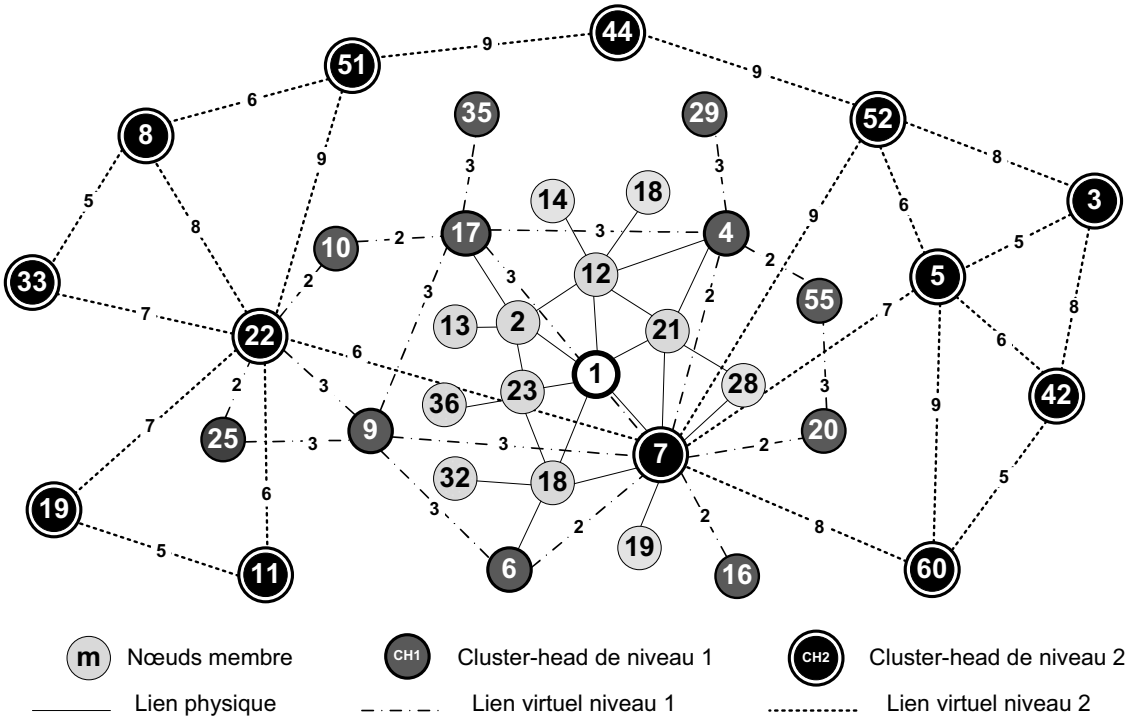


Figure 7.3 – Exemple de la connaissance de topologie au niveau du nœud (1).

7.3.2 Calcul des routes

Chaque nœud dans le réseau maintient une table de routage qui lui permet d'acheminer les paquets des données pour toutes les destinations dans le réseau. Cette table de routage contient une entrée pour chaque voisin dans le voisinage local à 2-sauts. Elle contient également une entrée pour chaque cluster-head connu. La table de routage est calculée comme suit :

1. Appliquer l'algorithme de DIJKSTRA sur le graphe des nœuds dans le voisinage local à 2-sauts. Ajouter une entrée pour chaque nœud du graphe.
2. Déterminer la liste des cluster-heads voisins.
3. Ajouter une entrée pour chaque cluster-head voisin.
4. Appliquer l'algorithme de DIJKSTRA sur le graphe des clusters. Ajouter une entrée pour chaque cluster-head du graphe.

7.3.3 Acheminement des données

Le processus d'acheminement des données est responsable d'acheminer les paquets de données de la source à la destination. Tous les nœuds du réseau ont assez d'information sur la topologie pour acheminer les paquets vers toutes les destinations grâce aux messages CTC. Chaque nœud, en plus de sa table de routage ordinaire, maintient une table de

localisation qui contient une association entre le nœud et son cluster. Quand un nœud intermédiaire reçoit un paquet de données, il commence par chercher dans sa la table de routage une entrée correspondante à un nœud de son voisinage. S'il trouve une entrée correspondante à l'adresse de destination, il envoie le paquet au prochain saut. Autrement, il cherche l'adresse du cluster de la destination dans sa table de localisation. Si la destination est localisée, le nœud intermédiaire cherche une entrée qui correspond au cluster auquel appartient la destination parmi les entrées des routes inter-cluster. Il envoie alors le paquet au prochain saut. Quand la destination n'est pas accessible, le paquet de données est supprimé.

Dans L-HCLSR, les routes entre les clusters représentent la distance entre leurs cluster-heads. Le calcul de ces routes dans un nœud non-cluster-head à partir de la topologie des clusters peut entrainer à des routes qui ne sont pas optimales. L'avantage apporté par L-HCLSR est que, tout le long des chemins, les nœuds intermédiaires peuvent court-circuiter les cluster-heads et apporter des corrections locales aux chemins des données. En effet, la connaissance au niveau de chaque nœud des clusters voisins et les plus courts chemins vers ces clusters offre l'opportunité à chaque nœud intermédiaire de sélectionner le meilleur chemin au cluster suivant qui ne passe pas obligatoirement par son cluster-head. Le cluster-head n'est choisi pour acheminer les données que s'il propose le plus court chemin vers le cluster suivant et aucun autre nœud non-cluster-head ne propose un chemin avec le même coût.

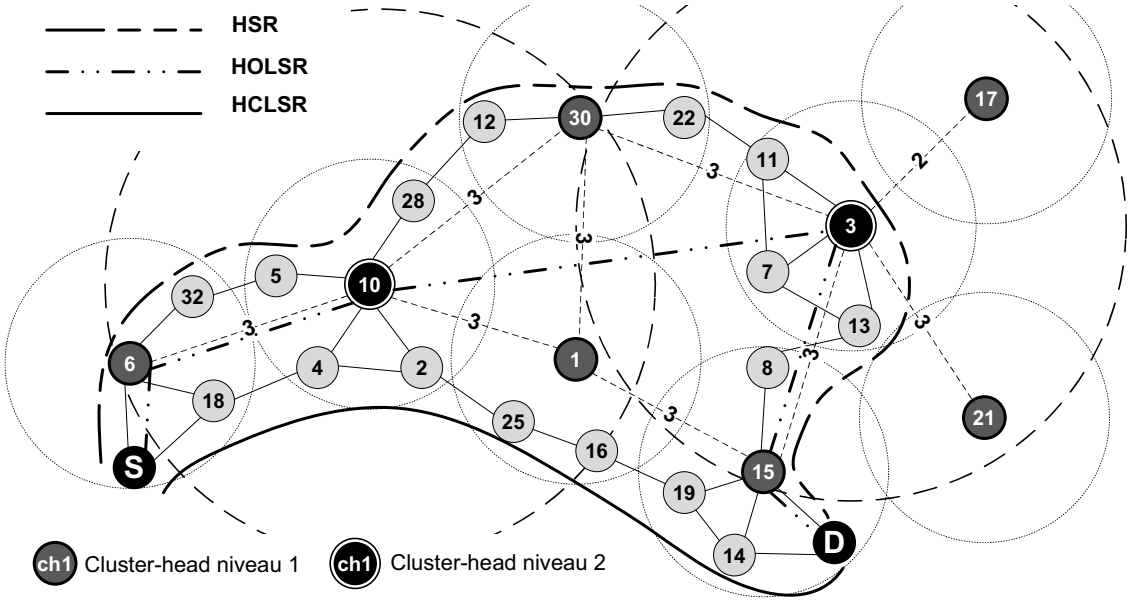


Figure 7.4 – Exemple de routes de données pour les protocoles HSR, HOLSR et L-HCLSR.

La figure 7.4 présente des exemples de chemins de données utilisés par les protocoles HSR, HOLSR et L-HCLSR dans le cas d'une topologie de clusters à deux niveaux. HOLSR suppose que les cluster-heads de même niveau ont une interface de communication de grande portée de transmission. Ainsi, ils peuvent communiquer directement à travers cette interface. Par la suite, le chemin sera composé par une montée dans l'hiérarchie des clus-

ters (la source, son cluster-head de premier niveau, son cluster-head de deuxième niveau) suivie d'une descente de cette hiérarchie (cluster-head de deuxième niveau de la destination, cluster-head de premier niveau de la destination, destination). Les protocoles HSR et L-HCLSR n'adoptent pas l'hypothèse de HOLSR, par la suite, l'acheminement doit se faire saut par saut. Contrairement à L-HCLSR, les chemins des données établies par HSR doivent obligatoirement passé par les cluster-heads de chaque niveau de l'hiérarchie. Ceci implique que les chemins établis par HSR sont plus longs que ceux utilisés par L-HCLSR.

7.4 Analyse théorique

Dans cette section, nous présentons les bornes théoriques supérieures du trafic de contrôle des messages de la gestion de la topologie déjà calculées des protocoles F-OLSR et SA-OLSR. Nous calculons aussi la borne supérieure de ce trafic pour le cas de notre protocole L-HCLSR[Guizani et al., 2012b]. Nous rappelons que ce trafic est représenté par les messages TC pour le cas de F-OLSR, le total des messages TC et TC_Cluster pour SA-OLSR et les messages CTC pour le cas de L-HCLSR. Pour ce faire, nous avons besoin d'introduire certaines hypothèses, d'énoncer quelques notions utiles.

Nous supposons que tous les nœuds ont le même rayon de transmission r. Le signal envoyé par un nœud u est reçu par tous les nœuds à une distance inférieure à r de u. Ainsi, le voisinage du nœud u est formé par tous les nœuds contenus dans un cercle de rayon r. Dans cette circonstance, c'est utile d'utiliser le modèle Unit Disk Graph proposé dans [Clark et al., 1991].

Dans la suite et en s'inspirant de [Adjih et al., 2004], nous considérons que le réseau est uniformément distribué avec une densité λ sur une surface de dimension finie $\mathcal{A}$. Le nombre total de nœuds dans le réseau est $N = \lambda \times \mathcal{A}$. Soit M le nombre moyen de voisins par nœud, $M = \lambda \times S_u$, où S_u est la surface du disque unité concentré à un nœud. Par conséquent, le nombre de nœuds dans le voisinage à d-sauts d'un nœud est égal au nombre de nœuds dans un disque de rayon $d \times r$ qui est en moyenne $d^2 M$. En plus, le rayon du réseau est $R = \sqrt{N/M}$.

Soit M_r le nombre moyen de MPRs sélectionnés par un nœud avec M la dimension du voisinage. Comme le réseau est modelé suivant le modèle Unit Disk Graph , il vient d'après [Jacquet et al., 2002] que $M_r \leq (9\pi^2 M)^{\frac{1}{3}}$.

7.4.1 Borne supérieure du protocole F-OLSR

Dans le protocole Fisheye OLSR [Adjih et al., 2004], la fréquence des TCs relayés par le nœud central depuis ses voisins à D-sauts est $F(D)$. La fonction de fréquence $F(d)$

augmente avec la distance au nœud émetteur. Dans [Adjih et al., 2004], un exemple de fonction de fréquence est proposé : $F(x) = \frac{4x}{3+x}$. Comme présenté dans [Canourgues et al., 2008] et inspiré de [Adjih et al., 2004], la borne supérieure du nombre total de messages TC envoyés et relayés par tous les nœuds du réseau est :

$$T_{OH} \leq N \sum_{D=1}^{\sqrt{N/M}} F(D) \times (D^2 - (D-1)^2) \times \frac{M_r}{TC_{Interval}} \tag{7.1}$$

7.4.2 Borne supérieure du protocole SA-OLSR

Dans cette section, nous présentons la borne supérieure du trafic de contrôle généré par SA-OLSR. Il a été montré dans [Mazieux et al., 2007] que pour l'heuristique Max-min utilisée par SA-OLSR [Canourgues et al., 2008], la borne supérieure du nombre moyen de cluster qu'on peut trouver est :

$$N_{CL} \leq \lambda . \nu(S). \left(1 + \sum_{n=1}^{\infty} \frac{1}{n} \frac{E^n}{n!}\right) exp(-E) \tag{7.2}$$

avec $E = \lambda \pi r^2$ où r est le rayon de propagation d'un nœud et $\nu(S)$ est la mesure Lebesgue de S. Cette borne supérieure est calculée pour un rayon du cluster égal 1.

Le trafic de contrôle généré par la diffusion des messages TC à l'intérieur du rayon du cluster en paquets/s est borné par la borne supérieure suivante :

$$B = (1 + r^2 \times M_R) \times \frac{1}{TC_{Interval}} \tag{7.3}$$

En plus, le trafic de contrôle moyen généré par le relayage du message TC_Cluster envoyé par un cluster-head en paquets/s est :

$$C = (1 + \frac{N}{M} \times M_R) \times \frac{1}{TC\_Cluster_{Interval}} \tag{7.4}$$

Enfin, le trafic de contrôle dû au relayage des messages TC et TC_Cluster est borné par :

$$T_{OH} \leq (B \times N) + (C \times N_{CL}) \tag{7.5}$$

7.4.3 Borne supérieure du protocole L-HCLSR

Les messages CTC de niveau K sont émis avec $TTL = 3^K$. Par conséquent, la distance entre deux cluster-heads voisins du niveau K, est entre $(3^{K-1} + 1)$ and 3^K. En plus, nous avons supposé que les nœuds sont uniformément distribués dans la surface du réseau. Ainsi, en absence d'un grand trou, nous supposons que la distance Euclidienne entre ces deux cluster-heads est supérieure à $(\frac{3^{K-1}+1}{2} \times r)$, où r est le rayon de transmission. Ainsi,

conformément avec le modèle Unit Disk Graph avec un rayon $(\frac{3^{K-1}+1}{2} \times r)$, la borne supérieure du nombre de cluster-heads dans le niveau K est :

$$N_{CL}(K) = \frac{\mathcal{A}}{\pi(\frac{3^{K-1}+1}{4} \times r)^2} \tag{7.6}$$

Soit C l'ensemble des cluster-heads sélectionnés dans le Unit-Disk Graph G, C représente un MIS (Maximal Independent Set). Il est prouvé dans [Alzoubi et al., 2003] que : pour u un cluster-head arbitraire, le nombre de cluster-heads qui sont au plus à trois sauts de u est au plus 47. Par conséquent, le nombre de nœuds de relais au voisinage à 3-sauts est au plus 2×47. En plus, le nombre moyen de nœuds dans le voisinage à 2-sauts est $4 \times M$. Alors, la borne supérieure du nombre de transmission du $CTC(3)$ est $min(2 \times 47, 4 \times M)$. cette valeur correspond aussi au nombre moyen de nœuds de relais dans un cercle de rayon $2 \times r$.

En général, le CTC d'ordre K est transmis au voisinage à 3^K sauts. Le nombre moyen de retransmission dans ce voisinage est égale au nombre moyen de nœuds de relai dans un cercle de diamètre 3^{K-1}. Ainsi, la borne supérieure du nombre moyen de retransmissions de $CTC(K)$ pour $(K < L)$ est :

$$N_{Tr}(K) = min(2 \times 47, 4 \times M) \times \frac{3^{2(K-1)}}{4} \tag{7.7}$$

Les messages CTC d'ordre L sont diffuser dans tous le réseau de rayon $R = \sqrt{N/M}$. Ainsi, la borne supérieure du nombre moyen de retransmissions de $CTC(L)$ est :

$$N_{Tr}(L) = min(2 \times 47, 4 \times M) \times \frac{\frac{N}{M}}{4} \tag{7.8}$$

Soit $F(K) = \frac{1}{2^K}$ la fréquence des messages CTC d'ordre K. La borne supérieure du nombre de messages CTC envoyés et relayés par tous les nœuds dans les L niveaux du réseau est :

$$T_{OH} \leq \sum_{K=1}^{L} F(K) \times N_{Tr}(K) \times \frac{N_{CL}(K)}{CTC_{Interval}} \tag{7.9}$$

7.4.4 Comparaison des bornes supérieures

Dans cette section, nous étudions le trafic de contrôle théorique de L-HCLSR pour différentes valeurs du nombre de niveaux L et nous le comparons avec celui de F-OLSR et SA-OLSR en se basant sur les expressions données dans les sections précédentes.

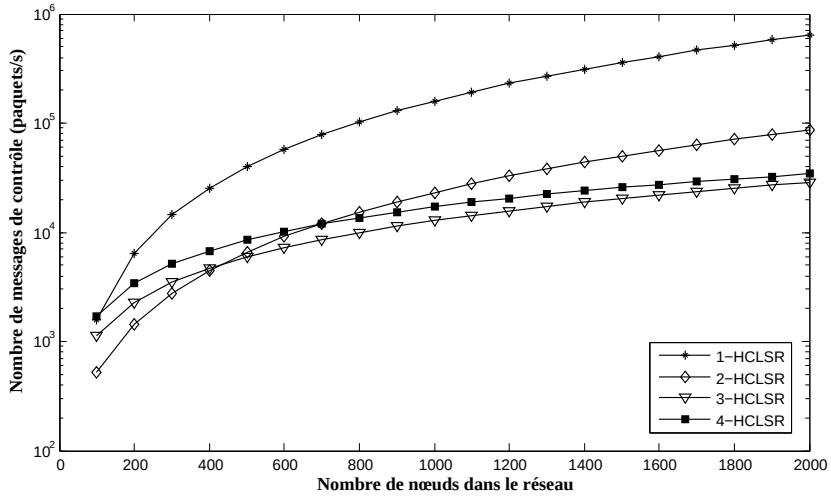


Figure 7.5 – Overhead généré par L-HCLSR, cas des réseaux larges

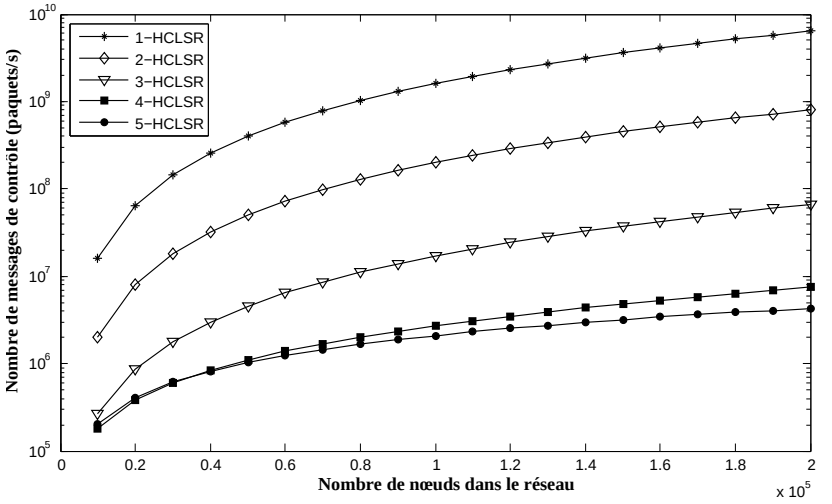


Figure 7.6 – CTC généré par L-HCLSR, cas des réseaux très larges.

Les figures 7.5 et 7.6 illustrent le trafic de contrôle généré par les messages CTC de L-HCLSR en fonction du nombre de nœuds pour une densité de 20 voisins par nœud. Les résultats montrent que le trafic de contrôle de CTC augmente quand le nombre de nœuds augmente. Nous pouvons remarquer aussi que dans un réseau très grand (figure 7.6), ajouter un nouveau niveau à la structure hiérarchique réduit de plus en plus le trafic de contrôle généré par les messages CTC. Mais, cette remarque ne s'applicable plus dans le cas des grands réseaux. Par exemple, dans la figure 7.5, quand le nombre de nœuds est moins que 400, le trafic de contrôle généré par les messages CTC est plus grand pour le HCLSR avec 3 niveaux que celui avec seulement 2 niveaux. En fait, introduire un nouveau niveau est bénéfique quand le rayon du réseau est suffisamment grand par rapport au rayon du niveau maximal courant. Dans le cas de 400 nœuds et de densité 20, le rayon du réseau surpasse 4 ($sqrt(400/20)$). Comme nous savons que le rayon du cluster du deuxième niveau est 4, le

troisième niveau devient bénéfique pour un nombre de nœuds supérieur à 400 nœuds avec la densité 20. De même, le quatrième niveau devient bénéfique au-delà de 3380 nœuds avec la densité 20.

Les figures 7.7 et 7.8 présentent une comparaison entre la borne supérieure théorique de F-OLSR, SA-OLSR et notre protocole proposé L-HCLSR avec deux valeurs du nombre de niveaux (L=2 et L=3).

Les résultats dans la figure 7.7 sont représentés avec la densité de 30 voisins par nœud. Comme illustré dans cette figure, le trafic de contrôle généré par les messages TC de tous les protocoles augmentent lentement quand le nombre de nœuds augmente. Au delà de 3000 nœuds, nous remarquons que F-OLSR donne des résultats meilleurs comparé à 2-HCLSR. En fait, quand le nombre de nœuds augmente, le nombre de clusters de 2-HCLSR augmentent et donc l'*overhead* augmente. Le meilleur trafic de contrôle est donné par 3-HCLSR.

La figure 7.8 montre le trafic de contrôle en fonction de la densité pour un nombre fixe de nœuds égal à 5000. Nous remarquons que F-OLSR ne profite pas de la densité. Pour le reste des protocoles, le trafic de contrôle est réduit considérablement quand la densité augmente. Ce résultat est attendu parce que F-OLSR n'utilise pas la clusterisation comme les autres protocoles comparés. Nous remarquons aussi que L-HCLSR profite plus de la densité que SA-OLSR parce qu'il utilise une structure hiérarchique des clusters.

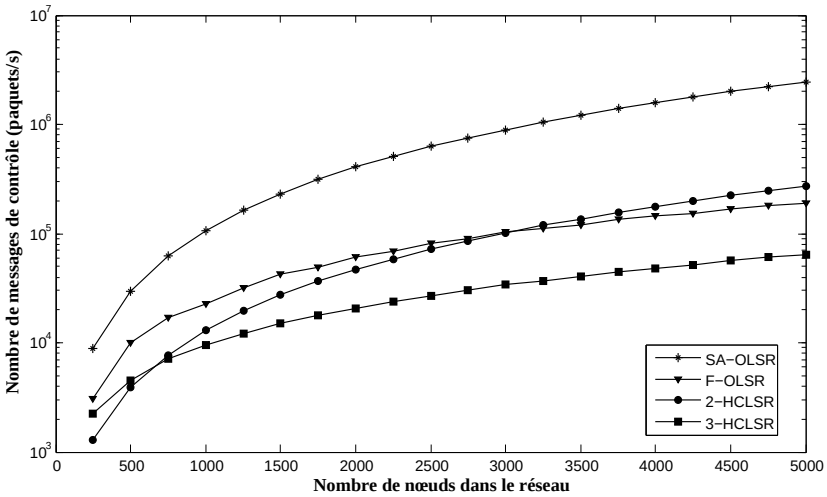


Figure 7.7 – Comparaison du trafic de contrôle généré par TC-CTC de F-OLSR, SA-OLSR et L-HCLSR pour (L=2 et L=3), densité=30

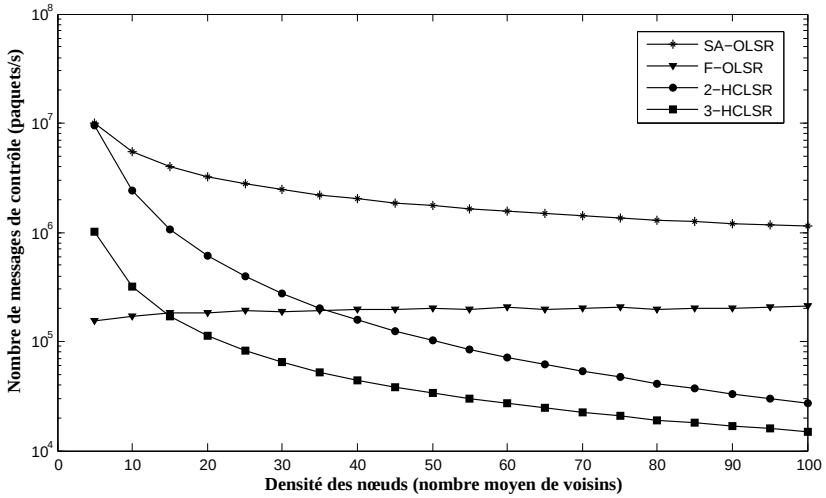


Figure 7.8 – Comparaison du trafic de contrôle généré par TC-CTC de F-OLSR, SA-OLSR et L-HCLSR pour (L=2 et L=3), Nombre de nœuds=5000

7.5 Analyse expérimentale

Dans cette section, nous évaluons par simulation la performance de notre protocole de routage L-HCLSR avec le simulateur du réseau NS-2 [Doe]. Nous étudions les performances de notre protocole L-HCLSR dans le cas d'un seul niveau (L=1) et le cas de deux niveaux (L=2). Nous comparons notre proposition au protocole SA-OLSR [Canourgues et al., 2008]. Le cas d'un seul niveau correspond en réalité avec le protocole CLSR décrit dans le chapitre 6. Pour les simulations présentées dans cette section, nous utilisons les mêmes paramètres concernant la section 6.7.1.Le tableau 7.2 résume l'ensemble de ces paramètres.

Pour évaluer et comparer les protocoles de routage, Nous recourons aux métriques employées dans l'évaluation expérimentale du protocole CLSR (section 6.7.2) et qui sont les suivantes :

- Taux de livraison des paquets.
- Taille de trafic de contrôle (Overhead).
- Le nombre moyen de retransmissions des paquets.
- Délai moyen de bout-en-bout.

Dans ce travail, nous nous sommes intéressés à la scalabilité des protocoles de routage. Pour cette raison, nous varions le nombre de nœuds pour la même valeur de densité.

Dans un premier temps, nous comparons les bornes supérieures théoriques et les valeurs obtenues par simulation pour 1-HCLSR (CLSR) et 2-HCLSR avec les mêmes paramètres. La figure 7.9 montre qu'il existe une différence importante entre les limites supérieures théoriques et les valeurs obtenues par simulation. Cette différence prouve que notre borne supérieure théorique est surestimée.

Paramètres globaux	
Couche liaison	IEEE 802.11
Durée de simulation	400 secondes
Bande passante	2 Mb/s
Portée	250m
Densité des nœuds	10 nœuds
Mobilité	
Modèle de mobilité	RWP
Vitesse des nœuds	[0.1-0.5] m/s
Temps de pause	[0-10] s
Trafic de données	
Type de trafic	CBR
Taille des paquets	512 octets
Nombre de connexions	50
Débit de connexion	4 paquets/secondes

Table 7.2 – Paramètres de la simulation

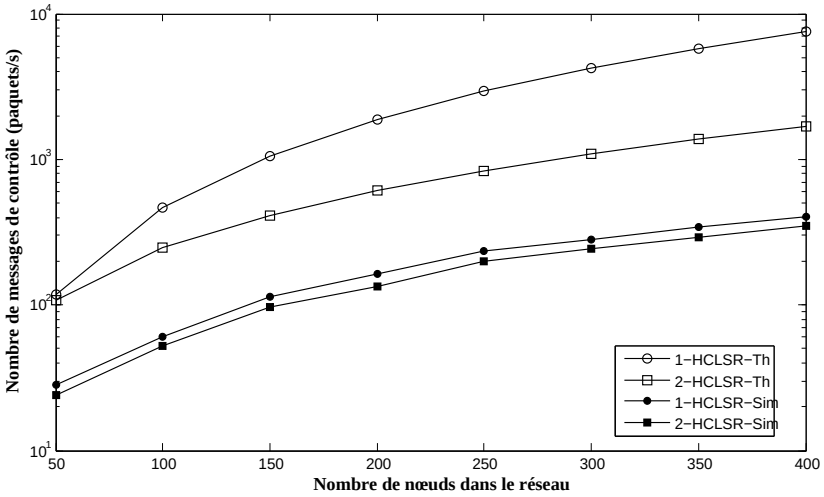


Figure 7.9 – Comparaison des bornes supérieures théoriques et les résultats de simulation du trafic de contrôle du routage en fonction du nombre de nœuds.

La figure 7.10 illustre le trafic de contrôle généré par les protocoles de routage CLSR, 2-HCLSR et SA-OLSR en fonction du nombre de nœuds dans le réseau. Elle montre que l'*overhead* du routage augmente de façon monotone pour les trois protocoles quand le nombre de nœuds augmente. 2-HCLSR produit l'*overhead* le plus réduit. Nous remarquons aussi que l'utilisation de deux niveaux d'hiérarchie dans la structure des clusters permet de réduire l'*overhead* du routage. En effet, les cluster-heads du premier niveau ne diffusent plus leurs messages d'état de liens dans tout le réseau, mais ils les diffusent localement à 3-sauts. Par la suite, le nombre de messages de contrôle diffusés dans tout le réseau est réduit ce qui a un effet direct sur la quantité totale des messages de contrôle générée par le protocole de routage.

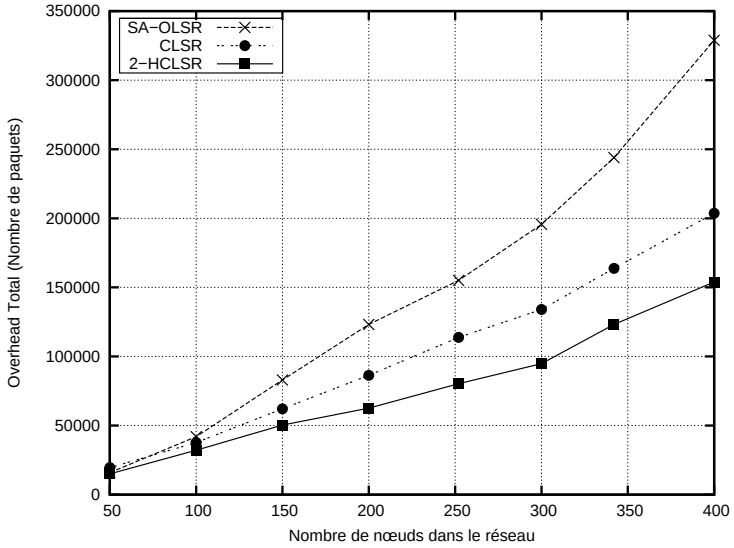


Figure 7.10 – Trafic de contrôle du routage en fonction du nombre de nœuds.

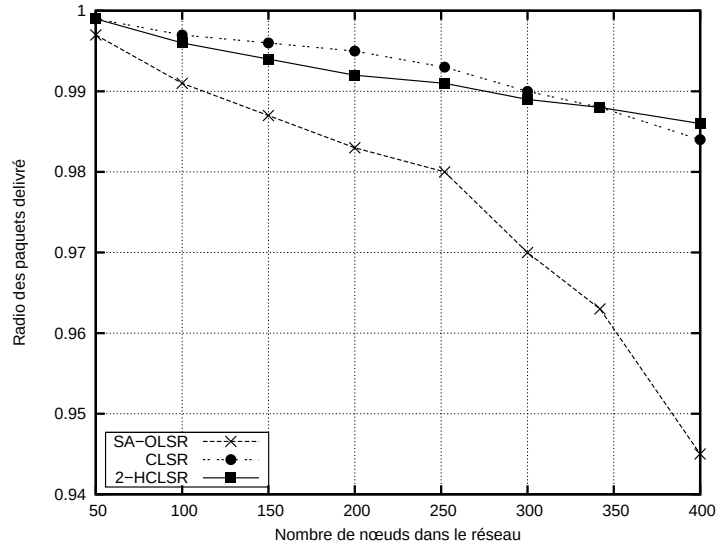


Figure 7.11 – Taux de livraison des paquets en fonction du nombre de nœuds.

La figure 7.11 présente le taux de livraison des paquets de données en fonction du nombre de nœuds dans le réseau. Les performances des trois protocoles diminuent quand le nombre de nœuds augmente. Pour ce cas d'expérimentation, nous remarquons que les trois protocoles donnent un grand taux de livraison des paquets supérieur à 95%. Dans un grand réseau, nous observons que SA-OLSR donne le taux de livraison des paquets le plus réduit. Ce protocole produit le plus important overhead, ce qui augmente la probabilité de collision et de perte de paquets. Ainsi, l'important trafic de contrôle généré par SA-OLSR est la cause majeure de la dégradation de son taux de livraison des paquets. Nous remarquons aussi que les performances des protocoles CLSR et 2-HCLSR en terme du taux de livraison de paquets sont très proches et comparables pour les cas des nombres de nœuds testés. L'introduction d'un deuxième niveau dans l'hiérarchie des clusters permet d'un côté

de réduire l'overhead lié au routage, mais d'un autre côté, il conduit à une vision plus floue de la topologie surtout pour les nœuds les plus éloignés. Lorsque le nombre de noeuds est assez réduit, le gain en overhead du protocole 2-HCLSR n'est pas très aperçu. Dans ce cas, CLSR avec ses routes les plus optimales est légérement meilleur dans la livraison des paquets de données. Par contre, lorsque le nombre de nœuds dépasse les 350 nœuds, l'effet de la différence entre les overheads générés commence à être avisé. Dans ce cas, le protocole 2-HCLSR commence à offrir le meilleur taux de livraison des paquets.

La figure 7.12 illustre le nombre moyen de retransmissions des paquets de données enregistré par les trois protocoles. Nous remarquons que le protocole SA-OLSR produit le plus grand nombre moyen de retransmissions des paquets de données. En effet, le protocole SA-OLSR calcule les chemins aux clusters destinations en considérant la première copie reçue des messages de mise à jour de la topologie. Ces chemins sont supposés être les plus rapides et les moins congestionnés, mais ils ne sont pas les plus courts en terme de distance. Ceci explique le nombre de retransmissions le plus élevé produit par SA-OLSR par rapport à celui fourni par CLSR et 2-HCLSR.

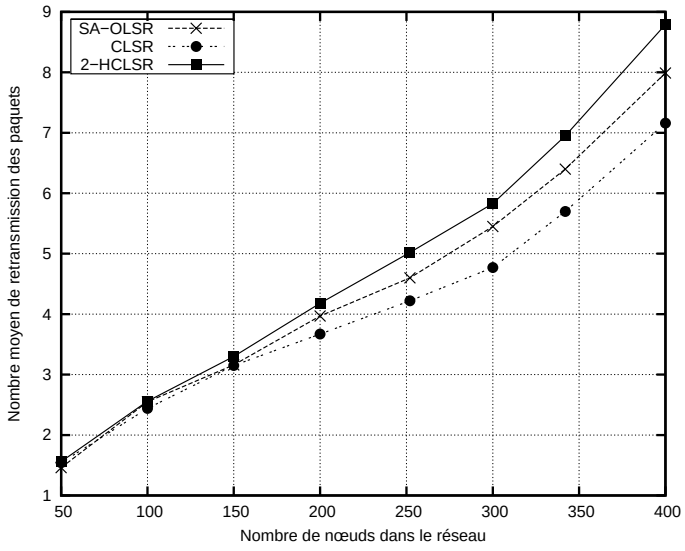


Figure 7.12 – Nombre moyen de retransmissions des paquets en fonction du nombre de nœuds.

Dans L-HCLSR, tous les nœuds maintiennent la topologie globale du réseau formée par l'ensemble des clusters. Chaque nœud est alors capable de calculer les routes d'une façon optimale en terme de nombre de sauts. Cette propriété permet de réduire le nombre de retransmissions des paquets, et ainsi de mieux optimiser l'utilisation de la bande passante. Cependant, l'ajout d'un nouveau niveau dans l'hiérarchie des clusters rend la topologie des nœuds éloignés assez vague. En plus, il réduit la fréquence des messages de mise à jour de la topologie. Ces raisons entraînent des chemins plus longs à chaque fois que le nombre de niveau dans l'hiérarchie augmente . Ceci explique le léger avantage de CLSR par rapport à 2-HCLSR en terme de nombre moyen de retransmissions des paquets de données.

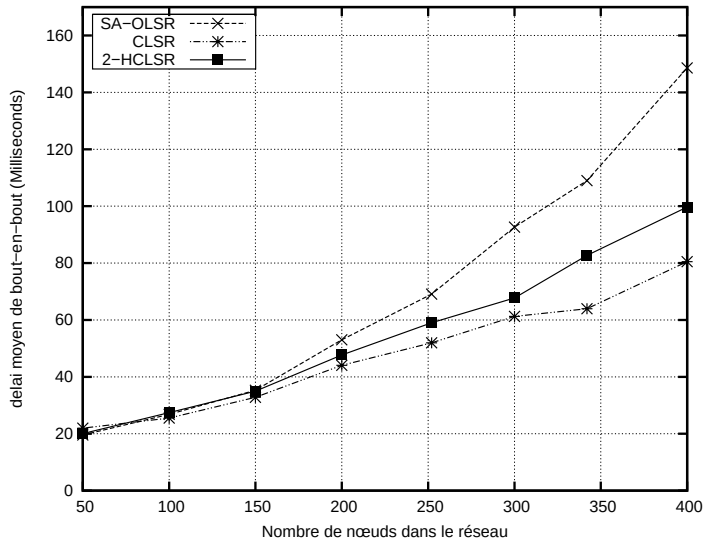


Figure 7.13 – Délai de bout en bout en fonction du nombre de nœuds.

La figure 7.13 présente le délai de bout en bout en fonction du nombre de nœuds. Comme illustré dans cette figure, quand le nombre de nœuds augmente, le délai de bout en bout des trois protocoles augmente. En effet, pour une même densité du réseau, quand le nombre de nœuds augmente, le chemin moyen entre la source et la destination augmente. Ainsi, le nombre de nœuds intermédiaires augmente, ce qui augmente le délai de bout-en-bout. La figure 7.13 illustre que le protocole CLSR donne la meilleure performance en terme de délai de bout en bout grâce à ses chemins les plus optimaux en terme de distance. Le protocole SA-OLSR utilise des chemins supposés être plus rapides malgré qu'ils ne sont pas les plus courts. En réalité, ces chemins étaient plus rapides lors de la diffusion des messages de la topologie. Leur adoption dans les nouveaux chemins de données augmente le trafic de données véhiculé. La non optimalité en terme de distance des chemins utilisés et l'augmentation du trafic de données circulé augmentent considérablement le délai moyen de bout-en-bout enregistré pour le protocole SA-OLSR.

7.6 Conclusion

Ce chapitre présente un protocole de routage à état de liens L-HCLSR pour les grands réseaux SONs qui est basé sur la clusterisation hiérarchique. Le protocole utilise la clusterisation hiérarchique et la technique Fisheye pour réduire le trafic de contrôle du protocole de routage. L'algorithme de clusterisation utilise seulement les informations de contrôle du routage échangées comme information pour former la structure des clusters ce qui réduit le trafic de contrôle total. Uniquement deux types de messages de contrôle sont employés : HELLO et CTC. Le message HELLO est envoyé par chaque nœud et n'est pas retransmis. Ce message est utilisé pour créer la structure logique du premier niveau. Les messages CTC sont générés par les cluster-heads et sont retransmis par un nombre limité de nœuds en

fonction de leur distance du cluster-head. Les messages CTC permettent aux cluster-heads de découvrir leurs cluster-heads voisins du même niveau et de choisir les cluster-heads du prochain niveau. Ce message est aussi utilisé par tous les nœuds du réseau pour construire la topologie globale du réseau et calculer la table de routage. Les analyses théoriques du trafic de contrôle de notre protocole montre que, dans un réseau très grand, ajouter un nouveau niveau à la structure hiérarchique réduit de plus en plus le trafic de contrôle des messages CTC générés. Mais, dans le cas de grands réseaux, le niveau maximal qui donne le trafic de contrôle le plus réduit dépend du nombre de nœuds. Les analyses théoriques du trafic de contrôle de notre protocole comparé au F-OLSR et SA-OLSR montrent que L-HCLSR tire plus de profits de la densité et donne un résultat comparable ou meilleur dans beaucoup de cas. Dans l'étude par simulation, nous varions le nombre de nœuds entre 50 et 400 pour une valeur fixe de densité des nœuds égale à 20 nœuds. Les résultats montrent que L-HCLSR donnent le trafic de contrôle le plus réduit et donne de meilleures performances que SA-OLSR en termes de taux de livraison des paquets et de délai de bout en bout.

CHAPITRE 8

CONCLUSION

8.1 Bilan et apports

Au travers de cette thèse, notre objectif a été d'aborder le problème de scalabilité du protocole de routage dans les réseaux sans fil mobiles afin de proposer une solution pour améliorer les performances du réseau. L'étude approfondie des précédentes solutions a guidé nos travaux vers le concept de clusterisation pour le mettre à profit de notre objectif . Dans cette optique, nous avons porté un grand intérêt à l'étude de la technique de clusterisation pour proposer un algorithme adapté au problème de passage à l'échelle de la fonction de routage. Nous avons conçu, dans un premier temps, un algorithme de clusterisation à un saut dont l'objectif est d'améliorer la stabilité de la topologie des clusters. Ensuite, nous avons envisagé de généraliser cet algorithme à K-sauts et mettre en évidence l'effet du choix des différents paramètres qui régissent le fonctionnement de l'algorithme. La maîtrise de la technique de clusterisation et l'étude approfondie des protocoles de routage existants nous ont permis alors de proposer un nouveau protocole de routage scalable. Ce protocole à état de liens basé sur une structure de clusters a comme ultime objectif de résoudre le problème de passage à l'échelle. Enfin, en introduisant une hiérarchie dans la structure de clusters formée, nous avons proposé une version hiérarchique du protocole de routage afin de le rendre de plus en plus scalable.

8.1.1 Proposition d'un algorithme de clusterisation à structure stable

Nous avons proposé, dans cette thèse, un algorithme de clusterisation appelé α-SSCA (α-Stability Structure Clustering Algorithm). L'objectif de cet algorithme est de construire et maintenir une structure stable des clusters afin qu'elle soit exploitée par la suite dans un protocole de routage à état de liens. α-SSCA est un algorithme de clusterisation à un saut qui utilise la connectivité des nœuds comme métrique de sélection des cluster-heads. Avec ce choix de métrique, les informations de connectivité échangées dans un protocole de routage sont aussi exploitées dans l'algorithme de clusterisation. Par conséquent, les informations explicites de clusterisation sont réduites considérablement. En plus, l'approche de maintenance adoptée par notre proposition autorise la présence temporaire de deux cluster-heads dans la même portée de transmission sous certaines circonstances. Ces circonstances

relaxent les conditions pour lesquelles un cluster-head abandonne son rôle ce qui permet de réduire la fréquence des changements dans la topologie des clusters. Nous avons évalué les performances de notre algorithme par simulation sous le simulateur des réseaux NS-2. Dans une première évaluation, nous avons comparé notre algorithme avec les algorithmes HCC, LCC-hc. Les résultats des simulations montrent que notre algorithme a de meilleures performances comparé à HCC et LCC en termes de nombre de changements de cluster-head et de durée moyenne de vie des cluster-heads. Dans une deuxième évaluation, nous avons étudié l'impact de la stabilité des clusters sur un protocole de routage basique à état de liens. Les résultats des simulations montrent que le protocole de routage avec l'utilisation de α-SSCA donne de meilleures performances en termes de ratio de paquets délivrés, de taille d'overhead généré et de délai de bout-en-bout comparé aux protocoles utilisant HCC ou LCC-hc.

8.1.2 Proposition d'un algorithme de clusterisation générique à K-sauts

Sur la base du l'algorithme α-SSCA, nous avons proposé un algorithme générique à K-sauts pour couvrir des réseaux plus larges. En effet, avec un diamètre de cluster égal à un seul saut, l'algorithme de clusterisation génère un nombre important de clusters. Ce nombre important de clusters peut nuire à certaines utilisations de l'algorithme tel que le cas du routage puisqu'il induit à l'augmentation du trafic de contrôle. Cet algorithme, nommé SKCA (Stability K-hops Clustering Algorithm) combine les deux algorithmes α-SSCA et 3HBAC à K-sauts. La contribution principale de SKCA est l'amélioration de la stabilité de la topologie des clusters tout en gardant assez réduit le nombre de clusters dans le réseau. Dans la phase de formation des clusters, SKCA reproduit l'idée de 3HBAC en introduisant le statut de membre-invité qui est un nœud qui n'est pas à K-sauts d'aucun cluster-head mais il est à une distance de ϵ-sauts à un nœud membre d'un autre cluster. Comme dans α-SSCA, un facteur de stabilité α est utilisé pour diminuer le renoncement des cluster-heads à leur rôle dans la phase de maintenance des clusters. SKCA est aussi indépendant de la métrique utilisée pour la sélection des cluster-heads. Pour réduire le coût de diffusion de la métrique dans le voisinage à K-sauts, SKCA prévoit deux tours d'élection des cluster-heads. Dans le premier tour, tous les nœuds participent à l'élection mais dans un voisinage réduit à r-sauts. Seuls les nœuds vainqueurs du premier tour participent au second tour d'élection qui se déroule dans tout le voisinage à K-sauts. L'analyse expérimentale de SKCA est faite à travers des simulations sous NS-2. L'objectif de cette analyse était d'étudier le comportement de cet algorithme en variant ses différents paramètres. Nous citons, dans ce cadre, l'impact du rayon de pré-sélection r. Quand ce paramètre augmente, il améliore la stabilité des clusters mais en contre partie il engendre un trafic croissant d'*overhead*.

8.1.3 Proposition d'un protocole de routage à état de liens des clusters

La clusterisation ne représente pas une fin en soi. Nous avons adopté cette technique afin d'améliorer les performances du protocole de routage à large échelle. Cette technique peut être déployée dans d'autres domaines comme la sécurité, la conservation d'énergie, etc. Le protocole de routage que nous avons proposé, appelé CLSR (Cluster-based Link State Routing), applique l'approche état de liens en intra-clusters et également en inter-clusters. CLSR utilise seulement deux types de paquets. Le paquet HELLO est envoyé à un seul saut et sert à la structuration du réseau en un ensemble de clusters et au calcul des routes à deux sauts. Le deuxième type de paquets est CTC qui est diffusé dans tout le réseau par les cluster-heads. Il est utilisé par chaque nœud pour calculer les routes inter-cluster. La diffusion des paquets CTC est optimisée à travers une dorsale virtuelle formée par un sous ensemble dominant connecté composé par les cluster-heads et des nœuds choisis comme passerelles.

L'apport principal de CLSR est l'utilisation de l'algorithme de clusterisation α-SSCA. Cet algorithme n'introduit pas de trafic de contrôle supplémentaire pour la clusterisation puisqu'il se contente des paquets de routage HELLO. Cette propriété réduit considérablement le trafic de contrôle total du protocole de routage. En outre, l'algorithme α-SSCA permet de créer une structure plus stable ce qui augmente la validité des routes calculées à partir de la topologie des clusters. D'autre part, le choix d'un algorithme de clusterisation à un seul saut permet d'éliminer les messages de routage à l'intérieur du cluster. En effet, toutes les informations à deux sauts sont contenues dans les paquets HELLO. En plus, avec cette distance les cluster-heads peuvent déterminer leurs cluster-heads voisins sans avoir recours à des messages supplémentaires. Pour évaluer les performances de notre proposition, nous l'avons implémentée sous le simulateur NS-2 et nous avons comparé ses performances à ceux des protocoles F-OLSR, OLSR-Trees et SA-OLSR. Les résultats montrent que CLSR offre de meilleures performances en termes de livraison de paquets, de taille du trafic de contrôle, du nombre moyen de retransmissions des paquets et du délai moyen de bout en bout.

8.1.4 Proposition d'un protocole de routage hiérarchique

La scalabilité du protocole CLSR peut être améliorer par l'introduction de la notion de hiérarchie dans la structure des clusters. En effet, comme CLSR s'appuie sur un algorithme à un saut, le nombre de clusters générés devient assez important quand le réseau devient assez large. Il sera intéressant de réduire le nombre de clusters par groupement des clusters proches dans un même cluster. Ce principe est adopté par le protocole proposé et nommé L-HCLSR (L-Hierarchical Cluster-based Link State Routing). Dans cette proposition, les nœuds sont organisés dans L niveaux logiques de clusters. Le premier niveau est structuré de la même façon proposé par le protocole CLSR. Les cluster-heads élus dans ce premier

niveau participent comme étant des nœuds ordinaires dans le second niveau et parmi eux seront sélectionnés des cluster-heads de deuxième niveau. Suivant le même principe, les cluster-heads de niveau K participent à l'élection des cluster-heads de niveau K + 1 jusqu'à construire le dernier niveau d'ordre L. Comme CLSR, L-HCLSR utilise deux types de messages HELLO et CTC. Le message HELLO garde le même rôle que celui dans CLSR. Les messages CTC sont générés par les cluster-heads et sont retransmis par un nombre limité de nœuds en fonction de leur distance du cluster-head. Les messages CTC permettent aux cluster-heads de découvrir leurs cluster-heads voisins du même niveau et de choisir les cluster-heads du prochain niveau. Ce message est aussi utilisé par tous les nœuds dans le réseau afin de construire la topologie globale du réseau et de calculer leur table de routage. Pour réduire le nombre des messages CTC, le protocole L-HCLSR intègre aussi la technique Fisheye. Par la suite, les destinations les plus proches reçoivent plus fréquemment les messages CTC que celles les plus éloignées. L'évaluation des performances de L-HCLSR est faite par analyse théorique et par simulation sous le simulateur NS-2. L'analyse théorique a seulement étudié le trafic de contrôle et elle a montré que le niveau de hiérarchie maximal de L-HCLSR dépend du nombre de nœuds dans un réseau très large. Les analyses théoriques du trafic de contrôle de notre protocole comparé au Fisheye-OLSR et SA-OLSR montrent que L-HCLSR tire plus de profits de la densité et donne un résultat comparable ou meilleur dans beaucoup de cas. Dans l'étude par simulation, les résultats montrent que L-HCLSR donnent le trafic de contrôle le plus réduit et donne de meilleures performances que SA-OLSR en termes de taux de livraison des paquets et de délai de bout en bout.

8.2 Perspectives

Le travail réalisé au long de cette thèse permet de soulever des questions qui pourraient faire l'objet de nouveaux axes de recherche ou d'extension.

- L'étude des deux algorithmes de clusterisation proposés, α-SSCA et SKCA, a été faite par simulation sous le simulateur des réseaux NS-2. Cette étude peut être complétée par une analyse théorique après avoir expliciter les formules nécessaires.

- Dans le cadre des deux protocoles de routage proposés CLSR et L-HCLSR, le message des informations topologiques CTC, diffusé sur tout le réseau, contient des informations nécessaires pour le calcul des routes. En particulier, ce message contient la liste des membres rattachés au cluster-head source du message. Cette liste est utilisée pour la localisation d'un nœud dans son cluster. Quand le nombre de nœuds du cluster croît, ce message augmente de taille. Ceci augmente par la suite le trafic de contrôle et peut dégrader les performances du protocole de routage. Il est donc intéressent de trouver une solution qui réduit la taille des messages CTC en suppri-

mant la liste des membres du cluster. Ainsi, la fonction de localisation sera déléguée à d'autres entités du réseau. Nous pouvons citer deux alternatives possibles pour résoudre le problème de localisation.

- ⊙ Via une solution distribuée où chaque nœud est tenu d'informer un cluster-head particulier du réseau de son cluster d'attache à chaque changement. Le choix de ce cluster-head peut se faire à l'aide d'une fonction de hachage. Cette même fonction est utilisée par tout nœud désirant connaître le cluster-head d'un autre nœud. Il suffit alors qu'il applique cette fonction au nœud destination, le résultat de cette fonction donne le cluster-head qui détient l'information de localisation du nœud destination. Il reste alors de le consulter.
- ⊙ Via la sélection de nœuds particuliers dans le réseau qui garderont l'information de localisation. Ces nœuds dispersés dans le réseau, répondront aux requêtes de tout nœud désirant connaître le cluster d'un autre nœud. Ce mécanisme rappelle le système du serveur DNS de l'Internet.

■ Pour améliorer la scalabilité du protocole CLSR, nous avons adopté dans la thèse la démarche hiérarchique où nous avons regroupé les clusters proches dans un même cluster de niveau supérieur. Nous avons ainsi proposé le protocole L-HCLSR et ses résultats obtenus sont motivants. Néanmoins, une deuxième démarche peut être suivie pour rendre CLSR plus scalable. Cette démarche consiste à intégrer un algorithme qui augmente le rayon des clusters. En fait, CLSR, tel qu'il a été conçu et évalué, s'appuie sur un algorithme de clusterisation à un seul saut qui est l'algorithme α-SSCA. L'utilisation d'un seul saut comme rayon des clusters peut générer un nombre très important de clusters. Il sera alors intéressant de voir l'impact d'utiliser un algorithme à K-sauts, comme l'algorithme SKCA, sur les performances de CLSR. Une étude comparative des résultats de cette démarche avec les résultats obtenus avec la démarche hiérarchique est aussi envisageable.

■ Étendre le protocole de routage pour permettre la diffusion de groupes (*Multicast*) représente une perspective intéressante de nos travaux. cette extension peut se baser sur le concept des arbres enracinés au noyaux (CBT, Core Based Tree) pour tirer profit de nos algorithmes de clusterisation. En sélectionnant plusieurs noyaux à K-sauts, trois solutions utilisant trois structures différentes peuvent être envisagées.

- ⊙ Structure en *mesh* : Sans construction préalable, chaque source et chaque nœud intermédiaire sélectionnent les noyaux potentiels du groupe dès qu'ils possèdent des données à envoyer. Ils diffusent alors les paquets vers tous les membres et tous les noyaux qui sont dans leur voisinage à K-sauts.
- ⊙ Arbre par source : Chaque source sélectionne plusieurs noyaux qui couvrent à K-sauts tous les membres du groupe. Elle envoie un message qui contient l'ensemble des noyaux sélectionnés. Ce message sera transmis d'un noyau à un autre pour

construire l'arbre qui sera utilisé dans la diffusion de groupe des données.

⊙ Arbre partagé : Tous les membres d'un groupe participent pour élire plusieurs noyaux qui forment un sous-ensemble dominant du groupe à K-sauts. Chaque nœud et chaque noyau se connectent alors au noyau le plus proche.

TROISIÈME PARTIE

Bibliographie

Bibliographie

Ameer-Ahmed Abbasi and Mohamed F. Younis. A Survey on Clustering Algorithms for Wireless Sensor Networks. *Computer Communications*, 30(14-15) :2826–2841, 2007. URL http://dx.doi.org/10.1016/j.comcom.2007.05.024.

Cedric Adjih, Emmanuel Baccelli, Thomas Heide Clausen, Philippe Jacquet, and Georgios Rodolakis. Fish Eye OLSR Scaling Properties. *IEEE Journal of Communications and Networks (JCN), Special Issue on Mobile Ad Hoc Wireless Networks*, 2004.

Ratish Agerwal and Mr Motwani. Survey of Clustering Algorithms for MANET. *International Journal on Computer Science and Engineering*, 1(2) :98–104, 2009.

Khaled M. Alzoubi, Peng Jun Wan, and Ophir Frieder. New Distributed Algorithm for Connected Dominating Set in Wireless Ad Hoc Networks. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS'02)*, pages 3849–3855, Hawaii, USA, January 2002. IEEE. ISBN 0-7695-1435-9. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=994519.

Khaled M. Alzoubi, Peng Jun Wan, and Ophir Frieder. Maximal Independent Set, Weakly-Connected Dominating Set, and Induced Spanners in Wireless Ad Hoc Networks. *International Journal of Foundations of Computer Science IJFCS*, 14(2) :287–303, 2003.

Alan Amis and Ravi Prakash. Load-Balancing Clusters in Wireless Ad Hoc Networks. In *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'00)*, pages 25–32, Richardson, Texas, USA, 24-25 March 2000. IEEE. ISBN 0-7695-0559-7. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=888028.

Alan Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM'00)*, volume 1, pages 32–41, Los Alamitos, Californie, USA, 26–30 March 2000. IEEE. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=832171.

E. Baccelli. OLSR Scaling with Hierarchical Routing and Dynamic Tree Clustering. *IASTED International Conference on Networks and Communication Systems (NCS), Chiang Mai, Thailand*, 2006.

Stefano Basagni. Distributed Clustering for Ad Hoc Networks. In *Proceedings of the 4th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'99)*, pages 310–315, Fremantle, Australia, June 1999. IEEE Computer Society. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=778957.

P. Basu, N. Khan, and T.D.C. Little. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. In *Proceedings of the International Conference on Distributed Computing Systems Workshop (ICDCSW'01)*, pages 413–418, Mesa, Arizona, USA, April 2001. IEEE. ISBN 0-7695-1080-9. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=918738.

Elizabeth M. Belding-Royer. Hierarchical Routing in Ad Hoc Mobile Networks. *Wireless Communications and Mobile Computing*, 2(5) :515–32, 2002.

Azzedine Boukerche. Performance Evaluation of Routing Protocols for Ad Hoc Wireless Networks. *Mobile Networks and Applications*, 9 :333–342, 2004. ISSN 1383-469X. URL http://dx.doi.org/10.1023/B:MONE.0000031592.23792.1c. 10.1023/B :MONE.0000031592.23792.1c.

Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, pages 85–97, New York, USA, October 25-30 1998. ACM. ISBN 1-58113-035-X. URL http://dl.acm.org/citation.cfm?id=288256.

Lucile Canourgues, Jérôme Lephay, Laurent Soyer, and André-Luc Beylot. Scalable adaptation of the olsr protocol for large clustered mobile ad hoc networks. In *Proceedings of the 2008 IFIP Annual Mediterranean Ad Hoc Networking Conference, Advances in Ad Hoc Networking (MED-HOC-NET'08)*, volume 265, pages 97–108, Palma, Majorca, Spain, June 23-27 2008. Springer.

I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet-draft, IETF MANET Working Group, July 26 2010. URL http://tools.ietf.org/html/draft-ietf-manet-dymo-21. Expiration : January 27, 2011.

Charles E.Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*, volume 24, pages 234–244. ACM, October 1994.

Mainak Chatterjee, Sajal K. Das, and Damla Turgut. An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'00)*, volume 3, pages 1697–1701, January 09 2000.

Mainak Chatterjee, Sajal K. Das, and Damla Turgut. WCA : A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Journal of Cluster Computing, Special Issue on Mobile Ad hoc Networks*, 5(2) :193–204, 2002. URL http://dx.doi.org/10.1023/A:1013941929408.

Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span : An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *Wireless Networks*, 8 :481–494, September 2002a. ISSN 1022-0038. URL http://dx.doi.org/10.1023/A:1016542229220.

Geng Chen, F.G. Nocetti, J.S. Gonzalez, and I. Stojmenovic. Connectivity Based k-Hop Clustering in Wireless Networks. In *Proceedings of the 5th Hawaii International Conference on System Sciences (HICSS'02)*, pages 2450–2459, Hawaii, USA, January 7-10 2002b. IEEE. ISBN 0-7695-1435-9. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=994183.

Tsu Wei Chen and Mario Gerla. Global State Routing : A New Routing Scheme for Ad-Hoc Wireless Networks. In *Proceedings of the 1998 IEEE International Conference on Communications (ICC'98)*, volume 1, pages 171–175, Atlanta, Georgia, USA, June 7-11 1998. IEEE. ISBN 0-7803-4788-9. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=682615.

Wenli Chen, Nitin Jain, and S. Singh. ANMP : Ad Hoc Network Management Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8) :1506–1531, August 1999. ISSN 0733-8716.

Yuanzhu Peter Chen and Arthur L. Liestman. Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 165–172, Lausanne, Switzerland, June 9-11 2002. ACM. URL http://dl.acm.org/citation.cfm?id=513800.513821&coll=DL&dl=GUIDE&CFID=67069589&CFTOKEN=29010825.

Ching Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing In Clustered Multihop, Mobile Wireless Networks With Fading Channel. In *Proceedings of the IEEE Singapore International Conference on Networks (SICON'97)*, pages 197–212, Singapore, August 19 1997. IEEE. URL http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8359.

Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit Disk Graphs. *Discrete Mathematics*, 86 :165–177, January 1991. ISSN 0012-365X. URL http://dx.doi.org/10.1016/0012-365X(90)90358-O.

Thomas Heide Clausen. Combining Temporal and Spatial Partial Topology for MANET routing - Merging OLSR and FSR. In *Proceedings of the IEEE conference on Wireless Personal Multimedia Communications (WPMC'03)*, Yokosuka, Japan, October 2003. IEEE. URL http://www.thomasclausen.org/Thomas_Heide_Clausens_Website/Conferences_files/paper_18.pdf.

Thomas Heide Clausen and Philippe Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003. URL http://www.ietf.org/rfc/rfc3626.txt.

M. Scott Corson and Anthony Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks . *Wireless Networks*, 1 :61–81, February 1995. ISSN 1022-0038. URL http://dx.doi.org/10.1007/BF01196259.

Samir R. Das, Robert Castañeda, and Jiangtao Yan. Simulation-based performance evaluation of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 5 :179–189, September 2000. ISSN 1383-469X. URL http://dx.doi.org/10.1023/A:1019108612308.

Dan J. Dechene, A. El Jardali, M. Luccini, and A. Sauer. A Survey of Clustering Algorithms for Wireless Sensor Networks. Technical report, The University Of Western Ontario - Department of Electrical and Computer Engineering, December 2006.

E. W. Dijkstra. A Note on Two Problems in Connexion With Graphs. *Numerische Mathematik*, 1 :269–271, 1959. ISSN 0029-599X. URL http://dx.doi.org/10.1007/BF01386390.

Ringo Doe. The Network Simulator - NS-2. June .

Chyi-Ren DOW, Jyh-Horng Lin, Shiow-Fen Hwang, and Yi-Wen Wang. An Efficient Distributed Clustering Scheme for Ad-hoc Wireless Networks. *IEICE Transactions Communications*, 85(8) :1561–1571, 2002. ISSN 0916-8516.

Jean E. Dunbar, Jerrold W. Grossman, Johannes H. Hattingh, Stephen T. Hedetniemi, and Alice A. McRae. On Weakly-connected Domination in Graphs. *Discrete Mathematics*, 167-168 :261–269, 1997. URL http://dx.doi.org/10.1016/S0012-365X(96)00233-6.

Anthony Ephremides, Jeffrey E. Wieselthier, and D.J. Baker. A Design Concept for Reliable Mobile Radio Networks With Frequency Hopping Signaling. *Proceedings of the IEEE*, 75(1) :56 – 73, January 1987. ISSN 0018-9219.

Inn Inn ER and Winston K.G. Seah. Mobility-based d-Hop Clustering Algorithm for Mobile Ad Hoc Networks. In *Proceedings of the 2004 IEEE Wireless Communications and Networking Conference (WCNC'04)*, volume 4, pages 2359–2364, Atlanta, Georgia, USA, March 21-25 2004. IEEE.

Laura Marie Feeney. A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks. Technical Report T99/07, Swedish Institute of Computer Science, October 1999.

Yaacov Fernandess and Dahlia Malkhi. K-Clustering in Wireless Ad Hoc Networks. In *Proceedings of the 2nd ACM international workshop on Principles of mobile computing (POMC'02)*, pages 31–37, Toulouse, France, 2002. ACM. ISBN 1-58113-511-4. URL http://doi.acm.org/10.1145/584490.584497.

Jose Joaquin Garcia-Luna-Aceves and Marcelo Spohn. Source-Tree Routing in Wireless Networks. In *Proceedings of the 7th IEEE International Conference on Network Protocols (ICNP'99)*, pages 273–282, Toronto, Canada, October 31 - November 3 1999. IEEE Computer Society. URL http://computer.org/proceedings/icnp/0412/04120273abs.htm.

Ying Ge, Louise Lamont, and Luis Villasenor. Hierarchical OLSR - A Scalable Proactive Routing Protocol for Heterogeneous Ad Hoc Networks. In *Proceedings of the IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'05)*, volume 3, pages 17–23, Montreal, Canada, August 22-24 2005. IEEE. ISBN 0-7803-9181-0. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1512880.

Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, Mobile, Multimedia Radio Network. *Wireless Networks*, 1(3) :255–265, 1995. URL http://dx.doi.org/10.1007/BF01200845.

Sudipto Guha and Samir Khuller. Approximation Algorithms for Connected Dominating Sets. *Algorithmica*, 20(4) :374–387, 1998. URL http://dx.doi.org/10.1007/PL00009201.

Badreddine Guizani, Béchir Ayeb, and Abderrafiâa Koukam. Impact of Stability in Cluster Based Link State Routing Protocol for Self-Organizing Networks. *Accepted in 7th IEEE International Wireless Communications and Mobile Computing Conference (ICWMC 2011)*, July 5-8 2011a.

Badreddine Guizani, Béchir el Ayeb, and Abderrafiâa Koukam. Hierarchical Cluster-based Link State Routing Protocol for Large Self-organizing Networks. In *Proceedings of the 12th IEEE International Conference on High Performance Switching and Routing (HPSR 2011)*, pages 203–208, Cartagena, Spain, July 4-7 2011b. IEEE Communications Society. ISBN 978-1-4244-8454-6. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5986027.

Badreddine Guizani, Béchir Ayeb, and Abderrafiâa Koukam. A New Cluster-Based Link State Routing for Mobile Ad Hoc Networks. In *Proceedings of the second International Conference on Communications and Information Technology (ICCIT-2012)*, Hammamet, Tunisia, June 26-2 2012a. IEEE.

Badreddine Guizani, Béchir Ayeb, and Abderrafiâa Koukam. A Scalable Hierarchical Routing for Mobile Ad Hoc Networks. *Submitted to the Ad Hoc Networks journal*, 2012b. ISSN 1570-8705.

SONG GUO and OLIVER W. YANG. Performance of Backup Source Routing (BSR) in Mobile Ad Hoc Networks. In *Proceedings of the 2002 IEEE Wireless Communications and Networking Conference (WCNC'02)*, volume 1, pages 440–444, Orlando, Florida, USA,, March 17-21 2002. IEEE Communications Society. ISBN 0-7803-7376-6. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=993535.

P. Gupta and P.R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46 (2) :388–404, 2000. ISSN 0018-9448.

Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet-draft, IETF MANET Working Group, July 2002a. URL http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt. Expiration : January, 2003.

Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Bordercast Resolution Protocol (BRP) for Ad Hoc Networks. Internet-draft, IETF MANET Working Group, July 2002b. URL http://tools.ietf.org/html/draft-ietf-manet-zone-brp-02. Expiration : January 2003.

Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Interzone Routing Protocol (IERP) for Ad Hoc Networks. Internet-draft, IETF MANET Working Group, July 2002c. URL http://tools.ietf.org/html/draft-ietf-manet-zone-ierp-02. Expiration : January 2003.

Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Intrazone Routing Protocol (IARP) for Ad Hoc Networks. Internet-draft, IETF MANET Working Group, July 2002d. URL http://tools.ietf.org/html/draft-ietf-manet-zone-iarp-02. Expiration : January 2003.

Xiaoyan Hong, Kaixin Xu, and Mario Gerla. Scalable Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, 16(4) :11–21, 2002. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=1020231.

Ting Chao Hou and Tzu Jane Tsai. An Access-Based Clustering Protocol for Multihop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications,*, 19(7) :1201–1210, 2001. ISSN 0733-8716. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=932689.

Atsushi Iwata, Ching Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu wei Chen. Scalable Routing Strategies for Ad Hoc Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17 :1369 –1379, 1999. ISSN 0733-8716. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=779920.

Philippe Jacquet, Paul Mï¿œhlethaler, Thomas H. Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Proceedings of the 2001 IEEE International Multi Topic Conference 2001 : Technology for the 21st Century (INMIC'01)*, pages 62–68, Pakistan, December 28-30 2001. IEEE. ISBN 0-7803-7406-1. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=779920.

Philippe Jacquet, Anis Laouiti, Pascale Minet, and Laurent Viennot. Performance of Multipoint Relaying in Ad Hoc Mobile Routing Protocols. In *Proceedings of the 2nd International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols ; Performance of Computer and Communication Networks ; and Mobile and Wireless Communications (NETWORKING '02)*, pages 387–398, 2002. URL http://hipercom.inria.fr/olsr/networking2002.ps.

Geetha Jayakumar and G. Gopinath. Ad Hoc Mobile Wireless Networks Routing Protocols : A Review. *Journal of Computer Science*, 3(8) :574–582, 2007. ISSN 1549-3636.

Mingliang Jiang, Jinyang Li, and Y.C. Tay. Cluster Based Routing Protocol(CBRP). Internet-draft, IETF MANET Working Group, August 14 1999. URL http://tools.ietf.org/html/draft-ietf-manet-cbrp-spec-01.

D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), feb 2007. URL http://www.ietf.org/rfc/rfc4728.txt.

David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Henry F. Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Series in Engineering and Computer Science*, pages 153–181. Springer US, 1996. ISBN 978-0-585-29603-6. URL http://dx.doi.org/10.1007/978-0-585-29603-6_5.

U.C. Kozat and al. Virtual dynamic backbone for mobile ad hoc networks. *IEEE ICC'01*, 1 :1201–10, 2001.

Taek Jin Kwon, Mario Gerla, V.K. Varma, M. Barton, and T.R. Hsing. Efficient Flooding with Passive Clustering an Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks. *Proceedings of the IEEE*, August 2003.

Myung Jong Lee, Jianling Zheng, Xuhui Hu, Hsin-hui Juan, Chunhui Zhu, Yong Liu, June Seung Yoon, and T.N. Saadawi. A new taxonomy of routing algorithms for wireless mobile ad hoc networks : the component approach. *Communications Magazine, IEEE*, 44(11) :116–123, November 2006. ISSN 0163-6804. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4014483.

Chun-hung Richard Lin and Mario Gerla. A Distributed Control Scheme in Multi-hop Packet Radio Networks for Voice/Data Traffic Support. In *Procedings of the 1995 IEEE International Conference on Communications (ICC'95)*, volume 2, pages 1238–1242, Seattle, WA, USA, June 18-22 1995. ISBN 0-7803-2486-2. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=524297.

Chunhung Richard Lin and Mario Gerla. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 15(7) :1265–75, 1997. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=622910.

Hwa Chun Lin and Yung-Hua Chu. A Clustering Technique for Large Multihop Mobile Wireless Networks. In *the 2000 IEEE 51st Vehicular Technology Conference Proceedings (VTC 2000)*, volume 2, pages 1545–1549, Tokyo , Japan, May 15-18 2000. ISBN 0-7803-5718-3. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=851386.

Beigh Bilal Maqbool and M.A. Peer. Classification of current routing protocols for ad hoc networks - a review. *International Journal of Computer Applications*, October .

Mahesh K. Marina and Samir Ranjan Das :. On-Demand Multi Path Distance Vector Routing in Ad Hoc Networks. In *Procedings of the 9th International Conference on Network Protocols (ICNP'01)*, pages 14–23, Riverside, CA, USA, November 11-14 2001. IEEE Computer Society. ISBN 0-7695-1429-4. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=992756.

Alexandre Delye de Clauzade de Mazieux, Michel Marot, and Monique Becker. An analysis of the generalised Max-Min d-cluster formation heuristic. In *The Sixth Annual Mediterranean Ad Hoc Networking WorkShop (Med-Hoc-Net'07)*, pages 114–120, Corfu, Greece, June 12-15 2007. URL http://di.ionio.gr/medhocnet07/wp-content/uploads/papers/190.pdf.

A. Bruce McDonald and Taieb Znati. A Mobility-based Frame Work for Adaptive Clustering in Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications,*, 17(8) :1466 –1487, August 1999. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=780353.

Nathalie Mitton, Anthony Busson, and Eric Fleury. Self-organization in Large Scale Ad Hoc Networks. In *Procedings of the Mediterranean ad hoc Networking Workshop (MED-HOC-NET'04)*, Bodrum, Turquie, June 2004. URL http://hal.archives-ouvertes.fr/hal-00383714/en/.

Shree Murthy and J.J. Garcia-Luna-Aceves. A Routing Protocol for Packet Radio Networks. In *Proceedings of the 1st annual International Conference on Mobile Computing and Networking (MobiCom'95)*, pages 86–95, Berkeley, California, USA, 1995. ACM. ISBN 0-89791-814-2. URL http://doi.acm.org/10.1145/215530.215560.

Radhika Nagpal and Daniel Coore. An Algorithm for Group Formation in an Amorphous Computer. In *Proceedings of the 11th International Conference on Parallel and Distributed Computing Systems (PDCS'98)*, Chicago, Illinois, USA, September 2-4 1998.

Navid Nikaein, Houda Labiod, and Christian Bonnet. DDR : Distributed Dynamic Routing Algorithm for Mobile Ad Hoc Networks. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'00)*, pages 19–27, Boston, Massachusetts, 2000. IEEE Press. ISBN 0-7803-6534-8. URL http://dl.acm.org/citation.cfm?id=514151.514155.

Tomoyuki Ohta, Shinji Inoue, and Yoshiaki Kakuda. An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks. In *Proceedings of the The Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, pages 293–300, Washington, DC, USA, April 9-11 2003. IEEE Computer Society. ISBN 0-7695-1876-1. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=1193960.

Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the (INFOCOM'97) 16th Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, volume 3, pages 1405–1413, Washington, DC, USA, April 7-12 1997. IEEE Computer Society. ISBN 0-8186-7780-5.

Marc R. Pearlman and Zygmunt J. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE Journal on Selected Areas in Communications*, 17 :1395–1414, 1999. ISSN 0733-8716. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=779922.

Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye State Routing : A Routing Scheme for Ad Hoc Wireless Networks. In *Proceedings of the 2000 IEEE International Conference on Communications (ICC 2000)*, volume 1, pages 70 –74, New Orleans, Louisiana, USA, June 18-22 2000. IEEE Communications Society. ISBN 0-7803-6283-7. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=853066.

Charles E. Perkins and Elizabeth M. Belding-Royer. Perkins99. In *Proceedings of the 2nd Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, New Orleans, LA, USA, February 25-26 1999. IEEE Computer Society. ISBN 0-7695-0025-0.

Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Ranjan Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003. URL http://www.ietf.org/rfc/rfc3561.txt.

Arvind Ramalingam, Sundarpremkumar Subramani, and Karthik Perumalsamy. Associativity-based Cluster Formation and Cluster Management in Ad Hoc Networks. In *Proceedings of the International conference on high performance computing (HiPC'02)*, Bangalore, India, December 18-21 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.4802.

Francisco J. Ros and Pedro M. Ruiz. Cluster-based OLSR Extension to Reduce Control Overhead in Mobile Ad Hoc Networks. In *Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing (IWCMC'07)*, pages 202–207, Honolulu, Hawaii, USA, August 12-16 2007. ACM. ISBN 978-1-59593-695-0. URL http://doi.acm.org/10.1145/1280940.1280985.

John Francis Sharmila and Elijah Blessing Rajsingh. Performance Analysis of Clustering Protocols in Mobile Ad hoc Networks. *Journal of Computer Science*, 4 :192–204, 2008. ISSN 1549-3636. URL http://citeseer.ist.psu.edu/viewdoc/summary;jsessionid=0AC32152FBE8E8484B279776CA23C38A?doi=10.1.1.165.8963.

Chai-Keong Toh. Novel Distributed Routing Protocol to Support Ad-hoc Mobile Computing. In *Proceedings of the 1996 IEEE 15th Annual International Phoenix Conference on Computers and Communications (PCCC'96)*, pages 480–486, Scottsdale, AZ , USA, March 27-29 1996. IEEE.

Dali Wei and H. Anthony Chan. Clustering Ad Hoc Networks : Schemes and Classifications. In *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON'06)*, volume 3, pages 920–926, Reston, Virginia, USA, September 28-28 2006. IEEE Communications Society. ISBN 1-4244-0626-9. URL http://ieeexplore.ieee.org//xpls/abs_all.jsp?arnumber=4068392.

Jie Wu and Hailan Li. A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks. *Telecommunication Systems*, 18 :13–36, 2001. ISSN 1018-4864. URL http://www.springerlink.com/content/m0w810731hu36610/.

Jie Wu, Wei Lou, and Fei Dai. Extended Multipoint Relays to Determine Connected Dominating Sets in MANETs. *IEEE Transactions on Computers*, 55 :334–347, 2006. ISSN 0018-9340. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1583562;http://doi.ieeecomputersociety.org/10.1109/TC.2006.40.

Ossama Younis and Sonia Fahmy. HEED : A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks. *IEEE Transactions on Mobile Computing*, 3 :366–379, October 2004. ISSN 1536-1233. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1347100.

Jane Y. Yu and Peter H. J. Chong. 3hBAC (3-Hop Between Adjacent Clusterheads) : A Novel Non-Overlapping Clustering Algorithm for Mobile Ad Hoc Networks. In *Proceedings of the 2003 IEEE Pacific Rim Conference On Communications, Computers, And Signal Processing*, volume 1 of *(PACRIM 2003)*, pages 318–321, Victoria, Canada, August 28-30 2003. ISBN 0-7803-7978-0. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1235781.

Jane Y. Yu and Peter H. J. Chong. A Survey of Clustering Schemes for Mobile Ad Hoc Networks. *IEEE Communications Surveys and Tutorials*, 7(1) :32–48, First Qtr 2005. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1423333.

Clustering Algorithms and Routing Protocols in Wireless Mobile Networks.

Scalability is one of critical challenges for routing protocols in large scale mobile wireless networks. In this context, clustering technique seems a promising approach to overcome the scalability problem. First, we propose a one hop clustering algorithm, alpha-SSCA (alpha-Stability Structure Clustering Algorithm), which aims to improve the stability of the clusters structure. Second, we present a proposal of a generic K-hops clustering algorithm which is independent of the metrics used to elect cluster-heads. The main contribution of this last algorithm is to enhance the stability of the clusters structure while reducing the number of clusters. Clustering mechanism is introduced in our proposed routing protocol CLSR (Cluster-based Link State Routing) in order to reduce the control overhead. The main objective of CLSR is to take profit of the stable structure of clusters to enhance the network scalability. We propose also a second proactive link-state protocol which is based on hierarchical clustering. This protocol makes use of hierarchical clustering to more reduce the routing overhead.

Keywords : Routing, Scalability, Clustering, Mobile wireless Networks..

Algorithmes de clusterisation et protocoles de routage dans les réseaux mobiles sans-fil

Le passage à l'échelle des protocoles de routage est un des problèmes les plus critiques pour les réseaux mobiles sans fil à grande envergure. Dans ce cadre, le concept de clusterisation peut être mis à profit dans la fonction de routage afin d'améliorer les performances de ces réseaux. En premier lieu, cette thèse présente notre algorithme de clusterisation à 1-saut alpha-SSCA (α-Stability Structure Clustering Algorithm) qui a pour objectif d'améliorer la stabilité de la structure des clusters. Un algorithme générique de clusterisation à K-sauts est également proposé en ayant le même but de stabilité visé par alpha-SSCA tout en réduisant le nombre de clusters générés et en étant indépendant de la métrique d'élection des cluster-heads. Ensuite, nous présentons notre proposition d'un protocole de routage à état des liens des clusters qui exploite les apports de notre mécanisme de clusterisation α-SSCA. Ce protocole, appelé CLSR (Cluster-based Link State Routing), vise à réduire le trafic de contrôle afin d'améliorer les performances du réseau à large échelle. Nous avons proposé aussi une version hiérarchique du protocole CLSR. Ce protocole de routage introduit une hiérarchie dans la structure des clusters qui permet de réduire le nombre de clusters en groupement des clusters proches dans un même cluster. L'objectif principal de ce protocole hiérarchique est d'améliorer la scalabilité de CLSR quand le nombre de nœuds dans le réseau augmente considérablement.

Mots-clés : clusterisation, Routage, scalabilité, réseaux mobiles sans-fil.
