CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Introduction

Ce chapitre est composé de trois parties : dans la première partie, nous allons présenter certains protocoles de routage développés dans le cadre du groupe de travail MANET de l'IETF. Ces protocoles sont définis au niveau IP et sont donc indépendants des couches physiques et MAC. Le routage IP permet en particulier une interconnectivité aisée avec toutes sortes d'autres réseaux ou matériel. Les protocoles présentés sont parmi les plus représentatifs des diverses techniques utilisées pour le routage dans les réseaux *ad hoc*.

Un protocole de routage a pour but d'établir et de maintenir des routes entre sources et destinations, pour que les messages soient correctement délivrés. Les caractéristiques des réseaux *ad hoc* rendent l'utilisation des protocoles filaires classiques inadéquate. Les protocoles de l'Internet ont été conçus pour des réseaux ayant des topologies statiques, et sont exécutés par des nœuds dotés de suffisamment de ressources. Les protocoles de routage dans les réseaux *ad hoc* opèrent dans des réseaux dont les changements de topologie sont fréquents, et sont exécutés sur des équipements ayant des contraintes de ressources (de batterie, de mémoire, de CPU, etc.).

La deuxième partie discute le problème de configuration et adressage dans les réseaux *ad hoc*. La nature des réseaux *ad hoc* nécessitent de nouvelles approches pour effectuer la configuration et l'adressage. Nous allons présenter les principales approches étudiées dans la littérature.

Dans la troisième partie de ce chapitre, nous nous intéressons à l'étude du contrôle de topologie. Depuis quelques années, plusieurs travaux de recherche ont été consacrés à l'étude du contrôle de la topologie dans les réseaux *ad hoc*. Ces travaux visent à

maintenir une topologie adéquate tout au long du fonctionnement du réseau. Le but est d'atteindre un ensemble d'objectifs comme :

- effectuer la mise à jour des informations de routage de manière plus efficace,
- échanger les paquets de contrôle plus rapidement,
- utiliser de manière optimale la bande passante dans le réseau tout en éliminant le trafic redondant,
- réduire la consommation d'énergie.

2.2 Protocoles de routage dans les réseaux *ad hoc*

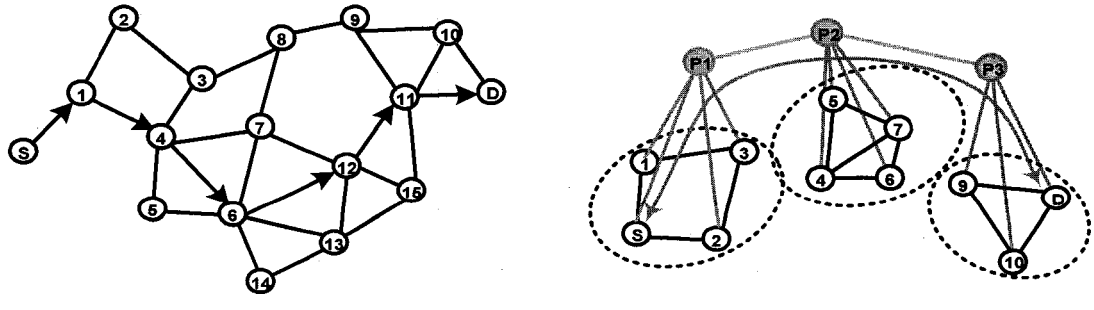
L'étude et la mise en œuvre de protocoles de routage dans les réseaux *ad hoc* doivent tenir compte d'un certain nombre de points :

- minimiser la charge du réseau : l'optimisation des ressources du réseau renferme deux autres sous-problèmes comme par exemple les boucles de routage et la concentration du trafic autour de certains nœuds ou liens.
- assurer un routage efficace : la stratégie de routage doit créer des chemins optimaux et pouvoir prendre en compte différentes métriques de coûts (bande passante, nombre de liens, ressources du réseau, etc.). Si la construction des chemins optimaux est un difficile problème, la maintenance de tels chemins peut devenir encore plus complexe et la stratégie de routage doit assurer une maintenance efficace des routes avec le moindre coût possible.
- minimiser le latence : certaines applications peuvent être sensibles au délai. Il est important alors de minimiser le délai de bout en bout.

2.2.1 Routage plat ou hiérarchique

Les protocoles de routage pour les réseaux *ad hoc* peuvent être classés suivant différents critères. Le premier concerne le type de voisin qu'ils possèdent et les rôles qu'ils accordent aux différents mobiles.

- **Les protocoles de routage plat** considèrent que tous les nœuds sont égaux (voir figure 8-a). La décision d'un nœud d'acheminer du trafic pour un autre dépendra de sa position et pourra être remise en cause dans le temps.
- **Les protocoles de routage hiérarchique** fonctionnent en confiant aux mobiles des rôles qui varient de l'un à l'autre. Certains nœuds sont alors élus pour assumer des fonctions spécifiques qui conduisent à une vision en plusieurs niveaux de la topologie du réseau. Par exemple, un nœud servira de passerelle pour un certain nombre de nœuds qui lui seront attachés. Le routage sera alors simplifié, puisqu'il se fera de passerelle à passerelle, jusqu'à celle directement attachée à la destination, figure 8-b. Dans ce type de protocole, les passerelles supportent la majeure partie de la charge de routage. Dans les réseaux où certains nœuds s'avèrent très sédentaires et disposent suffisamment d'énergie (par exemple, un réseau d'ordinateurs portables où certains sont reliés au secteur, des stations de base disposées là pour garantir la connectivité, etc.).



a- routage plat b- routage hiérarchique

Figure 8 Routage plat *vs* routage hiérarchique

2.2.2 État de lien ou vecteur de distance

Une autre classification, inspirée du monde filaire, est possible pour les protocoles de routage *ad hoc* :

- **Les protocoles à état de lien** cherchent à maintenir dans chaque nœud une image plus ou moins complète du réseau où figurent les nœuds et les liens les reliant. À partir de cette image, il est possible de construire les tables de routage. Un des avantages de ce type de protocole est leur capacité à pouvoir trouver des routes alternatives lorsqu'un lien n'est plus accessible. Il est même possible d'utiliser simultanément plusieurs routes pour la même destination, augmentant ainsi la répartition de la charge et la tolérance aux pannes dans le réseau. Cependant, dans le cas d'un réseau étendu, la quantité d'informations à stocker et à diffuser devient considérable.
- **Les protocoles à vecteur de distance**, plutôt que de maintenir une image complète du réseau, ne conservent que la liste des nœuds du réseau et l'identité du voisin par lequel passer pour atteindre la destination par le chemin le plus court. À chaque destination possible est donc associée le prochain nœud et une distance. Si un voisin envoie un paquet de contrôle dans lequel il indique être plus près d'une destination que le prochain nœud que l'on utilisait jusqu'alors il le remplace dans la table de routage. Un des inconvénients de cette technique est qu'il est difficile de conserver plusieurs routes alternatives (on ne dispose que de l'information sur le prochain nœud et on ne sait pas si la suite de la nouvelle route est indépendante de celle qui a été brisée).

2.2.3 Taxonomie des protocoles de routage *ad hoc*

La figure 9 présente une taxonomie des protocoles de routage pour les réseaux *ad hoc*. Ces protocoles se différencient d'abord par le niveau d'implication des nœuds dans le routage. Ils sont uniformes si tous les nœuds du réseau jouent le même rôle pour la

fonction de routage. À l'inverse, ils peuvent être non-uniformes si une structure hiérarchique est donnée au réseau et que seuls certains nœuds assurent le routage. Ainsi, dans les protocoles à sélection de voisins, chaque nœud sous-traite la fonction de routage à un sous-ensemble de ses voisins directs. Pour les protocoles à partitionnement, le réseau est découpé en zones dans lesquelles le routage est assuré par un unique nœud maître.

Les protocoles de routage uniforme peuvent également être regroupés selon les données qu'ils utilisent pour effectuer leur tâche. Dans les protocoles orientés topologie et à état de lien, chaque nœud utilise comme données l'état de ses connexions avec ses nœuds voisins; cette information est ensuite transmise aux autres nœuds pour leur donner une connaissance plus précise de la topologie du réseau.

Les protocoles orientés destinations (*vector protocols*) maintiennent pour chaque nœud destination une information sur le nombre de nœuds qui les en séparent (la distance) et éventuellement sur la première direction à emprunter pour y arriver (le vecteur).

Avec un protocole proactif, les routes sont disponibles immédiatement. Cependant, le trafic induit par les messages de contrôle et de mise à jour des tables de routage peut être important et partiellement inutile. De plus, la taille des tables de routage croît linéairement en fonction du nombre de nœuds. À l'opposé, dans le cas d'un protocole réactif, aucun message de contrôle ne charge le réseau pour des routes inutilisées. Mais, pour ces dernières, la mise en place d'une route par inondation peut être coûteuse et provoquer des délais importants avant l'ouverture de la route.

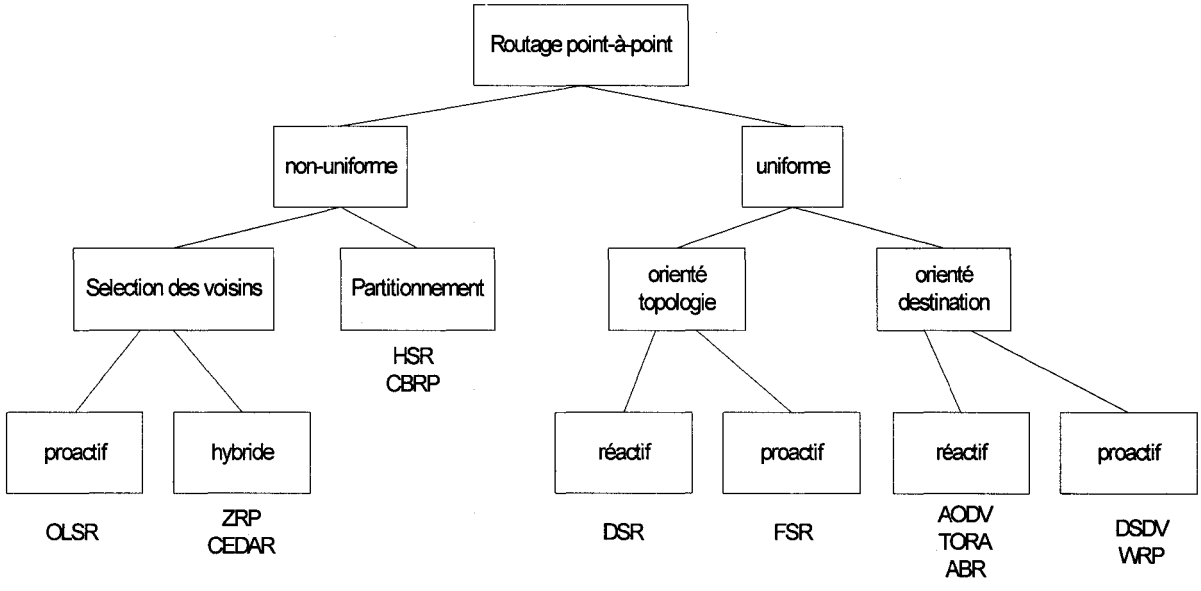


Figure 9 Taxonomie des protocoles de routage pour les réseaux *ad hoc*

2.3 Description des protocoles de routage

Dans cette section nous décrirons différents protocoles de routage dans les réseaux *ad hoc*. Nous choisissons les plus significatifs parmi les trois catégories suivantes : les protocoles proactifs, les protocoles réactifs et les protocoles hybrides.

2.3.1 Les protocoles proactifs

Le principe de base de ces protocoles est de maintenir à jour les tables de routage, de manière à ce que lorsqu'un mobile désire transmettre un paquet à un autre mobile, une route soit immédiatement déterminée. Ces tables de routage sont obtenues grâce à des messages envoyés périodiquement ou déclenchés à la suite d'événements importants pour le réseau.

Dans le contexte des réseaux *ad hoc* les nœuds peuvent apparaître ou disparaître de manière aléatoire et par la suite la topologie change. Cela signifie qu'il faut un échange

continuel d'information pour que chaque nœud ait une mise à jour de l'état du réseau. Les tables sont donc maintenues au moyen des paquets de contrôle, et il est possible d'y trouver directement et à tout moment une route vers chaque destination en fonction de différents critères. On peut par exemple privilégier les routes comportant moins de sauts, celles qui garantissent une bande passante minimale, ou encore celles où le délai est le plus faible. L'avantage dans ces protocoles est d'avoir les routes immédiatement disponibles quand les applications en ont besoin, mais cela se fait au coût d'échanges continus de messages (consommation de bande passante) qui ne sont certainement pas tous nécessaires (seules certaines routes seront utilisées par les applications en général). Comme exemple de protocoles réactifs, on trouve OLSR, DSDV, WRP, etc.

2.3.1.1 DSDV (*Destination-Sequenced Distance Vector Protocol*)

Ce protocole est basé sur l'idée classique de l'algorithme distribué de Bellman-Ford en rajoutant quelques améliorations [Perkins et Bhagwat (1994)]. Chaque station mobile maintient une table de routage qui contient :

- toutes les destinations possibles,
- le nombre de nœuds (ou de sauts) nécessaires pour atteindre la destination,
- le numéro de séquence, *SN* (*sequence number*) qui correspond à un noeud destination.

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

La mise à jour dépend donc de deux paramètres : le temps, c'est-à-dire la période de transmission, et les événements, par exemple : l'apparition d'un nœud, la détection d'un nouveau voisin etc. La mise à jour doit permettre à une unité mobile de pouvoir localiser, dans la plupart des cas, une autre unité de réseau.

Un paquet de mise à jour contient :

- le nouveau numéro de séquence incrémenté du nœud émetteur ;

et pour chaque nouvelle route :

- l'adresse de la destination,
- le nombre de nœuds (ou de sauts) séparant le nœud de la destination,
- le numéro de séquence (des données reçues de la destination) tel qu'il a été estampillé par la destination.

Ce protocole DSDV offre la disponibilité des chemins à toutes les destinations à n'importe quel moment ce qui engendre un délai minimal pendant le processus d'établissement du chemin. D'autre part, le mécanisme de mise à jour avec les *flags* de *SN* fait en sorte que le réseau filaire actuel est adaptable aux réseaux *ad hoc*. Le DSDV élimine les deux problèmes de boucle de routage (*routing loop*), et celui du *counting to infinity*. Par contre, DSDV souffre d'un *overhead* de contrôle excessif qui est proportionnel au nombre de nœuds dans le réseau. Par conséquent, il ne garantit pas la mise en échelle dans le réseau *ad hoc*. De plus, pour obtenir une information particulière pour un nœud de destination, la source doit attendre le message de mise à jour envoyé par le nœud de destination. Ce délai peut engendrer des informations inexactes dans les nœuds.

2.3.1.2 OLSR (*Optimized Link State Routing Protocol*)

OLSR est un protocole proactif, non uniforme et basé sur la sélection de voisins [Jacquet *et al.* (2003)]. C'est une optimisation de l'algorithme classique appelé «à états des liens». Le protocole repose sur le concept clef de relais multipoint MPR (MultiPoint Relay). Les MPR d'un nœud correspondent à l'ensemble des voisins qui permettent d'atteindre tous les nœuds situés à deux sauts. La diffusion des différents messages de contrôle se

fait uniquement vers les MPR (voir la figure 10). Cette technique permet de réduire les transmissions inutiles. D'autre part, OLSR distingue les liens unidirectionnels des liens bidirectionnels, seuls utilisés pour le routage.

Chaque nœud maintient de l'information sur les nœuds qui l'ont élu en tant que MPR. Ceci est fait grâce à des messages de présence (les messages *hello*) envoyés par chaque nœud à ses voisins. Ces messages contiennent :

- la liste des nœuds que l'émetteur a choisis comme MPR,
- la liste des nœuds avec lesquels l'émetteur possède des liens bidirectionnels,
- la liste des nœuds que l'émetteur peut entendre (reliés par des liens unidirectionnels).

La diffusion de ces messages permet aux nœuds du réseau de maintenir, dans leur table des voisins, les visions à deux sauts et de calculer l'ensemble de leurs MPR. Cet ensemble est recalculé dès qu'un changement est détecté dans le voisinage à deux sauts.

La diffusion sur la totalité du réseau (via les MPR) de messages de contrôle de topologie (*Topology Control*, TC) donne l'information topologique nécessaire au routage. Ces messages contiennent, pour chaque MPR, la liste des nœuds qui l'ont choisi. Grâce à ces messages, les nœuds peuvent maintenir une table de topologie (*Topology Table*), indiquant le dernier saut pour chaque destination.

Un algorithme de plus court chemin, appliqué à la table des voisins et à la table de topologie, permet de construire la table de routage de chaque nœud. Cette table mémorise, pour tous les nœuds du réseau, le nombre de sauts et le premier saut pour l'atteindre. Elle doit être recalculée dès que l'une des deux tables sources est modifiée. OLSR fournit des routes optimales en nombre de sauts. Il convient pour des grands réseaux grâce à son mécanisme de MPR, mais est moins efficace pour de petits réseaux.

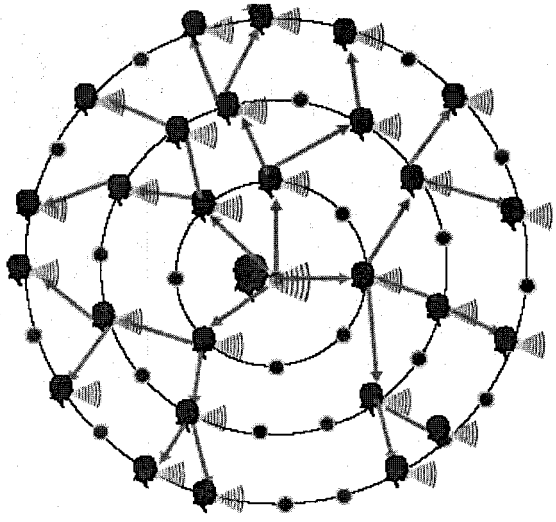


Figure 10 Inondation par les relais multipoint dans OLSR[10]

2.3.1.3 TBRPF (*Topology Dissemination Based on Reverse-Path Forwarding*)

Dans ce protocole, chaque nœud maintient en permanence un arbre dont il est la racine et qui fournit les chemins les plus courts pour tous les autres nœuds du réseau [Ogier *et al.* (2004)]. TBRPF est constitué de deux parties complémentaires : la découverte des voisins et le routage.

La découverte des voisins est assurée par un mécanisme de paquets *hello* diffusés régulièrement au voisinage direct. Ces paquets *hello* contiennent la liste des voisins du nœud, et permettent ainsi de connaître rapidement la topologie complète du réseau à deux sauts. Il faut noter que TBRPF utilise une technique de *hello* différentielle ou seuls les changements de topologie sont notifiés (diminuant ainsi la taille moyenne des paquets et autorisant leur envoi à une plus grande fréquence).

---

[10] www.ares.insa-lyon.fr/tarot/download/PhilippeJacquet_03_01.ppt

Le routage est basé sur un échange des arbres de routage entre nœuds voisins, conduisant progressivement à la diffusion de l'information dans l'ensemble du réseau. Là encore seules des parties d'arbres sont échangées. Normalement, un nœud ne diffuse qu'un sous-arbre à deux niveaux dont il est la racine. Au premier niveau apparaissent les liens vers tous les voisins directs du nœud, et au deuxième niveau un unique lien vers chaque voisin à deux sauts (on peut noter ici une certaine similitude avec le mécanisme des relais multipoints d'OLSR). En conjonction avec ce système de base, TBRPF peut également ajouter des informations sur d'autres liens à l'arbre diffusé, avant de réagir plus vite en cas de changement de la topologie. À noter enfin que dans un souci d'économie de bande passante, les sous-arbres et les paquets *hello* sont regroupés autant que possible dans un même paquet (on parle d'agrégation ou *piggybacking* puisque l'on profite des paquets *hello* pour envoyer en même temps les sous-arbres).

2.3.1.4 WRP (*Wireless Routing Protocol*)

Le WRP est un protocole basé sur l'utilisation des algorithmes de recherche des chemins, PFA (*Path-Finding Algorithm*) [Murthy et Garcia-Luna-Aceves (1996)]. Ces algorithmes utilisent des données sur la longueur et le nœud prédécesseur du chemin le plus court, correspondant à chaque destination; et cela pour éviter le problème du (*counting to infinity*). Le problème avec les algorithmes PFA est la présence des boucles de routage temporaires dans le chemin spécifié par le prédécesseur, avant qu'ils ne convergent. Pour résoudre ce problème, WRP utilise un algorithme de recherche de chemins qui permet de réduire les situations des boucles temporaires qui limitent les mises à jour, uniquement aux changements significatifs des entrées de la table de routage.

Dans le protocole WRP, chaque nœud du réseau maintient quatre tables :

- table de distance (la distance entre le nœud et les différents correspondants),

- table de prochain saut (le voisin à joindre pour contacter un mobile dans le réseau),
- table de coût (la latence entre le nœud et les différents destinataires),
- table de retransmission de messages, MRL (*Message Retransmission List*) (avec les informations de mise à jour).

Chaque mobile émet régulièrement un message indiquant les changements dans sa table de routage; il diffuse également des demandes de confirmation de présence à ses voisins pour les informer des changements topologiques tout en vérifiant la validité de son voisinage.

Le protocole WRP possède les mêmes avantages que le DSDV. En plus de ces avantages, il possède une mise à jour rapide des tables ce qui minimise la complexité de maintenance des tables multiples. Cependant, il a des inconvénients tels que par exemple : la nécessité d'une grande mémoire et un long temps de traitement. Pour une mobilité importante, l'*overhead* de contrôle intervient dans la mise à jour des tables ce qui n'est pas souhaitable pour les réseaux de taille importante et les réseaux ayant un grand dynamisme.

2.3.2 Les protocoles réactifs

Le principe d'un protocole réactif consiste à ne rien faire tant qu'une application ne demande pas explicitement d'envoyer un paquet vers un nœud distant. Cela permet d'économiser de la bande passante et de l'énergie. En revanche, il peut s'écouler un temps plus ou moins long avant que la route ne soit construite et que le mobile ne puisse envoyer ses paquets de données. Lorsqu'un paquet doit être transmis, le protocole de routage va déterminer une route jusqu'à la destination. Une fois la route déterminée, elle sera inscrite dans la table de routage et pourra être utilisée. D'une façon générale, cette recherche se fait par inondation (un paquet de recherche de routes est transmis de proche

en proche dans tout ou partie du réseau). L'avantage de cette méthode est qu'elle ne génère du trafic de contrôle que lorsqu'il est nécessaire. Les principaux inconvénients sont que l'inondation est un mécanisme coûteux qui va faire intervenir tous les nœuds du réseau en très peu de temps et qu'il va y avoir un délai d'établissement des routes. Comme exemple de protocoles réactifs, on trouve AODV, DSR, TORA, LAR, etc.

2.3.2.1 DSR (*Dynamic Source Routing*)

Dans ce protocole, le routage est basé sur le routage source [Johnson et Maltz (1996)]; la source des données détermine la séquence complète des nœuds à travers lesquels, les paquets de données seront acheminés. Le processus de découverte de routes se fait comme suit; la source diffuse un paquet *RREQ* (*Route REQuest*). Si l'opération de découverte est réussie, la source reçoit un paquet réponse de route, RREP (*Route REPly*), qui contient la séquence de nœuds à travers lesquels la destination peut être atteinte. Le paquet requête de route contient donc un champ enregistrement de route, dans lequel sera accumulée la séquence des nœuds visités durant la propagation de la requête dans le réseau (figure 44).

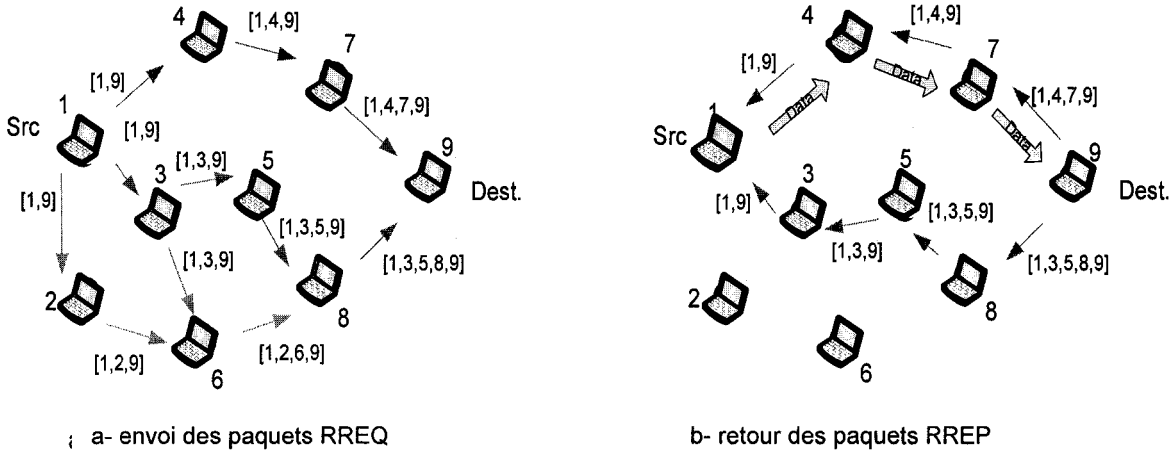


Figure 11 Le principe de découverte d'une route dans DSR

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

- quand un nœud détecte un problème fatal de transmission, à l'aide de sa couche de liaison, un message RERR (*Route ERRor*) est envoyé à l'émetteur original du paquet,
- 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 message RERR par l'hôte 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 l'émetteur.

Le protocole DSR présente les avantages suivants :

- L'utilisation de la technique «routage source», fait en sorte 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,

- Il autorise à la source de conserver dans sa table de routage plusieurs chemins valides pour la même destination, ce qui peut être utile dans le cas de ruptures de liens,
- L'absence totale de boucle de routage, car le chemin source–destination fait partie des paquets de données envoyés.

Cependant, le temps d'établissement d'une route est beaucoup plus important que pour les protocoles proactifs. De plus, l'*overhead* pour le routage est important parce que le routage est initié par la source. Cet *overhead* est directement proportionnel à la longueur du chemin et par la suite il y a un surcoût dans la signalisation. DSR performe mieux dans un environnement statique ou à faible mobilité, cette performance se dégradant quand la mobilité augmente.

2.3.2.2 AODV (*Ad hoc On-demand Distance Vector Protocol*)

Le protocole AODV représente essentiellement une amélioration de l'algorithme DSDV que nous avons décrit dans la section 2.3.1.1 dans un contexte réactif [Perkins *et al.* (2003)]. Contrairement à DSDV qui maintient toutes les routes possibles, le protocole AODV réduit le nombre des messages diffusés en créant les routes en cas de besoin. Il utilise le principe des numéros de séquences afin de maintenir la consistance des informations de routage. Les numéros de séquence permettent d'utiliser les routes les plus récentes (*fresh routes*).

Le protocole AODV repose sur trois principales composantes :

- l'initiation et la propagation des messages RREQ,
- l'initiation et la propagation des messages RREP
- la maintenance des tables « vecteurs à distance ».

AODV utilise le message RREQ dans le but de trouver un chemin vers une destination. La route peut ne pas exister si la destination n'est pas connue au préalable, ou si le chemin existant vers la destination a expiré ou est devenu défaillant. Cependant, AODV maintient les routes d'une façon distribuée en conservant une table de routage au niveau de chaque nœud de transit appartenant à la route cherchée. Afin de maintenir des routes cohérentes, une transmission périodique du message *hello* est effectuée. Si au bout d'un certain temps aucun message *hello* n'est reçu à partir d'un nœud voisin, le lien en question est considéré défaillant.

Un nœud diffuse une requête de route dans le cas où il aurait besoin de connaître une route vers une certaine destination et qu'une telle route n'est pas disponible dans le cas où : (i) la destination n'est pas connue au préalable, (ii) la durée de vie du chemin existant vers la destination a été expirée ou le chemin est devenu défaillant.

Les avantages de ce protocole sont :

- il demande moins de bande passante,
- il performe mieux pour les réseaux de grande taille,
- il offre une convergence rapide quand la topologie du réseau change, car il évite la boucle de routage et il évite le problème de «*counting to infinity*» de Bellman-Ford.

Les inconvénients du protocole AODV sont :

- une seule requête *RREQ* peut générer plusieurs paquets *RREP* qui peuvent engendrer un nombre important de message de contrôle,
- les nœuds intermédiaires peuvent mener à des chemins incohérents; c'est le cas où le *SN* de la source n'a pas été mis à jour et que les nœuds intermédiaires possèdent des *SN* plus grands mais plus petits que le *SN* de la dernière destination.

2.3.2.3 TORA (*Temporary ORdered Algorithm*)

Ce protocole avait pour objectif de minimiser l'effet des changements de topologie qui sont fréquents dans les réseaux *ad hoc* [Park et Corson (2001)]. Afin de s'adapter à la mobilité, le protocole stocke plusieurs chemins vers une même destination, ce qui fait que beaucoup de changements de topologie n'auront pas d'effet sur le routage des données, à moins que tous les chemins qui mènent vers la destination soient perdus (rompus). Il est caractérisé par le fait que les messages de contrôle sont limités à l'ensemble des nœuds proches du lieu de l'occurrence du changement de la topologie.

Dans ce protocole, l'utilisation des meilleurs chemins a une importance secondaire. Les longs chemins peuvent être utilisés afin d'éviter le contrôle induit par le processus de découverte de nouveaux chemins. Il consiste à établir un graphe acyclique orienté (*Directed Acyclic Graph* DGA) dont la racine est la destination (c'est-à-dire que le nœud destination est le seul sans arc sortant). Ainsi, depuis chaque station, on peut retrouver la destination en suivant l'orientation du graphe. Un DGA est orienté destination s'il y a toujours un chemin possible vers une destination spécifiée. Il devient non orienté destination si un lien (ou plus) devient défaillant. Dans ce cas, le protocole utilise la technique d'inversement de liens. Ceci assure la transformation du graphe précédent en un graphe orienté destination dans un temps fini.

Afin de maintenir le DAG orienté destination, l'algorithme TORA utilise la notion de taille de nœud. Chaque nœud possède une taille qu'il échange avec l'ensemble de ses voisins directs. Un lien est toujours orienté vers le nœud qui a la plus grande taille vers le nœud qui a la plus petite taille. Dans ce protocole, on trouve quatre fonctions de base : création de routes, maintenance de routes, élimination de routes et optimisation de routes. La figure 12-a montre la création du DAG dans le protocole TORA : le nœud source diffuse le paquet *Query* (QRY) spécifiant l'identifiant de la destination, ID-destination, qui détermine le nœud pour lequel l'algorithme est exécuté. Un nœud qui a

une taille différente de *Null* répond par un paquet *Update* (UPD) qui contient sa taille, figure 12-b. La mise à jour des liens se fait de façon réactive et pour cette raison dans certaines classifications le protocole TORA fait partie des protocoles hybrides.

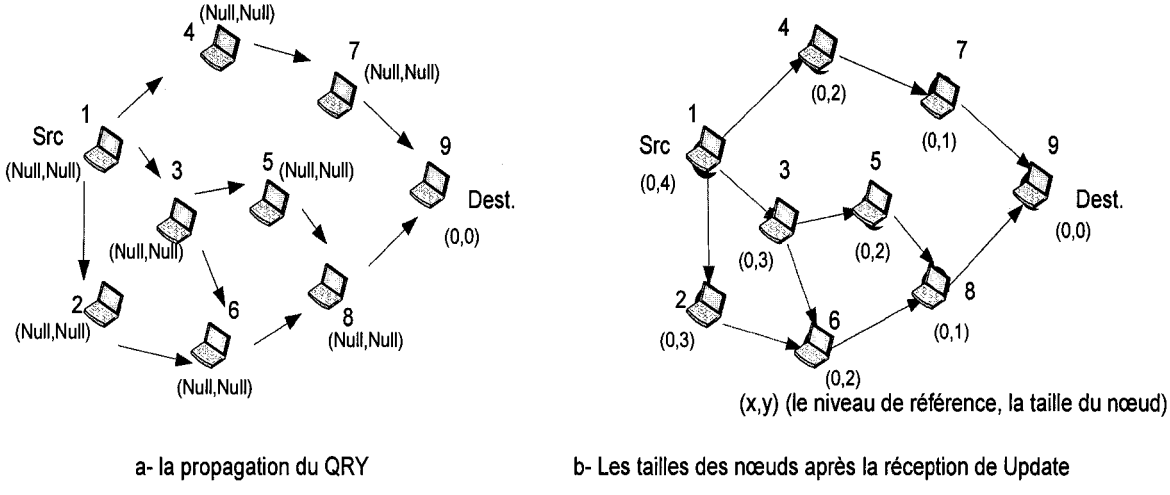


Figure 12 La création des routes dans le protocole TORA

Quand un nœud détecte une défaillance, c'est-à-dire l'invalidité d'un lien (sachant qu'il ne possède pas de suivants valides vers la destination), il lance un nouveau niveau de référence. L'objectif de ce nouveau niveau de référence consiste à indiquer à la source l'invalidité des chemins rompus. La figure 13 donne un exemple de ce processus. La fonction de suppression dans TORA est réalisée en diffusant un paquet CLR (*Clear*) dans le réseau afin de supprimer toutes les routes invalides qui sont sauvegardées par les nœuds du réseau.

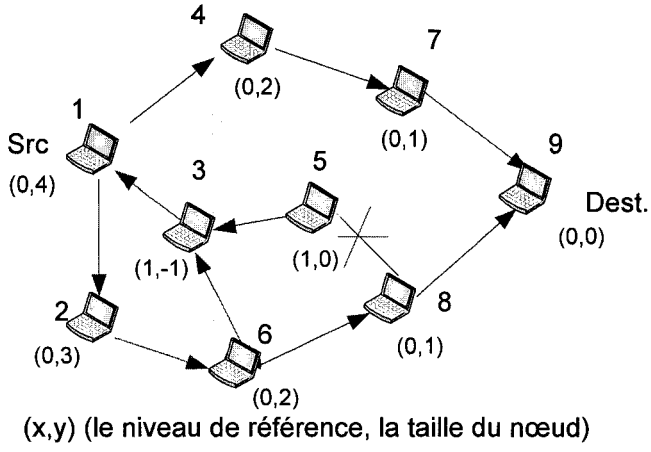


Figure 13 La réaction du protocole TORA à la défaillance du lien (5,8)

En limitant les paquets de contrôle pour la reconfiguration des chemins à petit rayon, TORA a pour avantage de générer moins d'*overhead.* Cependant, la détection de partition simultanée et la détection des routes subséquentes peuvent engendrer des oscillations temporaires. Avec TORA, la réparation des routes se fait localement, ce qui résulte en des chemins non-optimaux.

2.3.2.4 LAR (*Location-Aided Routing*)

Le protocole LAR appelé « Routage aidé par la localisation » est un protocole de routage réactif basé sur l'utilisation de localisation [Ko et Vaidya (1998)]. Ce protocole procède d'une manière très similaire au protocole DSR. La principale différence entre les deux protocoles réside dans le fait que le LAR utilise les informations des localisations, fournies par le système de positionnement global (*Global Positionning System*, GPS), dans le but de limiter l'inondation des paquets de requête de route. Afin d'assurer cela, deux approches peuvent être utilisées.

Dans la première approche, le nœud source définit une région circulaire dans laquelle la destination peut être localisée. La position et la taille de la région sont estimées en se basant sur :

- la position de la destination telle qu'elle est connue par la source,
- l'instant qui correspond à cette position,
- la vitesse moyenne du mouvement de la destination.

LAR permet un routage géographique classique et des fonctions intéressantes en optimisant le nombre de messages émis lors des routes non valides. Lorsqu'une route n'est plus valide, il est possible, avec ce protocole, de faire une découverte locale des routes quand l'information de localisation est disponible. Dans la zone de rupture, LAR route les paquets vers les nœuds du réseau les plus proches de la destination, tout en étant dans la zone de couverture du nœud précédent (figure 14). Une fois la nouvelle route établie, on utilise à nouveau le protocole de routage comme DSR.

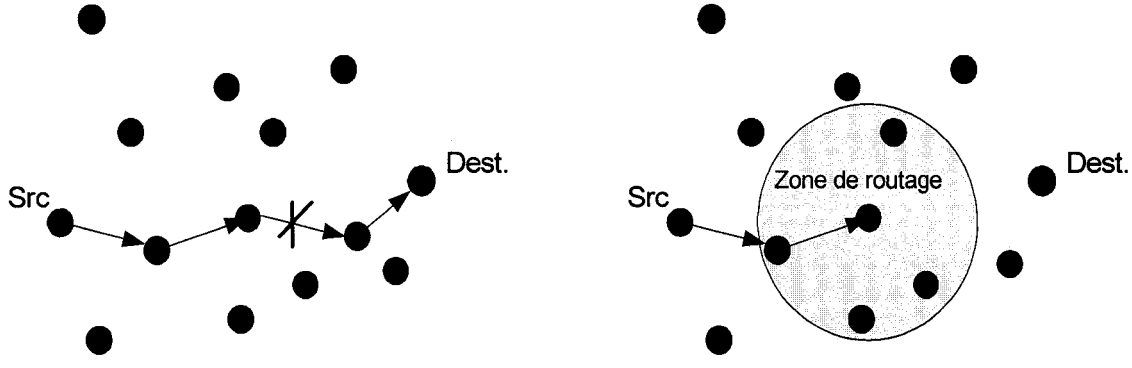


Figure 14 Exemple d'application du protocole LAR

Le protocole LAR a pour avantage de réduire l'*overhead* de contrôle en limitant la zone de recherche d'un chemin. L'utilisation efficace de l'information de position géographique réduit l'*overhead* de contrôle et augmente l'utilisation de la bande passante. Cependant, l'application de ce protocole dépend énormément de la disponibilité du GPS ou d'un autre mécanisme qui donne l'information sur la position. Il ne peut pas être applicable si de telles informations n'existent pas.

2.3.3 Les protocoles hybrides

Les deux approches décrites précédemment ne sont pas les seules alternatives pour les protocoles de routage dans les réseaux *ad hoc*. Une autre approche appelée hybride permet de combiner les approches proactives et réactives afin de bénéficier de la combinaison de leurs avantages.

Les protocoles hybrides combinent les deux approches précédentes, réactives et proactives. Le principe est de connaître le voisinage de manière proactive jusqu'à une certaine distance (trois ou quatre sauts) et si jamais une application cherche à envoyer un paquet à un nœud qui n'est pas dans cette zone, une recherche réactive sera effectuée à l'extérieur de cette zone. Avec ce système, on dispose immédiatement des routes dans le voisinage proche, et lorsque la recherche doit être étendue plus loin, elle sera optimisée. Un nœud qui reçoit un paquet de recherches de routes réactives va tout de suite savoir si la destination est dans son propre voisinage. Si c'est le cas, il va pouvoir répondre, sinon il va propager de manière optimisée la demande dans sa zone proactive. Selon le type de trafic et des routes demandées, ce type de protocole hybride peut cependant combiner les inconvénients des deux méthodes : échanger des paquets de contrôle réguliers et l'inondation de l'ensemble du réseau pour trouver une route vers un nœud éloigné. Comme exemple de protocoles réactifs, on trouve ZRP, CEDAR, etc.

2.3.3.1 ZRP (*Zone Routing Protocol*)

ZRP utilise un protocole réactif au niveau local, c'est-à-dire avec des voisins situés à une distance inférieure à k sauts et un protocole proactif pour le routage entre groupes, appelés *routing zone* [Haas (1997)]. Pour l'opération de recherche d'une route, chaque nœud sont comment rejoindre les autres nœuds dans son groupe. Si le mobile a besoin de contacter un nœud à l'extérieur de cette zone, il utilise le protocole IERP (*Interzone Routing Protocol*) qui envoie une demande de route aux voisins situés à la périphérie du groupe. Ces derniers vérifient alors si le destinataire est dans leur groupe et lui

transmettent le message si celui-ci est présent. Dans le cas contraire, l'algorithme est réitéré : chacun de ces nœuds procède à une demande de route vers les nœuds en périphérie de son groupe. Une fois le chemin trouvé, le nœud destinataire renvoie un message point-à-point vers la source.

En combinant les avantages des mécanismes de routage proactifs et réactifs, le protocole ZRP réduit l'*overhead* de contrôle comparé au mécanisme de diffusion de *RouteRequest* utilisé dans les approches à la demande et à la diffusion périodique de l'information de routage dans les approches proactives. Cependant, et en l'absence d'une demande de contrôle, ZRP produit un *overhead* de contrôle supérieur à CEDAR. Ceci peut avoir lieu du fait que les nœuds peuvent se chevaucher dans des zones de routage. La demande de contrôle doit s'assurer que les messages *RouteRequests* redondants ou dupliqués ne sont pas envoyés. Aussi, la décision pour déterminer le rayon de la zone a un impact significatif sur la performance du protocole.

2.3.3.2 CEDAR (*Core Extraction Distributed Ad hoc Routing*)

CEDAR est un protocole de routage réactif avec qualité de service basé sur une élection dynamique d'un cœur de réseau stable [Sinha *et al.* (1999)]. Des informations sur les liens stables disposant d'une grande bande passante sont propagées entre les nœuds du cœur.

Ce protocole est basé sur trois composantes essentielles :

- **Extraction d'un cœur du réseau :** un ensemble de nœuds est choisi pour calculer les routes et maintenir l'état des liens du réseau. L'avantage d'une telle approche est qu'avec un ensemble réduit de nœuds les échanges d'information d'état et de route seront minimisés, évitant ainsi plus de messages circulant dans le réseau. En outre, lors d'un changement de route, seuls les nœuds du cœur serviront au calcul.

- **Propagation d'état de lien :** le routage avec qualité de service est réalisé grâce à la propagation des informations sur les liens stables avec une grande bande passante.
- **Calcul de route :** celui-ci est basé sur la découverte et l'établissement d'un plus court chemin vers la destination satisfaisant la bande passante demandée. Des routes de «secours» sont utilisées lors de la reconstruction de la route principale, quand cette dernière est perdue. La reconstruction peut être locale (à l'endroit de la cassure), ou à l'initiative de la source.

Un des avantages de ce protocole est qu'il assure le routage avec QoS au moyen des nœuds du noyau. La diffusion du noyau fournit un mécanisme fiable pour établir un chemin qui satisfait la QoS. Cependant, la détermination d'un chemin est réalisée par les nœuds du noyau seulement, le mouvement de ces nœuds affecte la performance du protocole. De plus, la mise à jour des nœuds du noyau peut causer un *overhead* de contrôle assez important.

2.4 Configuration et adressage

Dans les réseaux fixes, la configuration se fait manuellement ou en utilisant un serveur DHCP[11] (*Dynamic Host Configuration Protocol*) qui alloue des adresses aux machines qui joignent le réseau. Dans les réseaux ad hoc, et en absence d'une entité fixe, il est clair que pour assigner une adresse à un mobile qui souhaite joindre le réseau, les autres mobiles doivent connaître cette assignation et l'approuver afin d'éviter la duplication d'adresse. Dans la suite, nous allons explorer les principales propositions présentées dans la littérature et qui s'intéresse à la configuration et à l'adressage dans les réseaux *ad hoc*.

---

[11] DHCP : IETF RFC 2131

2.4.1 DAD

La méthode de détection d'adresse duelle DAD (*Dual Address Detection*) a été proposée par le groupe de travail *Zeroconf* de l'IETF [Perkins *et al.* (2001)]. Avec cette méthode, un mobile qui souhaite rejoindre un réseau *ad hoc* sélectionne une adresse dans une gamme prédéfinie puis il diffuse un message RREQ (*Route Request*) pour cette adresse dans tous le réseau. Si aucun message RREP (*Route Response*) n'est reçu au cours d'un certain temps, il essaye encore jusqu'au temps de RREQ RETRIES pour tenir compte du fait que le message reçu a été perdu le long de la route. Dans ce cas, un problème peut avoir lieu. En effet, lorsqu'un nœud attend une réponse, il doit d'abord avoir une adresse. Il est alors préférable durant la phase d'attribution d'adresse permanente d'affecter une adresse obtenue aléatoirement à partir d'un intervalle d'adresse réservé. Quand une adresse de cet intervalle n'est pas encore choisie comme adresse permanente, et que la plage des adresses temporaires est grande, la probabilité que deux mobiles aient la même adresse durant la phase d'attribution d'adresse permanentes est assez petite.

2.4.2 ANANAS

Dans l'architecture ANANAS (*A New Ad hoc Network Architectural Scheme*), les auteurs proposent de séparer les niveaux 2, 2.5 (niveau *ad hoc*) et le niveau 3 (IP) [Chelius et Fleury (2002)]. Cette approche consiste alors à utiliser deux systèmes d'adressage indépendant : un système au niveau *ad hoc* et l'autre au niveau IP. Ceci nécessite à l'introduction d'une interface virtuelle. Elle est utilisée pour effectuer le lien entre les couches d'adressage *ad hoc* et IP. Avec cette architecture, l'adressage IP n'a pas besoin de configuration spécifique étant donné que le niveau ad hoc est transparent pour le niveau IP. Le niveau IP voit un bus Ethernet auquel tous les nœuds du réseau sont attachés. Puisque le niveau *ad hoc* est vu comme une couche Ethernet simple, alors il supporte pleinement les services IP. Plusieurs réseaux *ad hoc* peuvent être alors considérés par une même couche IP et ils seront vus comme des liens Ethernet

différents. Ainsi, le routage entre ces différents réseaux sera purement IP. La mobilité entre les réseaux *ad hoc* est donc réalisée par le protocole Mobile IP.

2.4.3 DDHCP

Le protocole DDHCP (*Distributed Dynamic Host Configuration Protocol*) est une solution plus complète qui prend en considération les pertes de messages pendant la phase d'attribution des adresses dues à la nature des liens physiques entre les mobiles [Nesgari et Prakash (2002)]. En effet, les auteurs proposent un mécanisme pour renvoyer des adresses libérées à un ensemble d'adresses disponible pour la réutilisation. En d'autres termes, un nœud mobile souhaitant joindre le réseau essaye d'entrer en contact avec un nœud déjà dans le réseau et lui demande d'effectuer l'attribution d'adresse en son nom. Le nœud se joignant est connu comme le demandeur et le nœud effectuant l'attribution d'adresse est connu comme initiateur. Étant donné que l'initiateur appartient au réseau, il peut être atteint par les autres nœuds déjà dans le réseau et peut recevoir ainsi toutes les réponses envoyées. L'initiateur est donc une sorte de procuration pour le demandeur jusqu'à ce qu'il lui assigne une adresse IP permanente et devienne lui-même un nœud du réseau *ad hoc*.

2.4.4 ACDAD

Le mécanisme de configuration et de détection d'adresse dupliquée ACDAD (*an Advanced Configuration and Duplicate Address Detection*) a été proposé pour configurer et optimiser la détection de conflit pour le protocole OLSR [Adjih *et al.* (2005a)]. Ce mécanisme est basé sur l'algorithme DAD, un nouveau message de contrôle MAD (*Mulitple Address Declaration*) est défini et il est utilisé par l'algorithme DAD. Ce mécanisme comprend trois étapes principales :

- **assignation d'adresse** : un nœud sélectionne une adresse IP pour rejoindre un réseau *ad hoc*,

- **détection d'adresse dupliquée** : chaque nœud vérifie qu'il n'y a pas un autre nœud qui utilise la même adresse IP,
- **résolution de conflit** : quand un nœud détecte qu'il existe un autre nœud utilise la même adresse, il va sélectionner une autre adresse.

L'assignation d'adresse se fait de manière simple, par le nœud lui-même sans utiliser un message spéciale. Le nœud choisit une adresse aléatoire. La détection de duplication est basée sur le message de contrôle MAD. Ce message est envoyé par le nœud et inclut un identificateur (ID) et l'adresse choisie. Il sera diffusé de façon périodique dans tous le réseau par les MPR, en assumant que chaque nœud possède un ID unique. Un conflit survient entre deux nœuds quand un nœud reçoit un message MAD incluant sa propre adresse avec un ID différent du sien. Ce nœud déduit que le message MAD n'est pas son propre message et il a été envoyé par un autre nœud qui utilise la même adresse. La règle dans ACDAD pour résoudre un conflit d'adresse est que le nœud ayant un ID plus petit change son adresse. Puisque le nœud a déjà reçu les messages MAD de tous les nœuds du réseau, il va choisir alors une nouvelle adresse qui n'est pas assignée.

2.5 Contrôle de la topologie dans les réseaux *ad hoc*

Il existe principalement trois différentes approches pour le contrôle de la topologie dans les réseaux *ad hoc* : contrôle basé sur la puissance de transmission, contrôle basé sur la formation des clusters et contrôle basé sur la formation d'une dorsale. Dans cette section nous détaillerons chacune de ces approches tout en présentant les principaux algorithmes développés pour chaque approche.

2.5.1 Contrôle basé sur la puissance de transmission

Dans ces approches, on assigne à chaque nœud sa puissance de transmission comme paramètre afin que la topologie du réseau ait certaines propriétés de connectivité et que

la consommation des nœuds soit optimisée. Il existe deux types de contrôle de la topologie :

- les algorithmes centralisés où on suppose que toutes les positions sont connues par une entité centralisée qui détermine leur puissance de transmission. Toutefois, ces algorithmes ont un problème du passage à l'échelle quand le nombre de nœuds augmente,
- les algorithmes distribués quant à eux sont *scalables* et adaptés à la mobilité des nœuds, vu que seules des informations locales suffisent pour calculer la puissance appropriée.

Hou et Li ont étudié le rapport entre la portée de transmission et le débit dans un réseau *ad hoc* [Hou et Li (1986)]. Un modèle analytique a été proposé pour permettre à chaque nœud d'ajuster sa puissance de transmission, de réduire l'interférence et d'ainsi maximiser le débit. Dans [Hu (1993)], l'auteur a développé un algorithme distribué pour que chaque nœud ajuste sa puissance de transmission et construise une topologie fiable avec un haut débit. La minimisation de l'énergie n'a pas été étudiée dans ces deux travaux.

Ramanathan *et al.* ont proposé deux algorithmes centralisés dont la topologie induite avec un critère d'optimisation MINMAX pour réduire au minimum la puissance maximale utilisée par n'importe quel nœud [Ramanathan et Rosales-Hain (2000)]. En plus, deux heuristiques distribuées ont été proposées à savoir LINT (*Local Information No Topology*) et LILT (*Local Information Link-state Topology*) pour ajuster de manière adaptative la puissance de transmission des nœuds afin de garantir une topologie connectée quand celle-ci change. Mais, ni LINT ni LILT ne garantissent la connectivité du réseau. Dans [Lloyd *et al.* (2002)], une généralisation a été faite dans le but de

démontrer que : si la propriété désirée de la topologie est « monotone »[12], et si l'objectif d'optimisation est MINMAX, alors la puissance peut être calculée dans un temps polynomial.

Une méthode distribuée *cone-based* a été développée dans [Wattenhofer *et al.* (2001)]. Chaque noeud augmente graduellement sa puissance de transmission jusqu'à ce qu'il trouve un noeud voisin dans chaque direction, formant ainsi un cône. Par conséquent, la connectivité globale est garantie avec la puissance minimum pour chaque noeud. Huang *et al.* ont étendu ce travail dans le cas où des antennes directionnelles sont utilisées [Huang *et al.* (2002)]. Dans [Marsan *et al.* (2002)], les auteurs ont présenté une méthode pour optimiser la topologie d'un réseau *Bluetooth*, qui vise à réduire la charge de trafic maximum des noeuds, réduisant de ce fait au minimum la puissance maximale des noeuds.

Dans [Singh *et al.* (1998)], les auteurs ont étudié plusieurs métriques pour un routage efficace, comme la minimisation d'énergie consommée par paquet, la réduction de la variation du niveau de consommation d'énergie des nœuds, la réduction du coût par paquet, etc. Wieselthier *et al.* ont analysé le problème d'ajustement de la puissance de transmission de chaque nœud de façon à ce que le coût exprimé en termes d'énergie de l'arbre de *broadcast/multicast* soit minimal [Wieselthier *et al.* (2000)].

2.5.2 Contrôle basé sur la formation des *clusters*

Baker et Ephremides étaient les premiers à introduire la notion de *clustering* dans les réseaux *ad hoc* en proposant une architecture appelée LCA (*Linked Cluster Architecture*) [Baker et Ephremides (1981)]. Ils ont démontré sa capacité à s'adapter au changement de la topologie. Dans [Krishna *et al.* (1997)], les auteurs proposent la

---

[12] Ici, une propriété est monotone si elle est toujours la même lorsqu'un nœud augmente sa puissance de transmission.

formation de *clusters* recouvrants. L'algorithme est exécuté par chaque nouveau nœud après la réception d'une liste des *clusters* actuels. Cet algorithme permet de choisir tous les *clusters* possibles auxquels être connecté, et dont la cardinalité est maximale. Ensuite, un algorithme de suppression est appliqué afin d'éliminer les *clusters* inutiles. Enfin, la liste des *clusters* sera mise à jour et diffusée.

Dans [Gerla *et al.* (2000)], la clustérisation et la diffusion ont été combinés pour diminuer l'*overhead* de la communication. Il n'existe pas de paquet de contrôle explicite pour la construction et la maintenance du *cluster*. Chaque hôte décide de son statut en écoutant le trafic existant. On ajoute à tout paquet qui traverse la couche MAC deux bits qui indiquent le statut de l'expéditeur. Un hôte ordinaire devient alors un *clusterhead* s'il n'existe aucun voisin *clusterhead* actif au moment où il envoie un paquet. Chaque hôte maintient une liste incluant une entrée pour chaque voisin *clusterhead*. Une hôte qui reçoit un paquet de ce dernier décide de son propre statut (*gateway* ou ordinaire) en observant sa liste de voisins *clusterheads*. Les auteurs mentionnent que la clustérisation passive est meilleure par rapport à la clustérisation conventionnelle au niveau de la stabilité et de la robustesse dans un environnement *ad hoc*.

Kawadia *et al.* ont proposé une méthode de clustérisation pour un routage dans les réseaux non homogènes [Kawadia *et al.* (2003)], où les nœuds sont distribués dans des clusters. Le but était de choisir un niveau de puissance de transmission adéquate, de telle façon que les niveaux bas seront utilisés pour des communications à l'intérieur du *cluster* et les niveaux élevées pour celles inter-*clusters*.

2.5.2.1 Algorithme de clustérisation

Un algorithme de clustérisation est basé sur les étapes suivantes :

- **Élection des *clusterheads* :** le réseau est ainsi divisé en plusieurs clusters, la phase d'élection ou de *cluster setup phase* utilise des heuristiques comme le plus grand/plus petit ID dans le voisinage, le degré de connectivité, la zone géographique, la puissance de transmission ou la vitesse de déplacement (exemple : utiliser les nœuds les moins mobiles), ou bien en utilisant un poids pour chaque nœud qui représente une combinaison des derniers attributs.
- **Communication entre les *clusterheads* :** dans un *cluster*, chaque paire de nœuds est à deux sauts de distance entre eux. De plus, comme les *clusterheads* ne sont pas directement reliés, des nœuds passerelles sont aussi élus et utilisés pour les communications entre *clusterheads*.
- **Maintenance des *clusterheads* :** dans le but de s'adapter aux changements de la topologie fréquents dans le réseau, une mise à jour des *clusterheads* élus est dynamiquement réalisée.

2.5.2.2 Clustérisation hiérarchique

La clustérisation hiérarchique adopte le même principe en assignant dynamiquement des *Clusters* ID avec différents niveaux. Un *cluster* peut dynamiquement se fusionner ou s'éclater en fonction du nombre de nœuds dans un *cluster*. Cependant, étant donné la difficulté de prévoir le temps nécessaire pour propager les messages de contrôle de clustérisation à travers les nœuds ainsi que le délai important de convergence de l'algorithme de clustérisation, cette approche dégrade rapidement les performances dans le réseau.

2.5.3 Contrôle basé sur la construction d'une dorsale

Une troisième approche pour effectuer le contrôle de la topologie est de sélectionner un sous-ensemble de nœuds qui forment le réseau de façon à ce que ce sous-ensemble forme une dorsale pour tous les nœuds. Nous avons intérêt à ce que cet ensemble soit de taille minimale étant donné qu'il y aura un *overhead* supplémentaire qui est proportionnel à la taille de la dorsale. De plus, le rôle de cette dorsale nécessite que les nœuds qui forment cette dorsale soient connectés. De ce fait, la détermination de l'ensemble de domination de taille minimale (*Minimum Connected Dominating Set*, MCDS) dans un graphe présente un bon candidat pour construire la dorsale. Cependant la détermination de l'ensemble MCDS dans un graphe est un problème *NP-complet* [Das et Bharghavan (1997)]. Mais il existe plusieurs propositions qui ont pour objet de trouver une approximation de cet ensemble. A titre d'exemple, nous citons les travaux de [Das et Bharghavan (1997)] [Guha et Khuller (1998)] [Sivakumar *et al.* (1998)] [Sinha *et al.* (1999)] [Sinha *et al.* (2001)] [Wu et Li (1999)] [Wan *et al.* (2002)] [Butenko *et al.* (2003)].

Guha *et al.* ont proposé deux algorithmes pour calculer l'ensemble de domination connexe (CDS) [Guha et Khuller (1998)]. Dans le premier algorithme, tous les nœuds sont initialement colorés en blanc. Par la suite, le nœud ayant un degré maximum est coloré en noir et tous ses voisins sont colorés en gris. À chaque itération, un nœud gris ou une paire de nœuds (un nœud gris et un de ses voisins blancs) qui a un nombre maximum de nœuds voisins blancs (chaque nœud blanc est considéré au moins une fois) sont colorés en noir et tous les voisins blancs sont colorés en gris. Le taux de performance de cet algorithme est de $2.\ln(\Delta_{max})+2$, avec Δ_{max} qui représente le degré maximum. Le deuxième algorithme est basé sur l'idée de former l'ensemble de domination connexe. Spécialement à chaque étape, un nœud est coloré en noir s'il connecte un nombre maximum de composants (au moins deux composants). Un composant est ou bien un nœud blanc ou un graphe induit par un ensemble de nœuds noirs. Quand un nœud est coloré en noir, tous ses voisins sont colorés en gris. Cette

étape prend fin quand le graphe ne contient aucun nœud blanc. Dans la seconde étape, deux nœuds gris seront choisis pour connecter deux composants. Cet algorithme a un taux de performance de $\ln(\Delta)+3$, avec Δ qui représente le degré moyen. Dans les deux algorithmes, tous les noeuds colorés en noir forment l'ensemble de domination connexe (CDS).

Dans [Sivakumar *et al.* (1998)], on a présenté deux implémentations distribuées pour ces deux algorithmes proposés par [Guha et Khuller (1998)]. Ces implémentations souffrent du nombre très élevé de message et du temps de complexité. [Sinha *et al.* (1999)] et [Sinha *et al.* (2001)] déterminent un cœur (*core*) qui forme l'ensemble de domination. Chaque nœud choisit son propre dominant ayant le plus haut degré et qui a un nombre maximum de dominés en considérant les voisins à un saut. Il est possible que le nœud choisisse lui-même son dominé. Chaque hôte appartenant au cœur envoie des paquets, en *unicast*, à tous ses voisins à trois sauts. Dans [Sinha *et al.* (2001)], les résultats de simulation montrent que les protocoles de routage DSR et AODV performent mieux quand la structure de dorsale virtuelle qui est employée.

Dans [Wu et Li (1999)], on a présenté un algorithme simple pour calculer l'ensemble CDS. Une version plus étendue de cet algorithme est présentée dans [Wu et Li (2001)]. Chaque hôte a besoin de connaître l'information de ses voisins à deux sauts. Une hôte se déclare comme dominant s'il existe deux voisins qui n'ont pas de chemin direct entre eux. Un dominant u change son statut à dominé si un de ses voisins ayant un *id* plus grand est aussi un dominant et couvre tous les dominés de u, ou si deux de ses voisins connexes v et w possédant des *ids* supérieurs à u et qui sont aussi dominants peuvent couvrir tous les dominés de u.

Dans [Alzoubi *et al.* (2002)], les auteurs ont proposé un algorithme ayant une efficacité de 8-*approximatif* demandant la connaissance d'information des voisins à un saut. Dans cet algorithme, un arbre de recouvrement minimum est arbitrairement construit dans une première étape. Chaque nœud connaît son propre niveau ainsi que les niveaux de ses

voisins à un saut. La racine déclare le statut de ses dominants en diffusant un message DOMINATOR. Chaque hôte recevant le message DOMINATOR déclare son statut comme dominé. Quand un hôte détecte que tous ses voisins de niveau inférieur et que tous les voisins du même niveau ayant un ID inférieur ont le statut de dominé, il se déclare comme un dominant. Un dominé change son statut à dominant s'il détecte qu'un de ses fils est un dominant. Cette règle d'assignation de statut assure que chaque arbre de recouvrement peut générer un ensemble de domination connexe de taille, dans la plupart du temps, 8.OPT, avec OPT étant la taille de l'ensemble de domination connexe optimale.

Butenko *et al.* ont proposé un algorithme (B-CDS) pour déterminer l'ensemble MCDS en deux versions : une version centralisée et une version distribuée [Butenko *et al.* (2003)]. Dans la version centralisée, un nœud est sélectionné à chaque itération, en utilisant une méthode gloutonne, pour être ajouté à la solution finale ou être éliminé de l'ensemble courant tout en vérifiant que la solution finale soit toujours connectée. Dans la version distribuée, on commence par élire des chefs de groupes (*leader*). Par la suite, un leader va exécuter une procédure *self-removal* qui va vérifier si en éliminant un nœud u le sous-graphe induit reste toujours connecté. Si c'est le cas une procédure *remove-vertex* est exécutée, sinon le nœud u doit être présent dans la solution finale et une procédure *fix-vertex* est alors exécutée.

Dans la suite de cette section, nous détaillerons les algorithmes WCDS, CDS-based et B-CDS de [Wu et Li (2001)], [Guha et Khuller (1998)] et [Butenko *et al.* (2003)] respectivement. Nous utiliserons ces trois algorithmes pour comparer et évaluer notre algorithme proposé dans le chapitre 3 pour déterminer l'ensemble MCDS.

Nous détaillerons également un algorithme qui permet de déterminer l'ensemble des relais multipoint MPR (MultiPoint Relay), nous allons utiliser cet algorithme pour comparer différentes techniques de diffusion (chapitre 4).

2.5.3.1 Description de l'algorithme WCDS

L'algorithme WCDS proposé par Wu et Li est basé sur le principe de marquage. Cet algorithme a été proposé initialement dans [Wu et Li (1999)] pour le cas de graphes non-directs en utilisant la notion de l'ensemble dominant uniquement. Cet algorithme a été étendu pour le cas de graphes directs dans [Wu et Li (2001)] en introduisant la notion de l'ensemble absorbant. Chaque nœud est marqué (devient un *gateway*) s'il possède deux voisins non-connectés. Il a été démontré que la collection de nœuds aboutit à l'objectif global (un ensemble de nœuds connectés CDS de taille minimale). Basé sur le principe de marquage, les nœuds *u* et *w* dans la figure 15 sont marqués et forment un ensemble dominant dans le réseau. L'ensemble CDS dérivé de cet ensemble est déduit en appliquant deux règles d'élagage : dans la règle n°1, un nœud peut devenir non marqué si l'ensemble de ses voisins est couvert par un autre nœud marqué; celui-ci, si tous ses voisins sont connectés ensemble via un autre *gateway*, peut abandonner ses responsabilités de *gateway*. Dans la figure 16, un seul des nœuds *u* ou *w* peut enlever le marquage (mais pas les deux). D'après la règle n°2, un nœud marqué peut devenir non marqué si ses voisins sont couverts par deux autres nœuds marqués qui sont directement connectés.

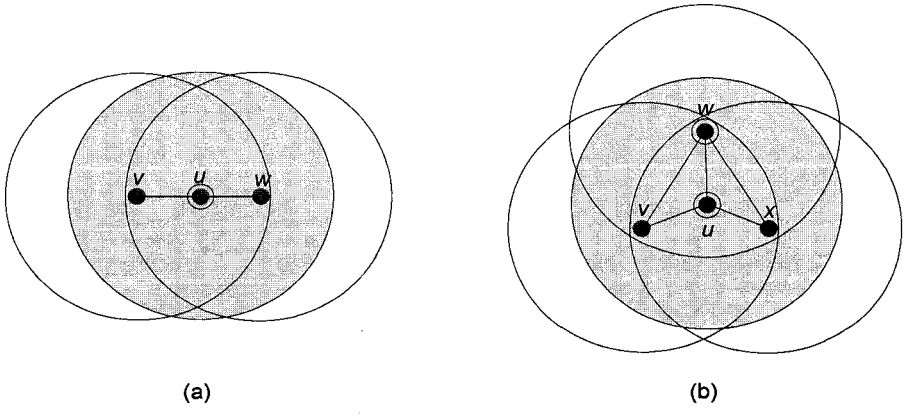


Figure 15 Principe de marquage dans WCDS

Plus formellement, cet algorithme peut se résumer de la façon suivante :

Soit $V' \subset V$ l'ensemble des dominants de G si chaque nœud $v \in V \backslash V'$ est dominé par au moins un nœud $u \in V'$. Aussi, l'ensemble $V \backslash V'$ est appelé l'ensemble des absorbants si pour chaque nœud $u \in V \backslash V'$, il existe un nœud $v \in V'$ qui est un absorbant de u.

L'ensemble des nœuds voisins dominant $N_d(u)$ pour le nœud u est défini par $\{w : (w,u) \in E\}$. L'ensemble des nœuds voisins absorbant $N_a(u)$ pour le nœud u est défini par $\{v : (u,v) \in E\}$. $N(u) = N_d(u) \cup N_a(u)$ représente l'ensemble de voisins pour le nœud u.

À chaque nœud $v \in V$, on assigne un identificateur $id(v)$. Tous les nœuds sont initialement marqués en F (non-marqué).

Règle n°1

> Considérons deux nœuds u et v de G' (sous-graphe induit de V'). Si $N_d(u) \setminus \{v\} \subseteq N_d(v)$ et $N_a(u) \setminus \{v\} \subseteq N_a(v)$ dans G et $id(u) < id(v)$, alors changer le marquage de u à F.

Règle n°2

> Supposons deux nœuds v et w sont bidirectionnellement connectés dans G'. Si $N_d(u) \setminus \{v, w\} \subseteq N_d(v) \cup N_d(w)$ et $N_a(u) \setminus \{v, w\} \subseteq N_a(v) \cup N_a(w)$ dans G et $id(u) = min\{id(u), id(v), id(w)\}$, alors changer le marquage de u à F.

Toutefois, l'utilisation de ces deux règles n'aboutit pas à une solution minimale, et en voici deux contre-exemples :

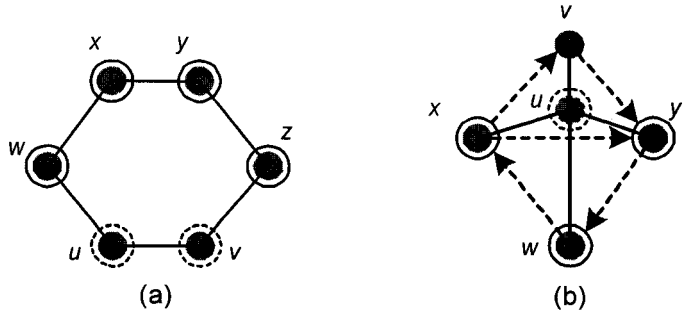


Figure 16 Limitations des règles n°1 et n°2

Dans la figure 16-a, les nœuds *u* et *v* peuvent être éliminés; ils sont couverts par les nœuds *w* et *z*. Dans la figure 16-b, le nœud *u* peut être éliminé car il est couvert par les nœuds *w*, *x* et *y*. Notons que *x* et *y* ne sont pas directement bidirectionnellement connectés; ils peuvent se rejoindre via le nœud *w*. Cependant, aucun de ces nœuds ne peut être éliminé en appliquant les règles n°1 et n°2, car ils ne peuvent pas être couverts par un ou deux nœuds directement bi-directionnellement connectés.

2.5.3.2 Description de l'algorithme CDS-based

L'algorithme CDS-based a été proposé pour la première fois par Guha et Kuller dans [Guha et Khuller (1998)]. Il a été amélioré par Chen et Liestman dans [Chen et Liestman (2003)]. Étant donné un graphe $G = (V, E)$, on associe une couleur (blanche, grise ou noire) pour chaque nœud. Tous les nœuds sont initialement de couleur blanche et changent de couleur au fur et à mesure que l'algorithme progresse. L'algorithme est essentiellement un processus itératif pour le choix des nœuds noirs parmi les nœuds blancs et gris. Quand un nœud est noir, tous ses voisins blancs vont changer de couleurs et devenir gris. L'ensemble des nœuds noirs forme l'ensemble CDS. Le terme morceau (*piece*) est utilisé pour définir une sous structure du graphe. Un morceau blanc est tout simplement un nœud blanc. Un morceau noir contient un maximum de nœuds noirs qui forment un graphe faiblement connecté et un ensemble de nœuds gris voisins à au moins un nœud noir. La figure 17 illustre un morceau blanc et un morceau noir. Les morceaux

sont définis par un trait discontinu. Les nœuds 4, 5, 6 et 7 sont des morceaux blancs. Les autres nœuds font partie de deux morceaux noirs, un contient le nœud 1 et l'autre contient les deux nœuds noirs 2 et 3.

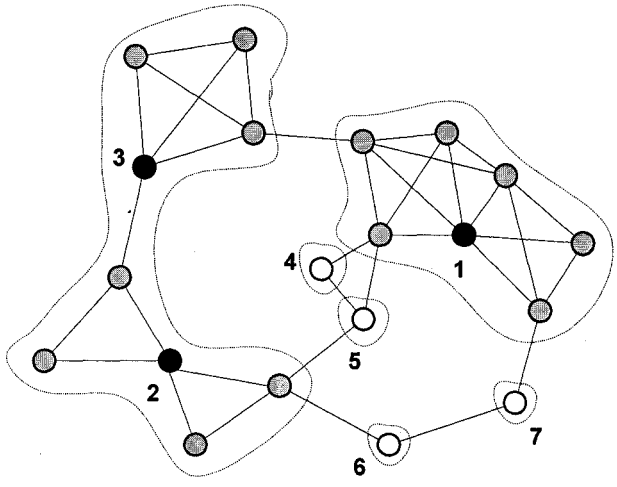


Figure 17 Illustration de morceaux blancs et noirs

À chaque itération, l'algorithme choisit un nœud blanc ou gris pour devenir noir. Le nœud est choisi de manière gloutonne (*greedy*) afin de réduire le nombre de structures autant que possible jusqu'à ce qu'il ne reste une seule structure. L'exemple illustré à la figure 17 montre l'exécution de l'algorithme à sa troisième étape. À l'étape suivante, si le nœud 5 est choisi pour devenir noir, le nœud 4 sera alors fusionné dans la structure, ce qui réduit le nombre de structure à 3. Choisir un nœud (autre que le nœud 5) pour devenir noir, ceci entraînera au plus une fusion de trois structures. C'est pourquoi le nœud 5 sera choisi dans la prochaine étape.

La taille de l'ensemble CDS-based construit en appliquant l'algorithme décrit dans cette section est de l'ordre de $(\ln \Delta_{max}+1).|OPT|$, Δ_{max} est le degré maximum dans le graphe et *OPT* représente la taille minimale de l'ensemble MCDS.

2.5.3.3 Description de l'algorithme B-CDS

Comme mentionné ci-haut, les auteurs dans [Butenko *et al.* (2003)] ont proposé deux versions pour cet algorithme : une version centralisée et une version distribuée.

a- Version centralisée

Dans l'étape d'initialisation, l'ensemble de tous les nœuds du graphe est choisi comme l'ensemble CDS. Deux types de nœuds ont été définis : (i) un nœud fixe est un nœud qui ne peut pas être éliminé de l'ensemble CDS, son élimination engendrait une solution non réalisable ; (ii) un nœud non-fixe qui peut être éliminé, son élimination n'engendrait pas un sous-graphe induit non-connexe. A chaque itération de l'algorithme un nœud est choisi et il sera considéré soit fixe, et il fera partir de la solution finale, soit il sera éliminé de la solution réalisable. Plus formellement l'algorithme s'écrit :

```
/* D est l'ensemble CDS, F est l'ensemble des nœuds fixes*/
D ←V F←∅                                   /* initialisation */
tant que D\F ≠∅ fait
        u ← argmin {δ(v)|v∈D\F}            /* δ(v) représente le nombre des nœuds
                                              Si G[D\{u}] n'est pas connexe alors
                F←F∪{u}                       adjacents pour le nœud v */
        Sinon
                D←D\{u}
                pour tout s∈ D∩N(u) fait    /* N(v) représente l'ensemble des nœuds
                        δ(s)←δ(s) – 1              adjacents pour le nœud v δ(v) = |N(v)|*/
                et pour tout
                if N(u) ∩ F = ∅ alors
                        w←argmax {δ(v)|v∈N(u)}
                        F←F∪{w}
                end
        end
end
retourner D
```

Le temps d'exécution de la version centralisée de cet algorithme est de l'ordre de $O(nm)$ [Butenko *et al.* (2003)].

b- Version distribuée

Dans la version distribuée, la première étape consiste à élire des chefs de groupes (*leaders*). Pour cela, les auteurs utilisent l'algorithme proposé dans [Malpani*et al.* (2000)]. Cet algorithme d'élection s'exécute dans un temps de l'ordre de $O(n \log n)$. Chaque *leaders* élu va alors exécuter une procédure appelée *self-removal*. Cette procédure consiste à vérifier si l'élimination d'un nœud u mènera à un sous graphe induit qui n'est pas connexe. Dans ce cas, une procédure *fix-vertex* est alors exécutée et ce nœud doit être dans la solution finale. Dans le cas contraire, la procédure *Remove-vertex* est exécutée.

Dans cet algorithme, on dit qu'un lien (u, v) est actif pour le nœud v si le nœud u n'a pas été éliminé de l'ensemble CDS auparavant. De plus, les messages sont envoyés uniquement en utilisant des liens actifs, puisque les autres liens ne mènent pas à des nœuds qui font partie de l'ensemble CDS.

Dans la procédure *fix-vertex* différentes étapes sont exécutées : la première étape consiste à diffuser un message NEWDOM, par le nœud u, pour informer l'ensemble des nœuds qu'il est devenu dominant. La deuxième étape est de vérifier si ce nœud u pourra élimer d'autres nœuds. Ceci est réalisé en se basant sur le degré de chaque nœud voisin : le voisin qui possède le degré le plus petit sera considéré en premier et il reçoit un message TRY-DISCONNECT.

La procédure *Remove-vertex* est exécutée uniquement si le nœud v peut être éliminé. Il commence alors par envoyer un message DISCONNECTED à tous les voisins. Par la suite, il sélectionne le nœud qui va être un dominant pour v. S'il existe un dominant au voisinage, alors il sera utilisé. Sinon, un nœud ayant le plus grand nombre de nœuds

adjacents $N(v)$ sera choisi comme nouveau dominant. Finalement, un message SET-DISCONNECT est envoyé au nœud choisi.

Le temps d'exécution de la version distribuée de cet algorithme est l'ordre de $O(nlog^3n)$ [Butenko *et al.* (2003)].

2.5.3.4 Description de l'algorithme MPR

Les relais multipoint MPR (*Multipoint Relay*) sont une technique pour déterminer l'ensemble de domination connexe [Qayyum *et al.* (2002)] [Lim et Kim (2000)] [Calisnescu *et al.* (2001)]. Dans cette technique, on sélectionne une source. Partant de cette source, chaque hôte qui a besoin de diffuser un paquet, choisit un ensemble de relais multipoint de taille minimale parmi ses propres voisins à un saut de façon à ce que cet ensemble de relais couvre tous les hôtes qui sont loin de deux sauts. Chaque hôte diffuse son propre ensemble relais multipoint à tous ses voisins d'un saut. Quand un hôte v reçoit un paquet de u, et v est dans l'ensemble MPR de u, v détermine son propre MPR et retransmet le paquet. Dans un MRS, seuls les voisins de u sont responsables de retransmettre le paquet. L'union de tous les MPRs forme l'ensemble transitaire « *forwarding* ». Qayyum *et al.* ont démontré que le problème de trouver l'ensemble de relais multipoint de taille minimale est un problème *NP-Complet* [Qayyum *et al.* (2002)]. Dans [Jacquet et Claussen (2003)], le protocole OLSR utilise l'ensemble de relais multipoint et optimise ainsi la diffusion des paquets d'information dans le réseau. De plus, [Calisnescu *et al.* (2001)] ont présenté un autre algorithme d'approximation avec une performance fixe de rapport 6. Tous ces algorithmes ont besoin de l'information de voisinage à deux sauts. Une mise à jour périodique de l'information au voisinage consomme de la bande passante qui augmente la communication *overhead.* Plus formellement, l'algorithme de sélection des relais multipoint peut se résumer comme suit [Adjih *et al.* (2005)] :

Soit $N(i)$ l'ensemble des voisins directes de i, $N_2(i)$ l'ensemble des voisins à deux sauts et MPR(i) l'ensemble des relais multipoint de i :

- commencer par un ensemble de relais multipoint vide $MPR(i) = \phi$,
- choisir les nœuds de l'ensemble des voisins $N(i)$ qui sont les seuls ayant un lien avec un voisin du second niveau. Ajouter ces nœuds sélectionnés de $N(i)$ à l'ensemble $MPR(i)$ et éliminer tous les nœuds du second niveau couverts par ces derniers de l'ensemble $N_2(i)$,
- tant que $N_2(i) \neq \phi$, refaire,
    - calculer le degré de chaque nœud dans $N(i)$. Le degré pour un nœud est le nombre de voisins du second niveau couverts par celui-ci présent dans $N_2(i)$,
    - ajouter le nœud de $N(i)$, ayant le degré maximal à l'ensemble des relais multipoint $MPR(i)$, et enlever tous les nœuds du second niveau couverts par celui de $N_2(i)$.

2.6 Conclusion

Bien que d'autres protocoles de routage aient été proposés pour les réseaux *ad hoc*, un chapitre ne suffirait pas pour les présenter. Les principales approches ont été décrites et il est important de retenir qu'aucun des protocoles proposés n'a tous les avantages et la performance recherchée.

Les techniques de configuration et d'adressage (comme DHCP) utilisées dans les réseaux filaires ne sont plus applicables dans le contexte des réseaux *ad hoc.* D'autres solutions, qui tiennent compte des caractéristiques des réseaux *ad hoc*, ont été proposées

dans le but d'assurer la configuration et l'adressage. Dans ce chapitre, nous avons présenté les principales propositions.

Dans ce chapitre, nous avons présenté également les techniques proposées pour effectuer le contrôle de la topologie dans les réseaux *ad hoc*. Ce contrôle de la topologie est primordial étant donné l'absence d'une infrastructure (fixe ou mobile) dans les réseaux *ad hoc*. Ce contrôle permet en quelque sorte de fournir une information utile pour les protocoles de routage surtout quand la mobilité est introduite.

Dans notre recherche, nous nous intéressons particulièrement au contrôle basé sur la construction d'une dorsale et par la suite à la détermination de l'ensemble MCDS pour les raisons suivantes :

- le contrôle basé sur la puissance de transmission ne garantit pas la connectivité du réseau. De plus, nous considérons les réseaux *ad hoc* homogènes, les terminaux possédant le même type de source d'énergie et d'autonomie,
- le contrôle basé sur la formation des *clusters* nécessite une justification adéquate pour choisir les *clusterheads*. De plus, deux nœuds adjacents mais n'appartenant pas au même *cluster* doivent acheminer leurs trafics par une route qui passe par les *clusterhead*s au lieu de l'acheminer directement. Un autre problème survient lorsqu'un *clusterhead* change de position et n'est pas dans le centre du cluster. Dans ce cas les membres du groupe vont s'occuper de l'élection d'un nouveau *clusterhead* ce qui affecte la performance du protocole de routage.

Nous allons proposer un algorithme qui fournit une taille minimale de la dorsale dans un graphe UDG. Dans la littérature, les algorithmes WCDS et CDS-based sont les plus utilisés comme références pour évaluer d'autres algorithmes [Wu et Li (2001)] et [Guha et Khuller (1998)]. Ces deux algorithmes performent mieux que les autres au niveau du temps de calcul, de la complexité et de l'approximation de la solution finale. Dans le chapitre 4, nous allons proposer un nouveau algorithme pour déterminer une

approximation de l'ensemble MCDS et construire ainsi la dorsale. Nous utiliserons ces deux algorithmes pour évaluer et pour comparer la solution finale de notre algorithme.