Chapitre 3

Les protocoles de routage dans le réseaux ad-hoc

3.1 Introduction

Comme nous avons déjà vu, un réseau ad hoc est un ensemble de nœuds mobiles qui sont dynamiquement et arbitrairement éparpillés d'une manière où l'interconnexion entre les nœuds peut changer à tout moment. Dans la plupart des cas, l'unité destination ne se trouve pas obligatoirement dans la portée de l'unité source ce qui implique que l'échange des données entre deux nœuds quelconques, doit être effectué par des stations intermédiaires. La gestion de cet acheminement de données, ou routage, implique l'établissement d'une certaine architecture globale que l'on doit tenir compte de la mobilité des unités et de la versatilité du médium physique.

La stratégie (ou le protocole) de routage est utilisée dans le but de découvrir les chemins qui existent entre les nœuds. Le but principal d'une telle stratégie est l'établissement de routes qui soient correctes et efficaces entre une paire quelconque d'unités, ce qui assure l'échange des messages d'une manière continue.

Vu les limitations des réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante. Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être séparés en deux catégories, les protocoles pro-actifs et les protocoles réactifs. Les protocoles pro-actifs établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage, alors que les protocoles réactifs cherchent les routes à la demande. Certains travaux de recherche (comme dans [GER ??]), prennent en considération le facteur de localisation dans la classification des

protocoles, par conséquent les protocoles qui utilisent les informations de localisation (comme les protocoles LAR et DREAM) sont classés dans une classe indépendante.

Dans ce qui suit, nous allons présenter les protocoles les plus connus, proposés pour effectuer le routage dans les réseaux ad hoc. 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.

3.2 Les protocoles de routage pro-actifs

Les protocoles de routage pro-actifs pour les réseaux mobiles ad-hoc, sont basé sur la même philosophie des protocoles de routage utilisés dans les réseaux filaires conventionnels. Pour cela, nous allons examiner les deux principales méthodes utilisées dans le routage des réseaux filaires avant de présenter quelques protocoles de cette classe.

Les deux principales méthodes utilisées sont : la méthodes *Etat de Lien* ("*Link State*") [ISO 90, JOH 80, MOY 94] et la méthode du *Vecteur de Distance* ("*Distance Vector*") [HED 88, JOH 77, GUR 90, PAU 90, XER 81]. Les deux méthodes exigent une mise à jour périodique des données de routage qui doit être diffusée par les différents nœuds de routage du réseau. Les algorithmes de routage basés sur ces deux méthodes, utilisent la même technique qui est *la technique des plus courts chemins*, et permettent à un hôte donné, de trouver le prochain hôte pour atteindre la destination en utilisant le trajet le plus cours existant dans le réseau. Généralement le calcul du plus court chemin entre deux stations, est basé sur le nombre de nœuds (on dit aussi le nombre de *sauts*) que comportent les différents chemins qui existent entre les deux stations. Mais on peut aussi associer des coûts aux liens de communication ; ces coûts seront utilisés en appliquant une fonction de calcul qui est en générale linéaire. Un coût peut matérialiser le taux de l'utilisation d'un lien, les délais relatifs de communication ou un autre facteur qu'on veut le minimiser lors du routage des données.

Dans le protocole "Link State", chaque nœud de routage maintient sa propre vision de la topologie du réseau et qui inclut l'état de ses canaux de sortie. Pour que cette vision soit à jour, chaque nœud diffuse (par inondation) périodiquement l'état des liens de ses voisins à tous les nœuds du réseau. Cela est fait aussi quand il y a un changement d'état de liens. Un nœud qui reçoit les informations concernant l'état des liens, met à jour sa vision de la topologie du réseau et applique un algorithme de calcul des chemins optimaux afin de choisir le nœud suivant pour une destination donnée. Un exemple des algorithmes les plus connus appliqué dans le calcul des plus courts chemins, est celui de Dijkstra [BER 92]. Notons que le nœud de routage calcule la plus courte distance qui lui sépare d'une destination donnée, en se basant sur l'image complète du réseau formée des liens *les plus récents* de tous les nœuds de routage. Cela veut dire que pour qu'un nœud *p* puisse déterminer le nœud de routage suivant

pour une destination donnée, *p* doit recevoir les messages de *la dernière* mise à jour des liens, propagé par le réseau. Le protocole OSPF (*Open Shoretest Path First*) [MOY 94], est l'un des protocoles les plus populaires basé sur le principe "Etat de lien". Comme nous allons voir par la suite, l'algorithme "Distance Vector" de base a été adopté pour le routage dans les réseaux ad hoc sans fil, et cela en traitant chaque hôte mobile comme un nœud de routage [JOH 87, PER 94, NAC 87].

Dans l'approche de routage "Distance Vector", chaque nœud de routage diffuse à ses nœuds de routage voisins, sa vision des distances qui lui séparent de tous les hôtes du réseau. En se basant sur les informations reçus par tous ses voisins, chaque nœud de routage fait un certain calcul pour trouver le chemin le plus court vers n'importe quelle destination. Le processus de calcul se répète, s'il y a un changement de la distance minimale séparant deux nœuds, et cela jusqu'à ce que le réseau atteigne un état stable. Cette technique est basée sur l'algorithme distribué de Bellman-Ford (DBF) [BER 92].

L'algorithme DBF est basé sur l'utilisation des messages de mise à jour. Un message de mise à jour contient un vecteur d'une ou plusieurs entrées dont chaque entrée contient, au minimum, la distance vers une destination donnée. Le principe du DBF est utilisé par une grande partie des protocoles de routage des réseaux filaires. Un problème de performance majeur de cet algorithme est qu'il prend beaucoup de temps pour mettre à jour les tables de routage des hôtes, après une partition du réseau, des défaillances de nœuds, ou quand il y a un grand nombre de nœuds dans le réseau. D'autres problèmes du DBF sont dus à l'absence de coordination entre nœuds, dans les modifications des tables de routage qui peuvent être faites en se basant sur des données erronées (le problème de "*routing loops*"). En plus de cela, le DBF ne possède pas de mécanisme précis qui peut déterminer quand est ce que le réseau doit arrêter l'incrémentation de la distance qui correspond à une destination donnée, ce problème est appelé : "*counting to infinity*".

La circulation inutile des paquets de messages, qui peut arriver avec le DBF, est intolérable dans les réseaux mobiles ad hoc, caractérisés par une bande passante limitée et des ressources modestes. En plus de cela, la mobilité fréquente des nœuds met que la convergence du DBF prend beaucoup de temps, ce qui pénalise le routage dans de tels environnements. Dans ces dernières années, beaucoup d'efforts ont été faits dans le but de résoudre les problèmes du DBF, et de l'adapter dans le contexte des réseaux mobiles. Une de ces efforts est le protocole DSDV que nous allons voir par la suite.

Dans les algorithmes de routage basés sur le principe "Link State", le problème *counting to infinity* n'existe pas. En plus de cela, la convergence d'un nœud de routage, est moins lente par rapport au DBF, ce qui a fait que "Link State" est plus préféré et utilisé dans beaucoup de réseaux modernes, tel que Internet [MOY 94] et ATM [ATM 96]. Cependant, l'approche du "Link State" exige que chaque nœud, doive maintenir une version mise à jour de la topologie complète du réseau, ce qui nécessite un grand espace de stockage et implique une surcharge d'échange de paquets de contrôle dans le cas des réseaux dynamiques. En outre, aucun algorithme, implémenté en se basant sur le principe "Link State", n'a pu éliminer - totalement - la création des boucles temporaires de routage.

Les protocoles de routage pro-actifs rassemblent les idées des deux approches précédentes, et essaient de les adapter pour les environnements mobiles en essayant de réduire ou d'éliminer leurs limitations tout en prenant en considérations, les caractéristiques du nouvel environnement.

3.2.1 Le protocole de routage DSDV

L'algorithme de routage de Perkins appelé "Vecteur de Distance à Destination Dynamique Séquencée" ou DSDV (Dynamic Destination-Sequenced Distance-Vector) [PER 94] a été conçu spécialement pour les réseaux mobiles. Il est basé sur l'idée classique de l'algorithme distribué de Bellman-Ford (DBF : Distributed Bellman-Ford) en rajoutant quelques améliorations. Chaque station mobile maintient une table de routage qui contient :

- *Toutes les destinations possibles*.
- *Le nombre de nœud (ou de sauts) nécessaire pour atteindre la destination*.
- *Le numéro de séquences (SN : sequence number) qui correspond à un nœud destination*.

Pour chaque nœud *i*, le numéro de séquence (NS) de la destination *j*, est associé à chaque entrée de distance D^{i}_{jk}, pour chaque voisin *k*. Le NS est utilisé pour faire la distinction entre les anciennes et les nouvelles routes, ce qui évite la formation des boucles de routage [PER 94].

Afin de maintenir la consistance des tables de routage dans une topologie qui varie rapidement, chaque nœud du réseau transmet périodiquement sa table de routage à ses voisins directs. Le nœud peut aussi transmettre sa table de routage si le contenu de cette dernière subit des changements significatifs par rapport au dernier contenu envoyé. La mise à jour dépend donc de deux paramètres : *Le temps*, c'est à dire la période de transmission, et *Les événements (ou les déclencheurs)*, exemple : apparition d'un nœud, 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 du réseau.

La mise à jour de la table de routage peut se faire de deux façons [MIS 99] :

- *Une mise à jour complète*.
- *Une mise à jour incrémentale*.

Dans la mise à jour complète, la station transmet la totalité de la table de routage aux voisins ce qui nécessite l'envoi de plusieurs paquets de données ; alors que dans une mise à jour incrémentale, juste les entrées qui ont subit un changement par rapport à la dernière mise à jour, sont envoyées ce qui réduit le nombre de paquets transmis. La façon de faire la mise à jour des tables de routage est liée à la stabilité du réseau. Dans le cas où le réseau serait relativement stable, la mise à jour incrémentale est utilisée pour réduire le trafic de la communication, la mise à jour complète n'est pas fréquente dans ce genre de situation. Dans le cas opposé, où le réseau subit des changements rapides, le nombre de paquets incrémentals envoyés augmente, ce qui fait que la mise à jour complète est fréquente.

Un paquet de mise à jour contient :

1- *Le nouveau numéro de séquence incrémenté, du nœud émetteur*.

Et pour chaque nouvelle route :
2- *L'adresse de la destination*.
3- *Le nombre de nœuds (ou de sauts) séparant le nœud de la destination*.
4- *Le numéro de séquence (des données reçues de la destination) tel qu'il a été estampillé par la destination*.

Les données de routage reçues par une unité mobile, sont comparées avec les données déjà disponibles. La route étiquetée par la plus grande valeur du numéro de séquence (i.e. la route la plus récente), est la route utilisée. Si deux routes ont le même numéro de séquence, alors la route qui possède la meilleure métrique, est celle qui sera utilisée. La métrique utilisée dans le calcul des plus courts chemins est, tout simplement, le nombre de nœuds existant dans le chemin. Les valeurs des métriques des routes, choisies après réception des données de routage, sont incrémentées. Les modifications faites sur les données de routage locales, sont immédiatement diffusées à l'ensemble courant des voisins. Les routes reçues par une diffusion, seront aussi envoyées quand le récepteur procédera à l'envoi de ses paquets de routage. Le récepteur doit incrémenter les métriques des routes reçues avant l'envoi, car le récepteur représente un nœud en plus, qui participe dans l'acheminement des messages vers la destination. Un lien rompu est matérialisé par une valeur infinie de sa métrique, i.e. une valeur plus grande que la valeur maximale permise par la métrique [PER 94].

Le DSDV élimine les deux problèmes de boucle de routage "routing loop", et celui du "counting to infinity". Cependant, dans ce protocole, 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. Ce qui fait que le DSDV est lent. On trouve ce même problème dans l'algorithme DUAL [GAR 93] - utilisé dans des protocoles Internet tel que EIGRP[ALB 94] - et dans les algorithmes similaires basés sur la synchronisation explicite. En outre, le DSDV utilise une mise à jour périodique et basée sur les événements, ce qui cause un contrôle excessif dans la communication.

3.2.2 Le protocole de routage WRP

Le protocole de routage sans fil WRP (Wireless Routing Protocol) [MUR 95, MUR 96] est basé sur l'utilisation de la classe des *algorithmes de recherche de chemins,* PFA (Path-Finding Algorithm). Beaucoup d'algorithmes PFA existent dans la littérature [CHE 89, GAR 86, HAG 83, HUM 91, RAJ 91], ces algorithmes utilisent des données concernant la longueur et le nœud prédécesseur du chemin le plus court, correspondant à chaque destination ; et cela afin d'éviter le problème "counting to infinity" du DBF. Le problème des PFAs, est la présence des boucles de routage temporaires dans le chemin spécifié par le prédécesseur, avant qu'ils convergent. Afin de résoudre ce problème, le WRP utilise un algorithme de recherche de chemins [MUR 94], qui réduit les situations des boucles temporaires, et qui limite les mises à jour - uniquement - aux changements significatifs des entrées de la table de routage.

Dans ce protocole, chaque nœud maintient : *une table de distance*, *une table de routage*, *une table de coûts des liens* et *une liste de retransmission de messages* MRL (Message Retransmission List). La table de distance d'un nœud *i*, est une matrice qui contient pour chaque destination *j* et pour chaque voisin *k* de *i*, la distance D^i_{jk} et le prédécesseur P^i_{jk} de *k*. La table de routage d'un nœud i, est représentée par un vecteur dont chaque entrée est associée à une destination j connue. Chaque entrée spécifie :

1- L'identificateur (ou l'adresse) de la destination.
2- La distance vers la destination D^i_j.
3- Le nœud prédécesseur P^i_j, correspondant au plus court chemin choisi, pour atteindre la destination *j*.
4- Le successeur s^i_j, qui correspond au plus court chemin choisi pour atteindre j.
5- Une marque ou étiquette (tag^i_j), utilisée dans la mise à jour de la table de routage. Elle spécifie si l'entrée correspond à un chemin simple ($tag^i_j = correct$), une boucle ($tag^i_j = error$), ou à une destination qui n'a pas été marquée ($tag^i_j = null$).

La table des coûts des liens d'un nœud *i*, contient les coûts l^i_k pour chaque voisin *k*, et le nombre de durées périodiques de mise à jour (les timeouts ou les délais de garde) depuis que le nœud *i* avait reçut un message de type "error-free" provenant du nœud *k*. Le coût d'un lien défaillant est considéré comme étant infini.

La liste de retransmission de messages permet à un nœud donné, de connaître l'ensemble de voisins qui n'ont pas acquitté son message de mise à jour, et de retransmettre ce message à cet ensemble de voisins.

Un nœud envoi un message de mise à jour, s'il détecte un changement d'état d'un lien voisin, ou après la réception des données de mise à jour d'un autre voisin. Les nœuds présents dans la liste de réponse (Response List) du message de mise à jour (formé en utilisant la MRL), doivent acquitté la réception du message. S'il n'y a pas de changement, dans la table de routage, par rapport à la dernière mise à jour; le nœud doit envoyer un message "Hello" pour assurer la connexion. Lors de la réception du message de mise à jour, le nœud modifie sa distance et cherche les meilleurs chemins en se basant sur les informations reçues. La liste MRL, doit être mise à jour après chaque réception d'un acquittement "ACK".

Le WRP est caractérisé par sa vérification de la consistance des voisins, à chaque fois où un changement d'un lien voisin est détecté. La manière avec laquelle le WRP applique la vérification de la consistance, aide à éliminer les situations des boucles de routage et à minimiser le temps de convergence du protocole.

Pour illustrer le protocole WRP, nous allons, dans ce qui suit, reprendre l'exemple de Murthy et Garcia introduit dans [MUR 96]. Considérons un réseau formé de quatre unités représentées par les nœuds : I, J, B, K. Les coûts des liens sont indiqués dans la figure 3.1. Les nœuds source et destination sont respectivement *i*, *j*. Les flèches indiquent le sens de transfert des messages de mise à jour, et les étiquettes, sous forme de couples, donnent la distance et le prédécesseur de la destination *j*. Chaque message de mise à jour est acquitté par un message ACK, qui n'est pas représenté dans la figure, envoyé par le nœud voisin. La figure 3.1 représente que les messages destinés au nœud *j* seulement.

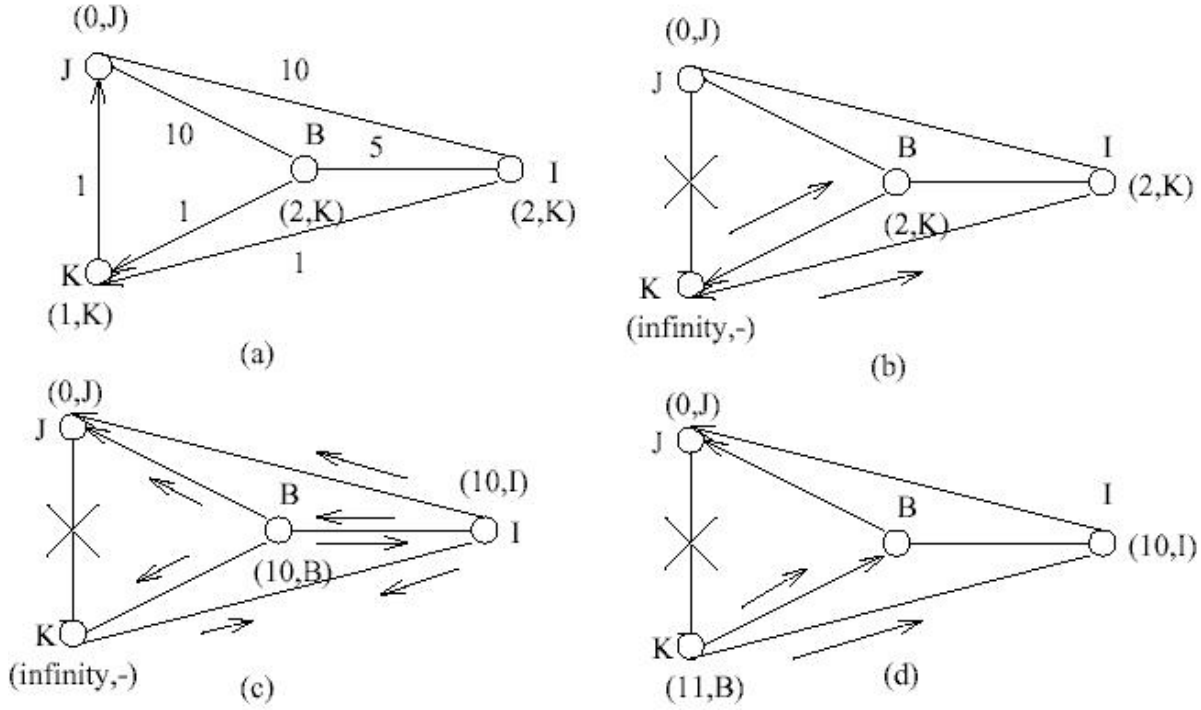


Figure 3.1 : Un exemple d'exécution du protocole WRP.

Quand le lien (*j*, *k*) devient défaillant, les nœuds *j* et *k* envoient des messages de mise à jour à leurs voisins, comme c'est représenté dans la figure 3.1(b). Dans cet exemple le nœud *k* doit envoyer la distance vers *j*, ayant la valeur "infinie". Car le nœud *k*, fait partie de leurs chemins de routage vers la destination *j*. Le nœud *b* traite le message de *k* et sélectionne le lien (*b*, *j*) pour la destination *j*. Quand le nœud *i*, reçoit le message de *k*, il met à jour sa table de distance et examine les chemins possibles vers la destination *j*, à travers les autres nœuds voisins et par la suite, il met à jour les entrées des tables de distance et de routage, celons les résultats obtenus. Comme c'est montré dans la figure précédente, le nœud *i*, sélectionne le lien (*i*, *j*) pour la destination *j*. Le nœud *i* ignore tous les messages de mise à jour, qui n'ont pas d'effet sur le chemin de routage de *i* vers *j*. Par exemple le message de *k*, qui comporte la distance 11 pour la destination *j*, est ignoré.

3.2.3 Le protocole de routage GSR

Le protocole appelé "Routage à Etat Global" ou GSR (Global State Routing) [CHE 98], est protocole similaire au protocole DSDV décrit précédemment. Ce protocole utilise les idées du routage basé dur l'état des liens (LS), et les améliore en évitant le mécanisme inefficace, de l'inondation des messages de routage. Le GSR utilise une vue *globale* de la topologie du réseau, comme c'est le cas dans les protocoles LS. Le protocole utilise aussi une méthode (appelée la méthode de dissémination) utilisée dans le DBF, qui a l'avantage de l'absence de l'inondation.

Dans ce protocole, chaque nœud *i* maintient : *une liste de voisins* A_i, *une table de topologie* TT_i, *une table des nœuds suivants* $NEXT_i$ (Next Hop), et *une table de distance* D_i. La table de la topologie TT_i, contient pour chaque destination, l'information de l'état de lien telle qu'elle a été envoyée par la destination, et une estampille de l'information. Pour chaque nœud de destination *j*, la table $NEXT_i$ contient le nœud, vers lequel les paquets destinés à *j* seront envoyés. La table de distance, contient la plus courte distance pour chaque nœud destination.

De la même manière des protocoles LS, les messages de routage sont générés suivant les changements d'états des liens. Lors de la réception d'un message de routage, le nœud met à jour sa table de topologie et cela dans le cas où le numéro de séquence du message reçu serait supérieur à la valeur du numéro de séquence sauvegardée dans la table (exactement comme le fait le protocole DSDV). Par la suite, le nœud reconstruit sa table de routage et diffuse les mise à jour à ses voisins. Le calcul des chemins, peut se faire avec n'importe quel algorithme de recherche des plus courts chemins. Par exemple, dans [CHE 98], l'algorithme du GSR utilise l'algorithme de Dijkstra [SED 83] modifié de telle façon qu'il puisse construire la table des nœuds suivants (NEXT) et la table de distance (D), en parallèle avec la construction de l'arbre des plus courts chemins (l'arbre dont la racine est le nœud source).

La différence clé entre le GSR et le LS traditionnel, est la façon dans laquelle les informations de routage circulent dans le réseau. Dans le LS, si on détecte des changements de la topologie, les paquets d'états de liens sont générés et diffusés par inondation dans tout le réseau. Par contre, le GSR maintient la table - la plus récente - d'état des liens reçus à travers les voisins, et l'échange uniquement avec ses voisins locaux, d'une façon périodique. En plus de cela, le GSR assure plus de précision, concernant les données de routage qui s'échangent dans le réseau.

3.2.4 Le protocole de routage FSR

Le protocole "Routage à Etat de l'œil du Poisson" FSR, (Fisheye State Routing) [IWA 99] peut être vu comme une amélioration du protocole GSR précédent. Le nombre élevé des messages de mise à jour échangés, implique une grande consommation de la bande passante, ce qui a un effet négatif dans les réseaux ad hoc caractérisés par une bande passante limitée. Le protocole FSR est basé sur l'utilisation de la technique "œil de poisson" (fisheye), proposée par Kleinrock et Stevens [KLE 71], qui l'ont utilisé dans le but de réduire le volume d'information nécessaire pour représenter les données graphiques. L'œil d'un poisson capture avec précision, les points proches du point focal. La précision diminue quand la distance, séparant le point vu et le point focal, augmente. Dans le contexte du routage, l'approche du "fisheye" matérialise, pour un nœud, le maintien des données concernant la précision de la distance et la qualité du chemin d'un voisin direct, avec une diminution progressive, du détail et de précision, quand la distance augmente.

Le protocole FSR est similaire au protocole LS, dans sa sauvegarde de la topologie au niveau de chaque nœud. La différence principale, réside dans la manière avec

laquelle les informations de routage circulent. Dans le FSR, la diffusion par inondation de messages n'existe pas. L'échange se fait uniquement avec les voisins directs.

Les données, de mise à jour, échangées périodiquement dans le FSR, ressemble au vecteur échangé dans le protocole DSDV, où les distances sont modifiées suivant l'estampille du temps ou le numéro de séquence associé au nœud qui a été l'origine de la mise à jour. Dans le FSR (comme dans le LS) les états de liens sont échangés, l'image complète de la topologie du réseau est gardée au niveau de chaque nœud, et les meilleurs chemins sont échangés en utilisant cette image.

Comme nous avons déjà dit, l'état des liens change fréquemment dans les réseaux ad hoc. Le FSR effectue la mise à jour de ces changements de la même manière que le protocole GSR; ce qui résout les problèmes du protocole LS concernant le volume des données de contrôle. Avec le GSR, et quand la taille du réseau devient très grande, les messages de mise à jour peuvent consommer considérablement la bande passante. Afin de réduire le volume de messages échangés sans toucher à la consistance et la précision des données de routage; le FSR utilise la technique "œil de poisson". La figure 3.2, illustre cette technique. Dans cette figure, on définit la portée, ou si on veut le champ de vision, de l'œil de poisson pour le nœud du centre, d'identificateur 5. La portée est définie en termes de nœuds, qui peuvent être atteints en passant par un certain nombre de sauts. Le nœud du centre (le nœud 11), maintient les données les plus précises des nœuds appartenant au cercle, la précision diminue progressivement, pour les cercles moins proches du centre.

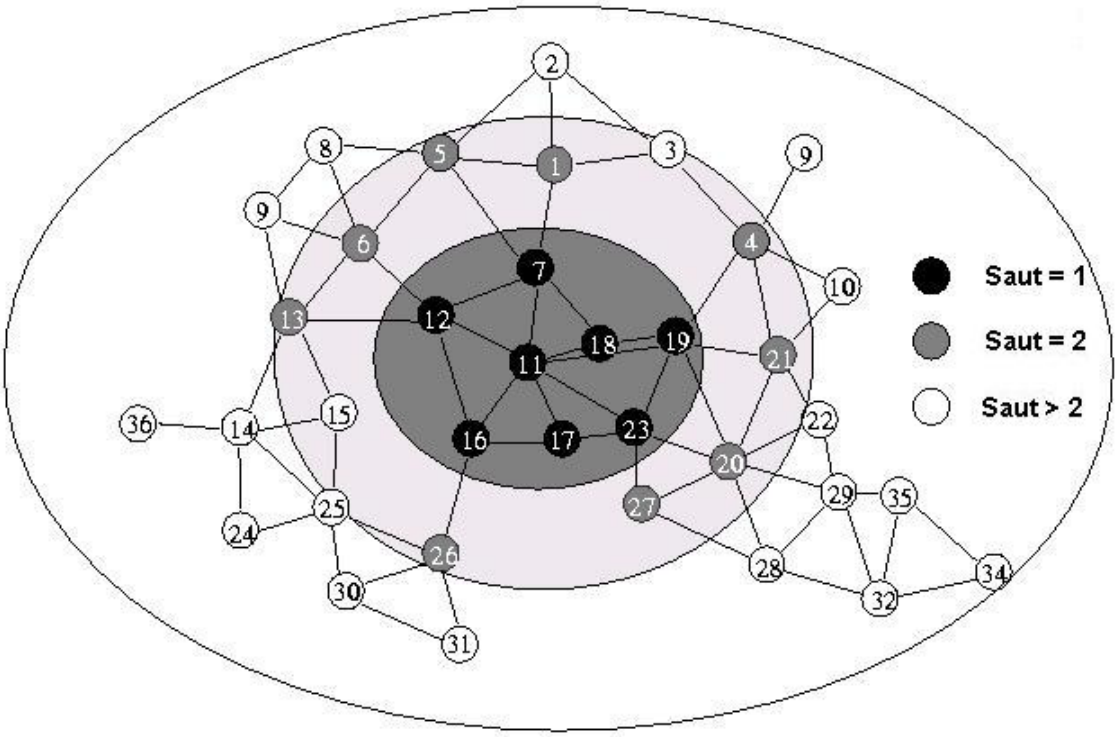


Figure 3.2 : La technique "œil de poisson".

La réduction de volume des données de mise à jour, est obtenue en utilisant des périodes d'échanges différentes pour les différentes entrées de la table. Les entrées qui

correspondent aux nœuds les plus proches, sont envoyés aux voisins avec une fréquence élevée (donc avec une période d'échange relativement petite). Par exemple, les entrées en gras (figure 3.3) des tables de routage, sont échangés fréquemment. Le reste des entrées, est échangé avec une fréquence moins élevée. De cette manière, un grand nombre de données de routage est évité, ce qui réduit le volume des messages qui circule dans le réseau.

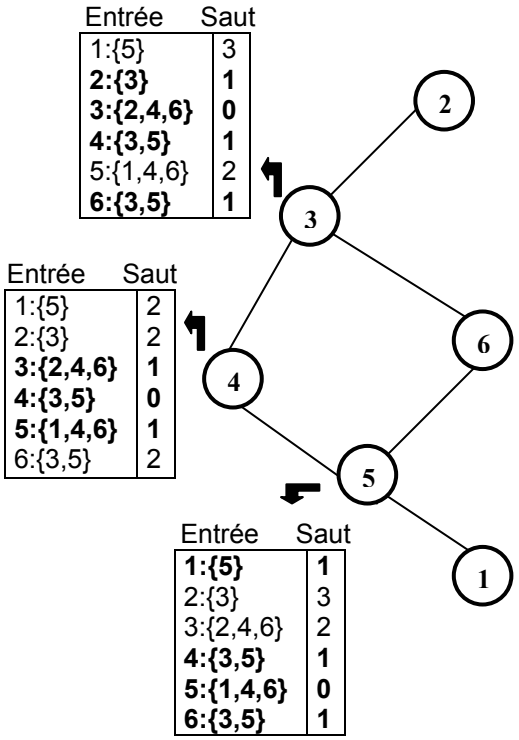


Figure 3.3 : L'échange de messages avec la technique "œil de poisson".

Nous pouvons évaluer le nombre de messages de mise à jour échangés *m*, durant une unité de temps et pour un nœud donné, par la formule suivante :

$$m = \sum_{i=1}^{K} f_i n_i.$$

Avec k est le nombre de sauts, f_i est la fréquence d'échange de message associée au saut i, et n_i est le nombre d'unités mobiles appartenant au saut i (la somme des n_i est égale à N, le nombre total d'unités dans le réseau). Notons que ce nombre est inférieur au nombre de messages échangés, sans l'utilisation de la technique "œil de poisson", et qui est égal à f.N, avec f est la fréquence d'échange (qui est constante pour toutes les entrées). La diminution progressive des fréquences peut être formulée comme suit :

$$f_1 > f_2 > \ldots > f_{k-1} > f_k.$$

Sachant que f_1 est inférieur ou égal à f.

Puisque la technique "œil de poisson" est réalisée, dans le protocole FSR, en se basant sur le changement des périodes. Les paquets de mises à jour, arrivent lentement aux nœuds qui sont loin de la source. Cependant, la connaissance imprécise du meilleur chemin vers une destination distante, est compensé par le fait que la route

devient progressivement plus précise quand les paquets s'approchent peut à peut de la destination.

Le protocole FSR peut être utilisé dans les réseaux ad hoc dont le nombre d'unités mobiles est grand. Le protocole utilise un volume raisonnable de messages de contrôle, en outre, il évite le travail énorme de recherche de chemins, effectué dans les protocoles réactifs (comme nous allons voir plus loin); ce qui accélère la transmission. En plus de cela, le FSR maintient des calculs précis concernant les destinations proches.

3.2.5 Le protocole de routage HSR

La notion de partitionnement et de groupes est très répandue en pratique dans les réseaux mobiles ad hoc [GER 95, CHI 97]. La notion de groupe, peut améliorer les performances des réseaux. Par exemple, les interférences des signaux, peuvent être réduites en utilisant différents codes étendus à l'aide des groupes
[GER ??]. En plus de cela, le partitionnement peut être exploité dans les réseaux de grande taille afin de réaliser un routage hiérarchique, ce qui réduit le contrôle des données de routage. Le problème principal du routage hiérarchique dans les réseaux sans fil, est la mobilité et la gestion de la localisation. Dans le but de résoudre ce problème, le protocole "Routage à Etat Hiérarchique" ou HSR (Hierarchical State Routing) a été proposé [IWA 99]. Le protocole combine les notions de groupes dynamiques, niveaux hiérarchiques avec une gestion efficace de localisation.

Dans le HSR, l'image de la topologie du réseau, est sauvegardée sous forme hiérarchique. Le réseau est partitioné 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 bas, 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 [GER 95, CHI 97]. Le but principal du partitionnement du HSR, est l'utilisation efficace des médiums de communication, et la réduction du contrôle de routage, effectué par la couche réseau (i.e. le la sauvegarde des tables de routage, le traitement et la transmission des données).

La figure 3.4, illustre l'application de ce mécanisme de partitionnement dans un réseau de 13 unités mobiles. Le réseau est décomposé en 4 groupes, qui sont : G0-1, G0-2, G0-3, et G0-4. Ces groupes forment le niveau le plus bas de la hiérarchie (niveau 0). A partir de ce niveau, les niveaux qui suivent (niveau 1 et 2), sont formés. Cela est fait, en prenant l'ensemble des représentants de groupes et le décomposer en groupes, de la même manière précédente. Dans la décomposition en groupe, on peut avoir 3 types de nœuds : *un nœud représentant du groupe* (appelé aussi, tête du groupe) par exemple, les nœuds 1, 2, et 3 de la figure 3.4; *un nœud de liaison*, qui relie deux groupes (exemple, les nœuds 7 et 9); et *un nœud interne* qui n'a aucun rôle spécial (exemple, les nœuds 4, 11 et 12). Le nœud représentant d'un groupe donné, peut être vu comme un coordinateur de transmission de données. Les identificateurs

(IDs) des nœuds représentés dans la figure 3.3 (niveau 0), sont des adresses physiques. Ils sont uniques pour chaque nœud. Une des méthodes qui peuvent être appliquées 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 précédente à l'adresse HID(6) = <1,1,6>, le chemin reliant la racine et le nœud 6, est composé de 3 nœuds : le représentant du groupe G1-1 (d'où la première composante est 1, i.e. le numéro du groupe), le représentant du groupe G0-1 (d'où la deuxième composante est 1), et enfin le nœud 6 d'ID égal à 6 d'où la dernière composante de l'adresse est 6.

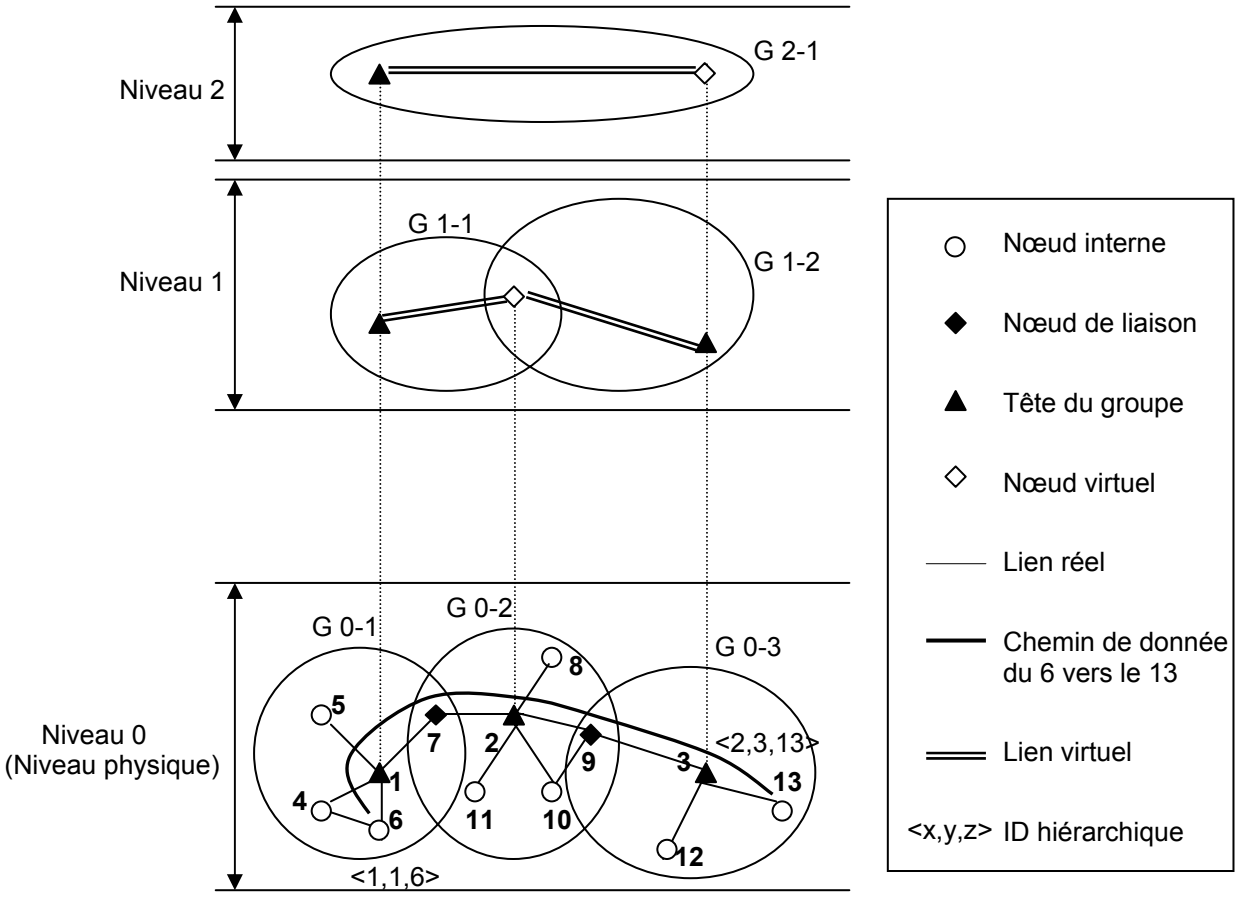


Figure 3.4 : Le partitionnement du réseau en groupes.

Un nœud de liaison, peut être atteint - a partir de la racine - en suivant plusieurs chemins, par conséquent, ce genre de nœud peut avoir plus d'une adresse hiérarchique. Cela ne pause 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 les quels appartient le nœud. Exemple : <1,1,7> est une adresse du nœud de liaison d'ID 9.

Dans la figure 3.4, le nœud 3 est membre du groupe hiérarchique le plus élevé

(niveau 2), il est aussi, le représentant du groupe G1-2. Le nœud 2 est un membre du groupe G1-2, et en même temps il est le représentant du groupe G0-2.

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 supérieurs. L'adresse hiérarchique, suffit pour délivrer les paquets de données à une destination, indépendamment de la localisation de la source, et cela en utilisant la table HSR. Prenant comme exemple le nœud 6 (figure 3.4) comme source, et le nœud 13 comme destination. Les adresses de ces nœuds sont respectivement : HID(6) = <1,1,6> et HID(13) = <2,3,13>. Pour acheminer une information du nœud 6 vers le nœud 13, le nœud 6 envoie l'information au nœud supérieur, qui le suit hiérarchiquement, i.e. le nœud d'ID 1. Le nœud 1, délivre l'information au nœud 3 qui suit le nœud destination dans l'ordre hiérarchique. Un "lien virtuel" existe entre les nœuds 1 et 3, qui est matérialisé par le chemin (1,7,2,9,3); par conséquent l'information suivra ce chemin pour atteindre la destination. Dans la dernière étape, le nœud 3, délivre l'information au nœud 13, en suivant le chemin hiérarchique qui lui relie avec la destination, dans notre cas, ce chemin se réduit en un seul saut.

En plus de la décomposition (ou du partitionnement) en groupes basé sur les relations géographiques (physiques) entre les différents nœuds, le protocole HSR utilise aussi un partitionnement logique. Ce partitionnement est basé sur des relations logiques qui peuvent exister entre les nœuds du réseau, comme par exemple l'appartenance à une même société ou organisme … etc. La partitionnement logique joue un rôle clé dans la gestion de localisation. En plus des adresses physiques, une adresse logique de la forme < *subnet*, *host* >, est associée à chaque nœud. Les adresses ont un format similaire à au format IP. En effet, elles peuvent être vues comme des adresses IP privées dans le réseau mobile. Chaque *subnet* correspond à un groupe particulier d'utilisateur (ensemble d'unités qui partagent des caractéristiques communes) qui possède un serveur de gestion de localisation dit LMS (Location Management Server). Différents ensembles de mobiles (partageants des caractéristiques plus restrictives), peuvent être définis indépendamment dans chaque *subnet*. Quand la couche de transport délivre au réseau un paquet contenant l'adresse IP privée. Le réseau doit trouver, à partir de l'adresse IP, l'adresse hiérarchique basée sur les adresses physiques.

Notons que le groupe à qui correspond l'adresse IP privée, est un ensemble de groupes du partitionnement physique. Chaque réseau virtuel a au moins *un agent principal* (qui aussi un membre du réseau) dans le but de gérer les différents membres. Tous les agents principaux annoncent leurs HIDs au niveau hiérarchique supérieur, les HIDs peuvent être aussi envoyés aux niveaux les plus bas hiérarchiquement [GER ??, MIS 99]. Chaque membre d'un *subnetwork* logique, connaît le HID de son agent principal (en utilisant la table de routage), il peut donc enregistrer son adresse hiérarchique. L'enregistrement est à la fois périodique et basé-événement (par exemple, dans le cas où le nœud se déplacerais vers une nouvelle partition physique). L'agent principal utilise la technique du timeout afin d'éliminer les adresses non renouvelées. Le trafic du contrôle induit par les opérations d'enregistrement d'adresses est réduit, car dans la plupart des applications, les membres

d'un même *subnet* se déplacent en groupe, ce qui implique qu'ils appartiendront à des partitions voisines.

Quand un nœud source veut envoyer des données à un autre nœud dont l'adresse IP est connue; il extrait d'abord le champ *subnet* de l'adresse, et en utilisant sa liste (ou celle du niveau hiérarchique supérieur) il obtient l'adresse hiérarchique de l'agent principal du nœud destination (rappelons que tous les agents principaux, annoncent leurs HIDs au niveau hiérarchique supérieur). Le nœud source envoie alors, les données à l'agent principal en utilisant l'adresse hiérarchique obtenue. Lors de la réception, l'agent principal trouve l'adresse de la destination enregistrée et cela à partir de l'ID de l'hôte extrait de l'adresse IP. Par la suite, l'agent délivre les données vers les nœuds destination. Une fois les deux nœuds, la source et la destination, connaissent leurs adresses hiérarchiques, les messages peuvent être délivrés directement sans l'intervention des agents principaux.

3.2.6 Le protocole de routage ZHLS

Le protocole "Routage à Etat de Liens Hiérarchique basé sur les Zones", appelé ZHLS (Zone-Based Hierarchical Link State Routing) [JOA 99], est basé sur la décomposition du réseau en un ensemble de zones. Dans ce protocole, les membres d'une zone n'élisent pas de représentants, contrairement à ce qui fait dans les autres protocoles hiérarchiques. Avec cette décomposition, on a deux niveaux de topologies : *le niveau nœud*, et *le niveau zone*. La topologie basée sur le premier niveau, donne la façon dans laquelle les nœuds, d'une zone donnée, sont connectés physiquement. Un *lien virtuel* peut exister entre deux zones, s'il existe au moins un nœud de la première zone, qui soit physiquement connecté à un nœud de l'autre zone (figure 3.5). La topologie basée sur le niveau zone, donne le schéma de la connexion des différentes zones.

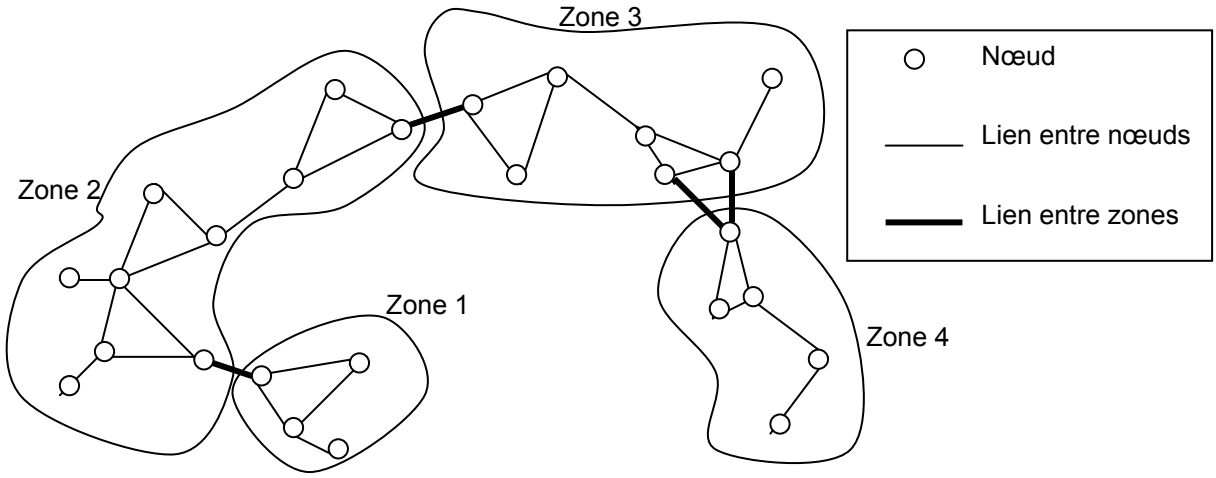


Figure 3.5 : La décomposition du réseau en zones.

Dans ce protocole, les paquets qui contiennent les états des liens ou les LSPs (Link State Packet), peuvent être divisées en deux classes : les LSPs orientés nœuds, et les LSPs orienté zones. Pour un nœud donné, un paquet LSP orienté nœud, contient l'information d'un nœud voisin, tandis qu'un paquet LSP orienté zone, contient l'information de la zone et il est échangé dans une manière globale. De cette façon, chaque nœud du réseau possède une connaissance complète concernant les nœuds de sa propre zone, et seulement une connaissance partielle du reste des nœuds, cette connaissance partielle est matérialisée par l'état de la connexion des différentes zones du réseau. Par conséquent, l'acheminement des données, se fait de deux façons : le routage *inter zone*, et le routage *intra zone*. Pour une destination donnée, les données sont envoyées entres les zones en utilisant les identificateurs des zones, jusqu'à ce que les données atteignent la zone finale de la destination. Par la suite, les paquets de données sont circulent à l'intérieur de la zone finale, en utilisant l'identificateur du nœud destination. L'adresse < ID zone, ID nœud >, est suffisante pour atteindre n'importe quelle destination même si le réseau change de topologie.

3.2.7 Le protocole de routage CGSR

Le protocole appelé CGST (Clusterhead Gateway Switch Routing) [CHI 97], utilise principalement l'algorithme de routage DSDV, décrit dans la section 3.3.1. L'ensemble des unités mobiles du réseau est décomposé en groupes. Un membre de chaque groupe est élu. Les nœuds appartenants à la portée de communication d'un représentant de groupe (ceux qui peuvent communiquer avec le représentant), appartiennent au groupe représenté par ce dernier. Un nœud de liaison, est un nœud qui appartient à la portée de communication de plus d'un représentant de groupe. Cette manière d'organisation, peut dégrader les performances des réseaux ad hoc à cause des changements fréquents de leur topologie. Pour s'adapter à ces changements, le CGSR utilise pour cela un algorithme appelé LGC (Least Cluster Change). Dans cet algorithme, un changement de représentants de groupes arrive, seulement dans le cas où il y aurait une fusion de deux groupes (les deux anciens groupes, se transforment en un nouveau groupe), ou dans le cas où un nœud sortirait complètement de la portée de tous les représentants du réseau.

Dans le protocole CGSR, le routage des informations se fait de la manière suivante : le nœud source transmet ses paquets de données à son représentant de groupe. Le représentant envoie les paquets au nœud de liaison, qui relie ce représentant avec le représentant suivant dans le chemin qui existe vers la destination. Le processus se répète, jusqu'à ce qu'on atteigne le représentant du groupe dans lequel appartient la destination. Ce représentant, transmet alors les paquets reçus vers le nœud destination. La figure suivante, donne le chemin de routage des paquets de données, à partir du nœud source 9, jusqu'au nœud destination 18.

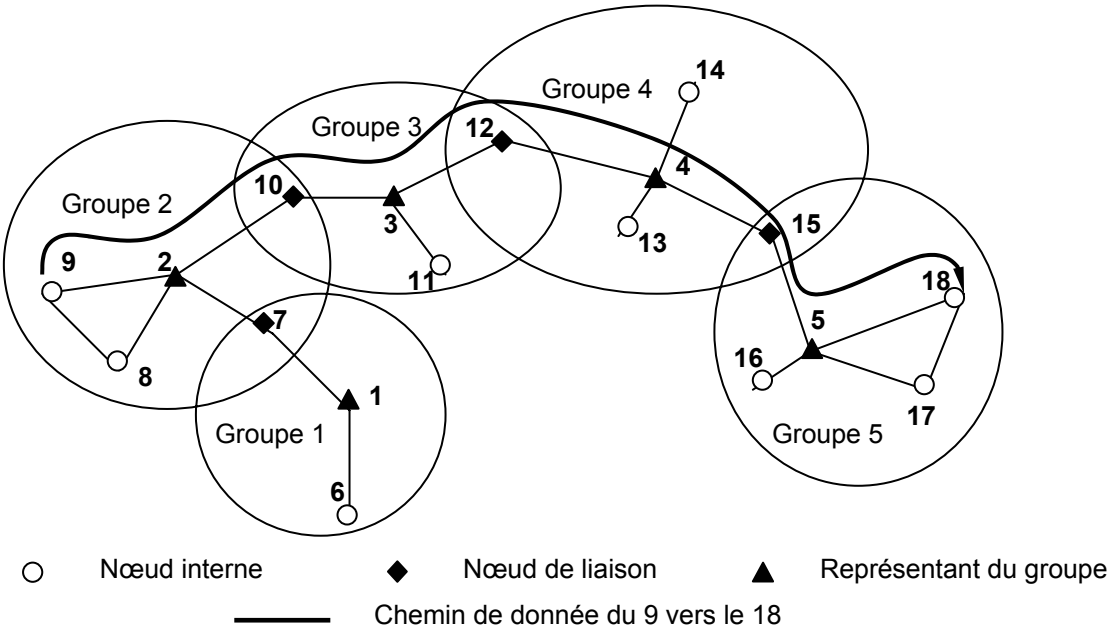


Figure 3.6 : Un exemple d'acheminement d'information dans le CGSR.

Chaque nœud maintient *une table de membres de groupes*, qui associe à chaque nœud, l'identificateur d'un représentant de groupe. Chaque nœud diffuse cette table d'une façon périodique et met à jour sa propre table (après la réception des données de mise à jour provenant d'un autre nœud), en utilisant l'algorithme DSDV. En outre, chaque nœud maintient une table de routage, qui détermine le nœud suivant correspondant au groupe destination.

Lors de la réception d'un paquet, le nœud intermédiaire trouve le représentant de groupe (*h* par exemple) le plus proche dans le chemin envisagé vers la destination, et cela en utilisant sa table de membres de groupes et sa table de routage. Par la suite le nœud consulte sa table de routage, pour trouver le nœud suivant afin d'atteindre le représentant *h*. Les paquets seront transmis alors, au nœud trouvé.

Notons que cette manière de routage assure un procédé déterministe et efficace pour l'acheminement des informations, cependant un chemin choisit peut ne pas être optimal. C'est le cas de l'exemple précédent, si on suppose que tous les coûts des liens sont égaux (unitaires par exemple), le chemin (9,2,10,3,12,4,15,5,18) entre la source 9 et la destination 18, ne représente pas le meilleur chemin qui existe. En effet le chemin (9,2,10,3,12,4,15,18) est meilleur. Cela est dû au fait, que tous les nœuds applique la même stratégie, le nœud d'identificateur 15 trouve - en utilisant sa table de routage- que le nœud suivant correspondant au nœud 18 est le nœud 18 lui-même. Le nœud 15, consulte sa table de membres de groupes pour connaître le représentant du groupe associé au nœud 18, le nœud trouvé est alors celui de l'ID 5 ce qui fait que les paquets passent par le nœud 5 et ne passent pas directement vers la destination.

3.2.8 Le protocole de routage DREAM

Le protocole appelé "Algorithme d'Effet de Routage basé sur la Distance, pour la Mobilité" ou DREAM (Distance Routing Effect Algorithm for Mobility) [BAS 98] est

un protocole pro-actif basé sur les informations des localisations des unités mobiles. Le protocole diffuse les données destinées à une certaine destination en effectuant une inondation (propagation) partielle.

Chaque nœud du réseau mobile ad hoc, échange périodiquement des messages de contrôle afin d'informer tous les autres nœuds de sa localisation. La distance influe dans cet échange, du fait que les messages de contrôle sont envoyés fréquemment aux nœuds les plus proches (cela nous rappelle la technique FSR, vue dans la section 3.2.4). En plus de cela, le protocole s'adapte à la mobilité du réseau par le contrôle de mise à jour de fréquence qui se base sur les vitesses des mouvements.

Lors de l'envoi des données, si la source possède des informations récentes sur la localisation du nœud destination, elle choisit un ensemble de nœuds voisins qui sont localisés dans la direction source/destination. Si un tel ensemble n'existe pas, les données sont inondées dans le réseau entier. Dans le cas où de tels nœuds existeraient, une liste qui contient leurs identificateurs, est insérée à la tête du paquet de données avant la transmission. Seulement les nœuds qui sont spécifiés dans la liste de tête, traitent le paquet. Lors de la réception du paquet, le nœud de transit, détermine sa propre liste des nœuds prochains, et envoie le paquet avec la nouvelle liste de tête. Si aucun voisin n'est localisé dans la direction de la destination, le paquet reçu est ignoré. Quand le nœud destination reçoit les données, il envoie des acquittements à la source d'une manière similaire. Cependant, dans le cas de réception par inondation, les acquittements ne sont pas envoyés. Dans le cas où la source envoie les données en spécifiant les nœuds suivants (en se basant sur les localisations), un timer associé à la réception des acquittements est activé. Si aucun acquittement n'est reçu avant l'expiration du timeout, les données seront retransmises en utilisant une diffusion ordinaire.

3.3 Les protocoles de routage réactifs (à la demande)

Comme nous avons vu dans la section précédente, les protocoles de routage pro-actifs essaient de maintenir les meilleurs chemins existants vers toutes les destinations possibles (qui peuvent représenter l'ensemble de tous les nœuds du réseau) au niveau de chaque nœud du réseau. Les routes sont sauvegardées mêmes si elles ne sont pas utilisées. La sauvegarde permanente des chemins de routage, est assurée par un échange continue des messages de mise à jour des chemins, ce qui induit un contrôle excessif surtout dans le cas des réseaux de grande taille.

Les protocoles de routage réactifs (dits aussi : protocoles de routage à la demande), représentes les protocoles les plus récents proposés dans le but d'assurer le service du routage dans les réseaux sans fil. La majorité des solutions proposées pour résoudre le problème de routage dans les réseaux ad hoc, et qui sont évaluées actuellement par le groupe de travail MANET (Mobile Ad Hoc Networking Working Groupe) de l'IETF (Internet Engineering Task Force) [MAC 2000], appartiennent à cette classe de protocoles de routage [PER 97, PAR 99, BRO 98].

Les protocoles de routage appartenants à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information

spécifiée, inconnue au préalable. Plusieurs approches peuvent être appliquées dans la découverte des routes. La majorité des algorithmes utilisés, sont basé sur le mécanisme d'*apprentissage en arrière* (backward learning) [GER ??]. Le nœud source, qui est à la recherche d'un chemin vers la destination, diffuse par inondation une requête dans le réseau. Lors de la réception de la requête, les nœuds intermédiaires (ou de transit) essaient de faire *apprendre* le chemin au nœud source, et de sauvegarder la route dans la table envoyée. Une fois la destination est atteinte, elle peut envoyer une réponse en utilisant le chemin tracé par la requête, un chemin full duplex est alors établit entre le nœud source et le nœud destination. Le travail peut être réduit, dans le cas où un nœud de transit posséderait déjà un chemin vers la destination. Une fois le chemin est calculé, il doit être sauvegardé et mis à jour au niveau de la source, tant qu'il est en cours d'utilisation. Une autre technique pour tracer les chemins demandés, est la technique appelé "routage source" (utilisé dans le protocole DSR que nous allons voir par la suite).

Le routage à la demande induit une lenteur à cause de la recherche des chemins, ce qui peut dégrader les performances des applications interactives (exemple les applications des bases de données distribuées). En outre, il est impossible de connaître au préalable la qualité du chemin (en termes de bande passante, délais,… etc.). Une telle connaissance est importante dans les applications multimédias [GER ??].

3.3.1 Le protocole de routage CBRP

Dans le "Protocole de Routage Basé sur les Groupes" appelé CBRP (Cluster Based Routing Protocol) [JIA 99], l'ensemble des nœuds du réseau est décomposé en groupes. Le principe de formation des groupes est le suivant : Un nœud *p* qui n'a pas de statut (i.e. qui n'est ni membre ni représentant de groupe), active un timer et diffuse un message "Hello". Lorsqu'un représentant de groupe reçoit le me ssage "Hello", il envoie immédiatement une réponse à l'émetteur. Lors de la réception de réponse, le nœud *p* change son état "indécidé" à l'état "membre". Si *p* dépasse un certain timeout en attendant la réponse et dans le cas où il possède un lien bidirectionnel vers aumoins un nœud voisin, il considère lui-même un représentant de groupe. Dans le cas contraire, p reste dans l'état indécidé et il répète la même procédure. A cause des changements rapides de la topologie des réseaux ad hoc, l'attente des nœuds indécidés est très courte.

Afin de sauvegarder la répartition des nœuds dans les groupes, chaque nœud maintient *une table des voisins*. Chaque entrée de cette table est associée à un voisin, elle indique l'état du lien (uni ou bidirectionnel), et le statut du voisin (membre ou représentant de groupe). Un représentant de groupe, maintient les informations des membres qui appartiennent à son groupe. Il possède aussi *une table des groupes adjacents*. Une entrée dans cette table est associée à un groupe voisin, elle contient : l'identificateur du groupe, et l'identificateur du nœud de liaison à travers lequel le groupe peut être atteint (voir la figure suivante).

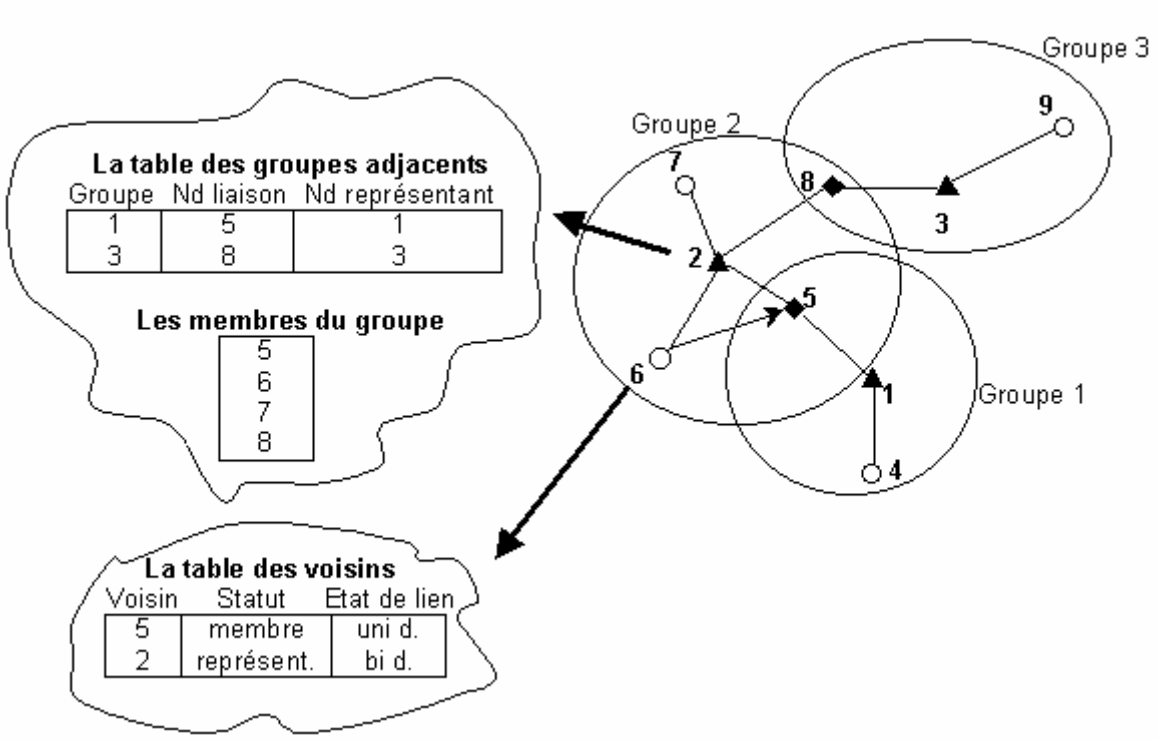


Figure 3.7 : L'organisation du réseau dans le CBRP.

Le routage dans le protocole CBRP se fait de la manière suivante : quand un nœud source veut envoyer des données à un nœud destination, il diffuse par inondation une requête de demande de chemin, *et cela uniquement aux représentants des groupes voisins*. Un représentant de groupe qui reçoit la requête de demande, vérifie - en utilisant sa table de membres de groupes - l'existence du nœud destination dans son groupe. Si la destination existe, le représentant y envoie directement la requête, dans le cas contraire, la requête est diffusée aux représentants des groupes voisins. L'adresse des représentants des groupes est incluse dans la requête de demande de chemin, un représentant de groupe ignore toute requête déjà traitée. Quand la destination reçoit le paquet contenant la requête, elle répond par l'envoie du chemin qui a été sauvegardé dans le paquet de la requête. Dans le cas où le nœud source ne reçoit pas de réponse en expirant une certaine période, il envoie de nouveau une requête de demande de chemin.

Lors de l'acheminement des données, si un nœud détecte qu'un lien est défaillant, il fait retourner un message d'erreur à la source et il applique *un mécanisme de réparation locale*. Dans ce mécanisme, si un nœud *p* trouve qu'un nœud suivant *n*, ne peut pas être atteint, il essaie de vérifier si le nœud *n* ou le nœud qui vient après *n*, peuvent être atteints à travers un autre nœud voisin. Si l'un des deux cas est vérifié, les données sont envoyées en utilisant le chemin réparé.

3.3.2 Le protocole de routage DSR

Le protocole "Routage à Source Dynamique" (DSR : Dynamic Source Routing) [], est basé sur l'utilisation de la technique "routage source". Dans cette technique, la

source des données détermine la séquence complète des nœuds à travers lesquelles, les paquets de données seront envoyés.

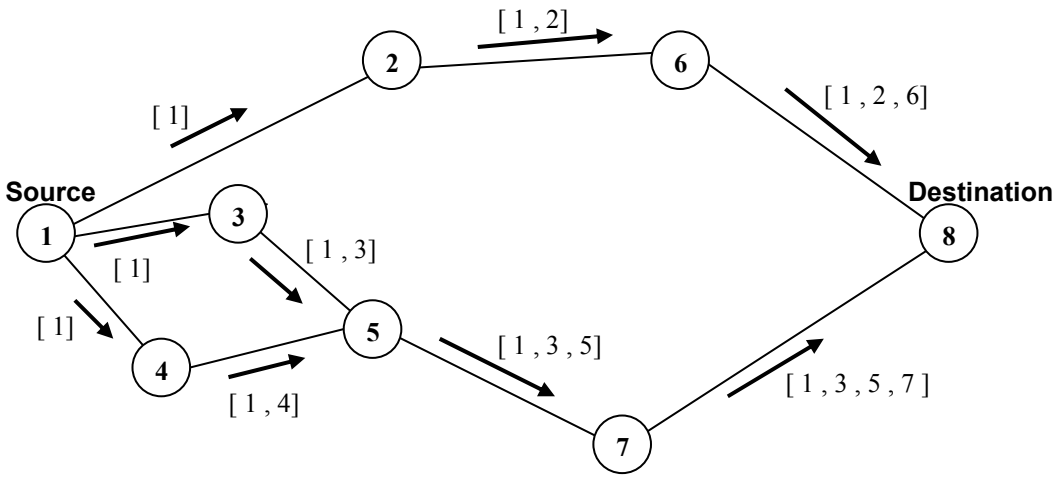
Afin d'envoyer un paquet de donnée à un autre nœud, l'émetteur construit une route source et l'inclut en tête du paquet. La construction se fait en spécifiant l'adresse de chaque nœud à travers lequel le paquet va passer pour atteindre la destination. Par la suite, l'émetteur transmet le paquet, à l'aide de son interface, au premier nœud spécifié dans la route source. Un nœud qui reçoit le paquet, et qui est différent de la destination, supprime son adresse de l'entête du paquet reçu le transmet au nœud suivant identifié dans la route source. Ce processus se répète jusqu'à ce que le paquet atteigne sa destination finale. Enfin, le paquet est délivré à la couche réseau de le dernier hôte. Les deux opérations de base du protocole DSR sont : *la découverte de routes* (route discovery) et *la maintenance de routes* (route maintenance).

L'opération (ou le protocole) de découverte de routes, permet à n'importe quel nœud du réseau ad hoc découvrir *dynamiquement* un chemin vers un nœud quelconque du réseau. Un hôte initiateur de l'opération de découverte, diffuse un paquet *requête de route* qui identifie l'hôte cible (la destination). Si l'opération de découverte est réussite, l'hôte initiateur reçoit un paquet *réponse de route* qui liste la séquence de nœuds à travers lesquels la destination peut être atteinte. En plus de l'adresse de l'initiateur, le paquet *requête de route* contient un champ *enregistrement de route*, dans lequel est accumulée la séquence des nœuds visités durant la propagation de la requête de route dans le réseau (voir la figure 3.8(a)). Le paquet *requête de route*, contient aussi un identificateur unique de la requête. Dans le but de détecter les duplications de réceptions de la requête de route, chaque nœud du réseau ad hoc maintient une liste de couples < adresse de l'initiateur, identificateur de requête>, des requêtes récemment reçues.

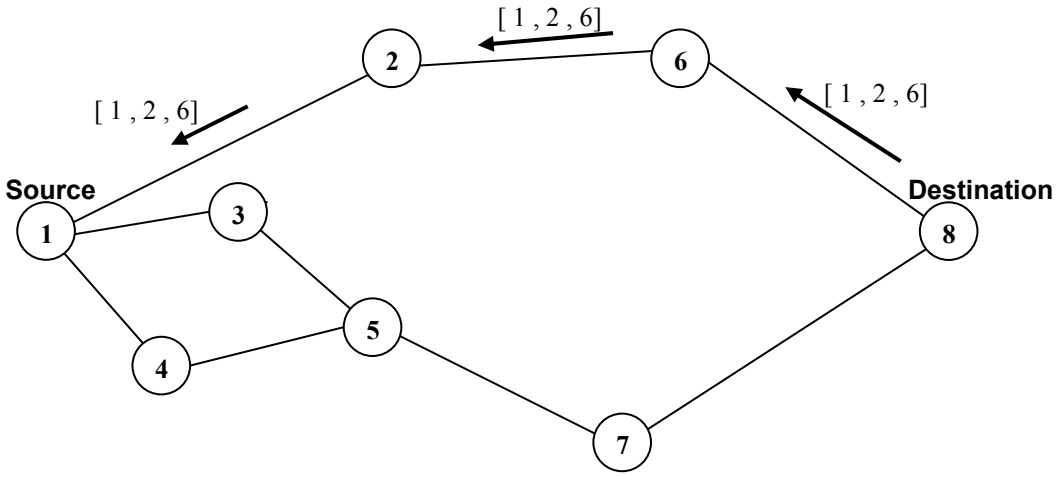
Lors de la réception d'un paquet *requête de route* par un nœud *p* du réseau, le traitement suivant est effectué :

1- Dans le cas où le couple < adresse de l'initiateur, identificateur de requête> du paquet reçu, existe déjà dans la liste des requêtes récemment reçues le paquet est ignoré.
2- Dans le cas contraire, si l'adresse de *p* existe dans le champ *enregistrement de route* du paquet de la requête, le paquet est ignoré.
3- Sinon, si l'adresse de *p* est la même que l'adresse de la destination, alors l'*enregistrement de route* (contenu dans le paquet de la requête) contient le chemin à travers lequel le paquet de la requête est passé avant d'atteindre le nœud *p*. Une copie de ce chemin est envoyée dans un paquet *réponse de route* à l'initiateur (voir la figure 3.8(b)).
4- Sinon, l'adresse de *p* est rajoutée dans l'*enregistrement de route* du paquet reçu, et le paquet est rediffusé (voir la figure 3.8(a)).

De cette manière, la requête de route est propagée dans le réseau, jusqu'à ce qu'elle atteigne l'hôte destination qui va répondre à la source. Le fait d'ignorer la requête, dans le cas où l'adresse du récepteur existe dans l'*enregistrement de route*, garantie que la propagation d'une unique copie de la requête ne peut pas se produire à travers des boucles de nœuds.



(a) Construction de l'*enregistrement de route*.



(b) Le renvoi du chemin.

Figure 3.8 : La découverte de chemins dans le DSR.

Dans le but de retourner le paquet *réponse de route* à l'initiateur de l'opération de découverte, l'hôte destination doit connaître un chemin vers l'initiateur. Dans le cas où la destination n'a pas déjà gardé une telle route, le chemin spécifié dans l'*enregistrement de route* contenu dans le paquet *requête de route* peut être inversé et utilisé (voir la figure 3.8(b)). Cependant, exige que les liens entre les nœuds participants dans le chemin doivent être bidirectionnels, ce qui n'est pas vérifié dans certains environnements. Dans [DSR article] l'approche de *piggybacking* est proposée pour ce genre de situation.

Afin de réduire le coût et la fréquence de la découverte de routes, chaque nœud garde les chemins appris à l'aide des paquets de réponses. Ces chemins sont utilisés jusqu'à ce qu'ils soient invalides.

Le protocole DSR n'intègre pas l'opération de découverte de routes avec celle de la maintenance, comme le fait les protocoles de routage conventionnels. Ces derniers intègrent les deux aspects précédents et s'adaptent aux changements de topologie du réseau par un échange périodique de messages de mise à jour. 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 *erreur de route* (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 paquet *erreur de route* 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.

Parmi les avantages - induites par l'utilisation de la technique "routage source" - du protocole DSR, le fait 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.

3.3.3 Le protocole de routage AODV

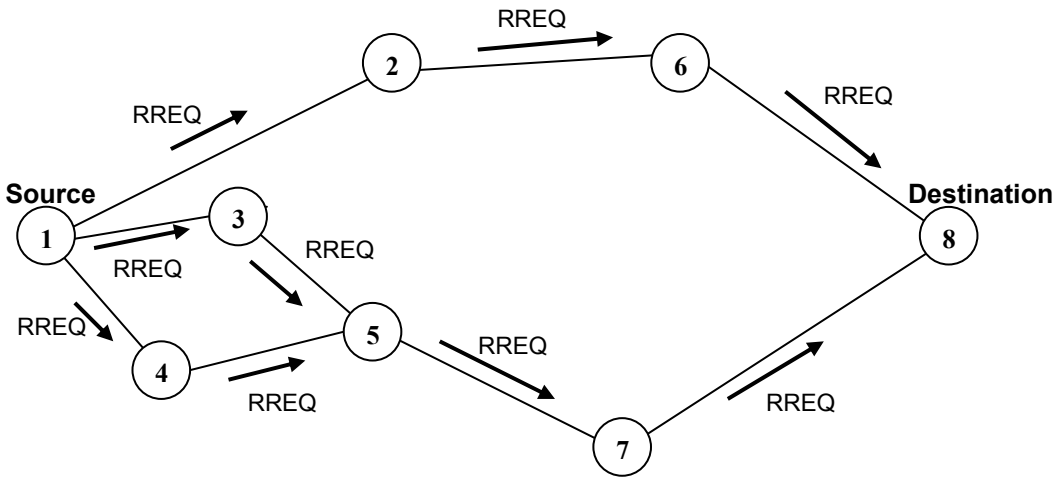
Le protocole "Routage avec Vecteur de Distance à la Demande" (AODV : Ad hoc On-demand Distance Vector) [PER 2000, PER 99], représente essentiellement une amélioration de l'algorithme DSDV discuté dans la section 3.2.1. Le protocole AODV, réduit le nombre de diffusions de messages, et cela en créant les routes lors du besoin, contrairement au DSDV, qui maintient la totalité des routes. L'AODV est basé sur l'utilisation des deux mécanismes "Découverte de route" et "Maintenance de route" (utilisés par le DSR), en plus du routage *nœud-par-nœud*, le principe des numéros de séquence et l'échange périodique du DSDV.

L'AODV utilise les principes des numéros de séquence à fin de maintenir la consistance des informations de routage. A cause de la mobilité des nœuds dans les réseaux ad hoc, les routes changent fréquemment ce qui fait que les routes maintenues par certains nœuds, deviennent invalides. Les numéros de séquence permettent d'utiliser les routes les plus nouvelles ou autrement dit les plus fraîches (fresh routes). De la même manière que dans le DSR, l'AODV utilise une *requête de route* dans le but de créer un chemin vers une certaine destination. Cependant, l'AODV maintient les chemins d'une façon distribuée en gardant une table de routage, au niveau de chaque nœud de transit appartenant au chemin cherché. Une entrée de la table de routage contient essentiellement :

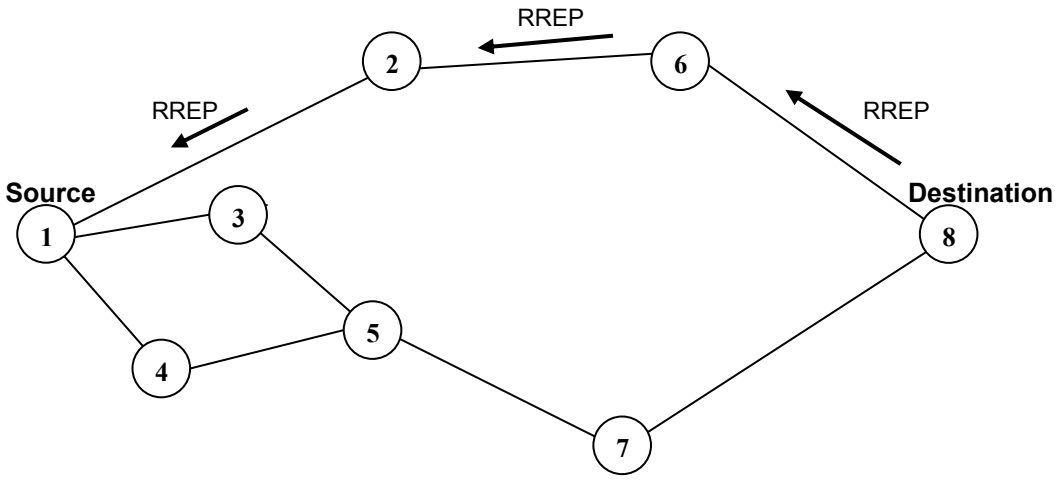
1- *L'adresse de la destination.*
2- *Le nœud suivant.*
3- *La distance en nombre de nœud (i.e. le nombre de nœud nécessaire pour atteindre la destination).*
4- *Le numéro de séquence destination.*
5- *Le temps d'expiration de l'entrée de la table.*

Quand un nœud de transit (intermédiaire) envoi le paquet de la requête à un voisin, il sauvegarde aussi l'identificateur du nœud à partir duquel la première copie de la requête est reçue. Cette information est utilisée pour construire le chemin inverse (figure 3.9(b)), qui sera traversé par le paquet *réponse de route* (cela veut dire que l'AODV supporte seulement les liens symétriques). Puisque le paquet *réponse de route* va être envoyé à la source, les nœuds appartenant au chemin de retour vont modifier leurs tables de routage suivant le chemin contenu dans le paquet de réponse.

Un nœud diffuse une *requête de route* (RREQ : Route REQuest), dans le cas où il aurait besoin de connaître une route vers une certaine destination et qu'une telle route n'est pas disponible (figure 3.9(a)). Cela peut arriver si la destination n'est pas connue au préalable, ou si le chemin existant vers la destination a expiré sa durée de vie ou il est devenu défaillant
(i.e. la métrique qui lui est associée est infinie). Le champ *numéro de séquence destination* du paquet RREQ, contient la dernière valeur connue du numéro de séquence, associé au nœud destination. Cette valeur est recopiée de la table de routage. Si le numéro de séquence n'est pas connu, la valeur nulle sera prise par défaut. Le *numéro de séquence source* du paquet RREQ contient la valeur du numéro de séquence du nœud source. Comme nous avons déjà dit, après la diffusion du RREQ, la source attend le paquet réponse de route (RREP : Route REPly). Si ce dernier n'est pas reçu durant une certaine période (appelée RREP_WAIT_TIME), la source peut rediffuser une nouvelle requête RREQ. A chaque nouvelle diffusion, le champ *Broadcast ID* du paquet RREQ est incrémenté. Si la requête RREQ est rediffusée un certain nombre de fois (RREQ_RETRIES) sans la réception de réponse, un message d'erreur est délivré à l'application.



(a) La propagation du paquet RREQ (requête de route).



(b) Le chemin pris par le paquet RREP (requête de réponse).

Figure 3.9 : Les deux requêtes RREQ et RREP utilisées dans le protocole AODV.

Afin de maintenir des routes consistantes, une transmission périodique du message "HELLO" est effectuée. Si trois messages "HELLO" ne sont pas reçus consécutivement à partir d'un nœud voisin, le lien en question est considéré défaillant. Les défaillances des liens sont, généralement, dû à la mobilité du réseau ad hoc. Les mouvements des nœuds qui ne participent pas dans le chemin actif, n'affecte pas la consistance des données de routage. Quand un lien, reliant un nœud *p* avec le nœud qui le suit dans le chemin de routage, devient défaillant, le nœud *p* diffuse un paquet *UNSOLICITED* RREP, avec une valeur de numéro de séquence égale à l'ancienne valeur du paquet RREP incrémentée de un, et une valeur *infinie* de la distance. Le paquet UNSOLICITED RREP est diffusé aux voisins actifs, jusqu'à ce qu'il arrive à la source. Une fois le paquet est reçu, la source peut initier le processus de la découverte de routes.

L'AODV maintient les adresses des voisins à travers lesquels les paquets destinés à un certain nœud arrivent. Un voisin est considéré actif, pour une destination donnée, s'il délivre au moins un paquet de donnée sans dépasser une certaine période (appelée *active timeout period*). Une entrée de la table du routage est active, si elle est utilisée par un voisin actif. Le chemin reliant la source et la destination en passant par les entrées actives des tables de routage, est dit un *chemin actif*. Dans le cas de défaillances de liens, toutes les entrées des tables de routage participantes dans le chemin actif et qui sont concernées par la défaillance sont supprimées. Cela est accomplit par la diffusion d'un message d'erreur entre les nœuds actifs.

Le protocole de routage AODV(et même le protocole DSR), n'assure pas l'utilisation du meilleur chemin existant entre la source et la destination. Cependant, des évaluations de performances récentes ont montré qu'il n'y a pas de grandes différences (en terme d'optimisation) entre les chemins utilisés par le protocole AODV et celles utilisées par les protocoles basés sur les algorithmes de recherche des

plus courts chemins [DAS 98]. En plus de cela, le protocole AODV ne présente pas de boucle de routage (une preuve de cela est détaillée dans [PER 99]), et évite le problème "counting to infinity" de Bellman-Ford, ce qui offre une convergence rapide quand la topologie du réseau ad hoc change.

3.3.4 Le protocole de routage TORA

L'Algorithme de Routage Ordonné Temporairement ou TORA (Temporary Ordering Routing Algorithme) [PAR 97, PAR 99] a été conçu principalement pour minimiser l'effet des changements de la topologie qui sont fréquents dans les réseaux ad hoc. L'algorithme s'adapte à la mobilité de ces environnements en stockant plusieurs chemins vers une même destination, ce qui fait que beaucoup de changements de topologie n'auront pas d'effets sur le routage des données, à moins que tous les chemins qui mènent vers la destination seront perdus (rompus). La principale caractéristique de TORA, est que les messages de contrôle sont limités à un ensemble réduit de nœuds. Cet ensemble représente les nœuds proches du lieu de l'occurrence du changement de la topologie.

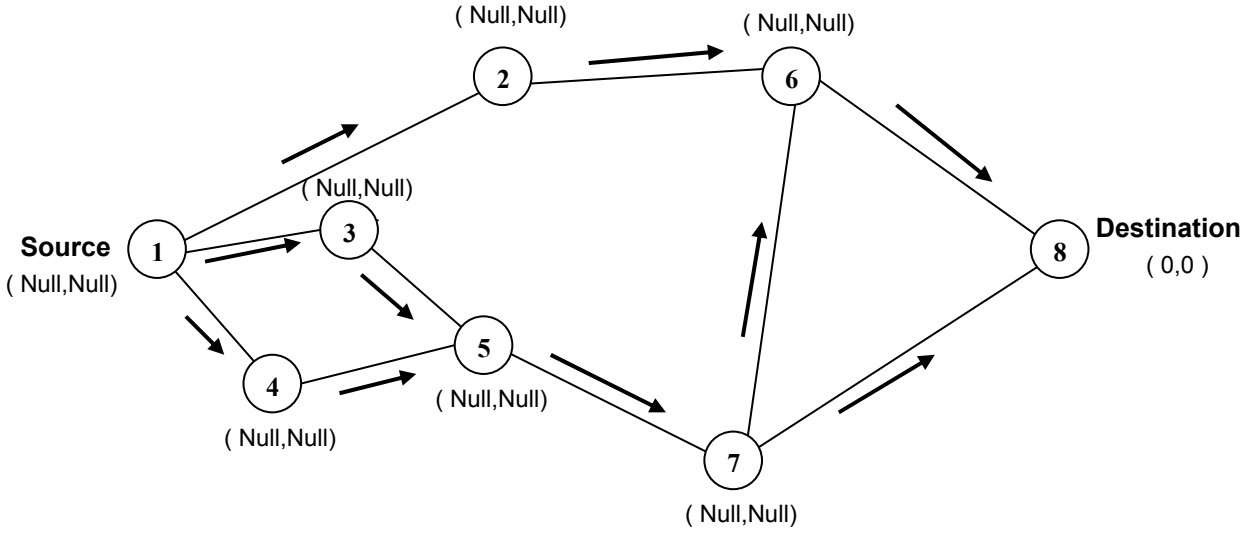
Dans ce protocole, la sauvegarde des chemins entre une paire (source, destination) donnée, ne s'effectue pas d'une manière permanente. Les chemins sont crées et stockés lors du besoin, comme c'est le cas dans tous les protocoles de cette catégorie. L'optimisation des routes (i.e. l'utilisation des meilleurs 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. Ce pouvoir d'initier et de réagir d'une façon non fréquente, sert à minimiser le temps de communication de contrôle utilisé pour découvrir continuellement le meilleur chemin vers la destination.

L'algorithme TORA appartient à la classe des algorithmes, appelée la classe "Inversement de Liens" (Link Revesral), qui est complètement différente des deux classes LS et Vecteur de Distance, vues déjà. TORA est basé sur le principe des algorithmes qui essaient de maintenir la propriété appelée "orientation destination" des graphes acycliques orientés (ou DAG : Directed Acyclic Graph) [GAF 81]. Un graphe acyclique orienté 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, les algorithmes utilisent le concept d'*inversement de liens*. Ce concept assure la transformation du graphe précédent, en un graphe orienté destination durant 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 qui l'échange avec l'ensemble de ses voisins directs. Cette nouvelle notion est utilisée dans l'orientation des liens du réseau. Un lien est toujours orienté du nœud qui a la plus grande taille, vers le nœud qui la plus petite taille. Les concepts de *taille* et d'*inversement de liens* sont orientés destination, cela veut dire que chaque nœud du réseau, exécute une copie logique indépendante de l'algorithme TORA pour chaque nœud destination.

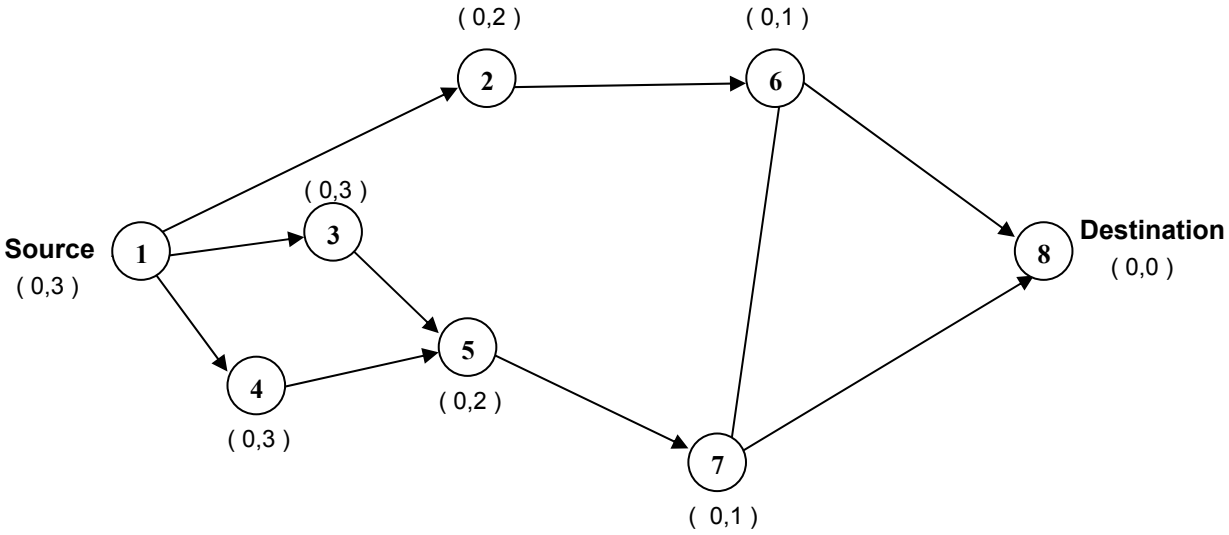
Le protocole TORA peut être divisé en quatre fonctions de base : création de routes, maintenance de routes, élimination de routes et optimisation de routes [PAR 99]. Chaque nœud *i* maintient un quintuplé qui lui est associé, ce dernier contient les

champs suivants : Le temps logique de défaillance (tau[*i*]), l'unique ID du nœud définissant le nouveau niveau de référence (oid[*i*]), un bit indicateur de réflexion (r[*i*]), le paramètre d'ordre de propagation (delta[*i*]), et l'unique ID du nœud (*i*).

Initié par la source, le processus de création (ou de découverte) de routes pour une destination donnée, crée un DAG orienté vers cette destination. L'algorithme commence dans l'état où : la *taille* (le paramètre d'ordre de propagation) de la source est initialisée à zéro, et la *taille* du reste des nœuds est indéfinie (i.e. égale à NULL). Le nœud source diffuse un paquet QRY (query) spécifiant l'identificateur de la destination, ID-destination, qui identifie le nœud pour lequel l'algorithme est exécuté. Un nœud qui a une taille indéfinie et qui reçoit le paquet QRY, rediffuse le paquet à ses voisins. Un nœud qui a une valeur de taille différente de NULL, répond par l'envoi d'un paquet UPD (update) qui contient sa propre taille. Lors de la réception du paquet UPD, le nœud récepteur affecte la valeur de taille contenant dans le paquet reçu plus un, à sa propre taille, à condition que cette valeur soit la plus petite par rapport à celles des autres voisins. Par exemple le nœud 6 de la figure 3.10(b), prend comme valeur de taille, la plus petite taille des voisins (i.e. la taille zéro qui correspond au nœud 8) plus un, ce qui donne la taille 1 (la même chose pour le nœud 1). De cette façon, un DAG est crée du nœud source vers le nœud destination.



(a) La propagation du paquet QRY.



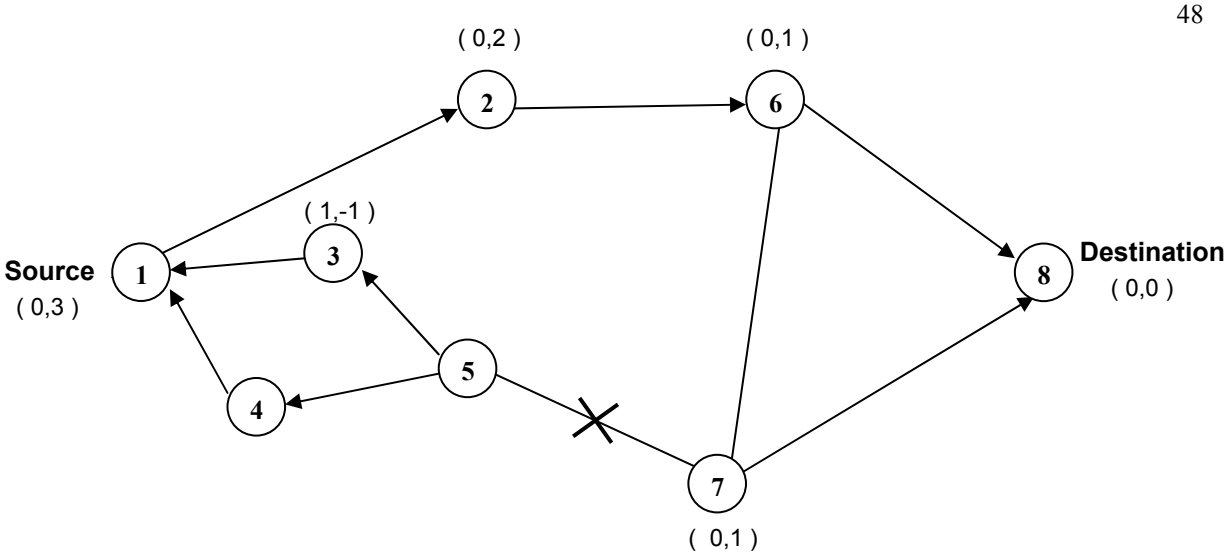
(x,y) : (le niveau de référence, la taille du nœud)

(b) Les tailles des nœuds après la réception du paquet UPD.

Figure 3.10 : La création des routes dans le protocole TORA.

La figure 3.10, montre la création de tel graphe dans le protocole TORA. Notons que les nœuds 5 et 7 reçoivent le paquet QRY deux fois, mais ils ne le diffusent qu'une seule fois. Dans la figure 3.10(b), le lien reliant les nœuds 6 et 7, n'est pas orienté car les tailles des deux nœuds sont égales.

A cause de la mobilité des nœuds dans les réseaux ad hoc, des routes du DAG peuvent être rompues, dans ce cas une maintenance de routes doit être effectuée afin de rétablir un DAG pour la même destination. Quand un nœud *i* détecte une défaillance (sachant qu'il ne possède pas de suivants valides vers la destination), il lance un *nouveau niveau de référence*, cela est effectué comme suit : le nœud *i* ajuste sa taille pour qu'elle représente le maximum de tailles des nœuds voisins. Le nœud *i*, transmet par la suite un paquet UPD contenant la nouvelle taille. Par conséquent tous les liens, vont être orienté du nœud *i* vers ses voisins, car la taille de *i* est devenue la plus grande taille. La diffusion du paquet UPD *inverse* le sens de tous les liens qui participent dans les chemins, où une défaillance est détectée; ce qui indique à la source l'invalidité des chemins rompus (figure 3.11).



(x,y) : (le niveau de référence, la taille du nœud)

Figure 3.11 : La réaction du protocole TORA de la défaillance du lien (5,7).

La fonction de suppression du protocole TORA est effectuée en diffusant un paquet CLR (clear) dans le réseau, et cela afin de supprimer les routes invalides qui sont sauvegardées localement par les nœuds du réseau. Cela est fait, par exemple, dans le cas de détection de partitions.

3.3.5 Le protocole de routage ABR

Le protocole "Routage Basé sur l'Associativité" (ABR : Associativity Based Routing) introduit dans [TOH 96, TOH 99], représente une nouvelle approche de routage pour les réseaux mobiles ad hoc. Le protocole défini une nouvelle métrique de routage appelée *degré de stabilité d'association*, et évite les problèmes : formation des boucles de routage, blocage et duplication de paquets. Dans l'ABR, le chois des routes est basé sur ce qu'on appelle *les états d'associativité des nœuds*. Les nœuds du réseau ad hoc génèrent périodiquement des signaux de contrôle afin de montrer leur existence par rapport aux autres nœuds. Quand un nœud reçoit un tel signal, il met à jour ses tables d'associativité. Pour chaque signal reçu, un nœud incrémente son intervalle d'associativité qui correspond au nœud émetteur du signal. La stabilité d'association représente la stabilité de la connexion existante entre deux nœuds, en se basant sur les deux paramètres temps et localisation. Une grande valeur de l'intervalle d'associativité, correspondante à un nœud voisin, indique *un état bas de mobilité* de ce nœud. Une petite valeur de cet intervalle indique *un état haut de mobilité* du voisin. Les intervalles d'associativité sont reinitialisés, quand les voisins d'un nœud, (ou le nœud lui-même) se déplacent et perdent la connexion.

L'objectif principal du protocole ABR, est de trouver des chemins de longue durée de vie pour les réseaux dynamiques. Le protocole ABR consiste en trois phases principales : la découverte des routes, la reconstruction des routes (RRC), et la suppression des routes.

La phase de découverte des routes, représente un cycle de diffusion de requête et d'attente de réponse (BQ-REPLY). Le nœud source diffuse un message BQ (Broadcast Query) afin de trouver les nœuds qui mènent vers la destination. Un nœud fait transiter le BQ reçu, au plus une fois. Un nœud de transit (ou intermédiaire), rajoute son adresse et ses intervalles d'associativité au paquet de la requête. Le nœud suivant dans le chemin, ne maintient que l'intervalle d'associativité qui lui est associé et celui du nœud précédent dans le chemin. De cette manière, chaque paquet qui arrive à la destination, va contenir les intervalles d'associativité des nœuds qui appartiennent au chemin reliant la source et la destination. Le nœud destination peut donc, choisir le meilleur chemin en examinant les intervalles d'associativité qui existent dans chaque chemin. Si plusieurs chemins ont le même degré de stabilité d'association, le chemin ayant le plus petit nombre de nœuds (i.e. le chemin le plus court) est choisi. Une fois cela est fait, le nœud destination envoi un paquet de réponse (appelé REPLY), au nœud source en utilisant le chemin choisi. Les nœuds qui appartiennent au chemin suivi par le paquet REPLY, marquent que leurs routes sont valides, le reste des routes reste inactif.

La phase de reconstruction de routes (RRC), consiste en une découverte partielle de routes, une suppression de routes invalides, une mise à jour de routes valides, et enfin une nouvelle découverte de routes, et cela suivant le cas du nœud qui a causer le mouvement d'une route [TOH 96, TOH 99]. Le mouvement du nœud source, implique un nouveau processus BQ-REPLY, car le protocole de routage est *initié-source*. Un message de notification de route RN (Route Notification), est utilisé dans le but d'éliminer les routes, des nœuds suivants dans le chemin. Si le nœud source est la source du mouvement, les nœuds qui le précèdent dans le chemin, suppriment la route invalide correspondante. Le protocole utilise aussi, un processus de requête de localisation (LQ[h]), pour déterminer si un nœud voisin - de rang h dans le chemin source/destination - peut être toujours atteint ou pas. Si la destination reçoit le paquet LQ, elle sélectionne le meilleur chemin partiel existant, et l'envoie dans un paquet de réponse REPLY. Dans le cas contraire, le nœud qui a initié le processus de localisation attend jusqu'à l'expiration de sont timeout, et relance par la suite le même processus pour le voisin suivant. Si le processus de localisation LQ échoue pour tous les voisins, en un certain nombre de fois; la source initie un nouveau processus BQ.

Quand un chemin trouvé devient non utilisé par une certaine source, une diffusion d'élimination de route RD (Route Delete) est lancée. Tous les nœuds qui appartiennent au chemin non utilisé, suppriment les entrées correspondantes de leurs tables de routage. La diffusion du message d'élimination de routes, doit être faite d'une manière globale pour supprimer toutes les routes qui pouvaient être construites suite à une phase de reconstruction des routes.

3.3.6 Le protocole de routage SSR

Le protocole "Routage basé sur la Stabilité du Signal" (SSR : Signal Stability-based Routing) proposé dans [DUB 97], est un protocole de routage réactif dont le chois des routes est basé sur la puissance du signal entre les nœuds, en plus de leur stabilité de

localisation. Ce critère de sélection de routes, fait que les chemins utilisés durant le routage des données, ont une forte interconnexion.

Le protocole SSR inclut deux protocoles qui coopèrent entre eux : le Protocole de Routage Dynamique appelé DRP (Dynamic Routing Protocol), et le Protocole de Routage Statique appelé SRP (Static Routing Protocol). Le premier protocole, le DRP, utilise deux tables : une table de stabilité de signal SST (Signal Stability Table), et une table de routage RT. La table SST sauvegarde les puissances des signaux des nœuds voisins, obtenues par l'échange périodique des messages avec la couche de liaison de chaque voisin. La puissance d'un signal est sauvegardée sous l'une de ces deux formes : "*canal de forte puissance*" ou "*canal de faible puissance*".

Toutes les transmissions sont reçues et traitées par le DRP. Après la mise à jour de l'entrée appropriée de la table, le protocole DRP fait passer le paquet traité au protocole SSR. Le SSR consulte sa table de routage RT pour la destination spécifiée, et envoie le paquet reçu au voisin suivant. Si aucune entrée (dans la RT) associée au nœud destination n'est disponible, le SSR initie un processus de recherche de routes en diffusant un paquet *requête de route*. Le paquet requête de route est envoyée une seule fois (pour éviter le bouclage), et uniquement aux voisins vers lesquels existe un lien de forte puissance. Le nœud destination choisit le premier paquet *requête de route* qui arrive. Car il y a une grande probabilité pour que ce paquet ait traversé le meilleur chemin (le plus court, le moins chargé … etc.) existant antre la source et la destination. Le DRP du nœud destination inverse le chemin choisi, et envoi un message de réponse de route au nœud source. Lors de la réception de cette réponse, le DRP d'un nœud intermédiaire met à jour la table de routage locale, suivant le chemin inclus dans le paquet reçu.

Les paquets de recherche de routes qui arrivent à la destination, prennent nécessairement le chemin de forte stabilité de signal; car les nœuds de transit n'envoient pas de paquets à travers les liens de faible puissance de signal. Si la source expire son timeout sans la réception de réponse, elle relance de nouveau, un processus de recherche de routes en indiquant cette fois ci que les canaux de faibles puissances peuvent être utilisées.

Quand une défaillance de liens est détectée dans le réseau, le nœud détectant envoie un message d'erreur au nœud source, en spécifiant le lien défaillant. Lors de la réception de ce message, la source envoie *un message de suppression* pour avertir tous les nœuds, de la défaillance du lien en question. La source initie par la suite un nouveau processus de recherche de routes, dans le but de trouver un nouveau chemin vers la destination.

3.3.7 Le protocole de routage LAR

Le protocole appelé "Routage aidé par la localisation" ou LAR (Location-Aided Routing) [KO 98], est un protocole de routage réactif basé sur l'utilisation des localisations. Ce protocole procède d'une manière très similaire au protocole DSR vu dans la section 3.3.2. 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 appelé GPS (Global Positioning System), 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 définie une région circulaire dans laquelle la destination peut être localisée. La position et la taille de la région, sont estimées en se basant sur :

1- *La position de la destination, telle qu'elle est connue par la source.*
2- *L'instant qui correspond à cette position.*
3- *La vitesse moyenne du mouvement de la destination.*

Le plus petit rectangle couvrant la région circulaire et le nœud source, est appelé *la zone de requête*. L'information calculée, est rattachée au paquet de requête de route. Cela est fait uniquement par le nœud source, et les nœuds qui appartiennent à la zone de requête.

Dans la deuxième approche, le nœud source calcule la distance qui lui sépare de la destination, et l'inclut dans le paquet de requête de route. Ce dernier est envoyé par la suite aux nœuds voisins. Quand un nœud reçoit le paquet de requête, il calcule la distance qui lui sépare de la destination, et la compare avec la distance contenue dans le paquet reçu. Dans le cas où la distance calculée est inférieure ou égale à la distance reçue, le nœud envoie le paquet reçu. Lors de l'envoi, le nœud met à jour *le champ de distance* avec sa propre distance qui lui sépare du nœud destination.
Dans les deux méthodes, si aucune réponse de route n'est reçue en dépassant une certaine période (i.e. le timeout), le nœud source rediffuse une nouvelle *requête de route* en utilisant une diffusion pure (i.e. une diffusion sans limitation).

3.3.8 Le protocole de routage RDMAR

Le protocole de Routage basé sur la Micro découverte des Distances Relatives ou RDMAR (Relative Distance Micro-discovery Ad hoc Routing) [AGG 99] est un protocole réactif conçu principalement pour s'adapter aux changements rapides des réseaux ad hoc en réduisant le contrôle utilisé. Un des principales caractéristiques de ce protocole, est que la réaction aux défaillances des liens est limitée à une petite région, qui se trouve proche du lieu du changement dans le réseau. Cela est assuré grâce à l'utilisation d'un nouveau mécanisme de découverte de routes, appelé *la Micro-découverte de Distance Relative* ou RDM (Relative Distance Micro-discovery). L'idée de base du RDM est que la diffusion des requêtes, peut se faire en se basant sur une distance relative (RD) entre les paires des unités mobiles. Afin de réaliser cela, une recherche de routes est déclenchée, chaque fois, entre deux nœuds du réseau. Un algorithme itératif est utilisé pour estimer la RD qui sépare les deux nœuds, et cela en utilisant les informations concernant la mobilité des nœuds, le temps écoulé depuis la dernière communication et l'ancienne valeur de la distance RD. Sur la base de la nouvelle distance calculée, la diffusion de requête est limitée à une certaine région du réseau dans laquelle la destination peut être trouvée. Cette limitation de diffusion, peut minimiser énormément le contrôle du routage, ce qui améliore les performances de la communication. Comme nous avons déjà vu, les protocoles LAR et DREAM (sections 3.2.8 et 3.3.7) visent aussi à réduire la zone de propagation de requêtes.

Cependant, ces protocoles sont basés sur l'utilisation du système de positionnement global GPS, cet outil qui n'est pas toujours disponible pour tous les utilisateurs mobiles.
Dans le protocole RDMAR, les données sont acheminées entre les nœuds du réseau en utilisant des tables de routage stockées au niveau de chaque nœud. Chaque table de routage contient la liste des nœuds destinations qui peuvent être atteints. Une entrée qui est associée à une destination donnée, contient les informations suivantes :

1- *Le routeur par défaut* : qui est un champ indiquant le nœud suivant à travers lequel le nœud courant peut atteindre la destination.
2- *Un champ RD* : qui donne la distance estimée entre le nœud et la destination.
3- *Le temps de la dernière mise à jour* : appelé TLU (Time Last Update), qui représente l'instant de la dernière réception des informations de routage qui proviennent de la destination.
4- *Un champ appelé "RT_Timeout"* : qui contient le temps représentant la durée de vie de la route, i.e. la durée après laquelle la route sera considérée invalide.
5- *Un champ appelé "Route Flag"* : qui précise si la route, correspondante à la destination, est activée ou non.

Le RDMAR comprend deux algorithmes principaux : *l'algorithme de Découverte de Route* qui est responsable de trouver - si c'est nécessaire - les chemins, et *l'algorithme de Maintenance de routes* dont le rôle est de détecter les changements de la topologie du réseau et de vérifier la validité des chemins utilisés.

Quand un nœud reçoit un appel pour une certaine destination *j*, sachant qu'il n'existe pas de routes disponibles vers cette destination, le nœud *i* initie une phase de découverte de routes. Ici le nœud a deux options : soit de diffuser la requête de route et cela dans le réseau entier; ou bien de limiter la découverte à une petite région du réseau si un certain modèle de prédiction de localisation, peut être établi pour la destination *j*. Dans ce dernier cas, le nœud source ou l'initiateur de la phase de découverte de routes, se réfère à sa table de routage pour extraire l'ancienne distance relative et le temps écoulé depuis la dernière réception des informations de routage, qui provient du *j*. Le nœud source calcule en utilisant les informations extraites, la nouvelle distance relative qui lui sépare de la destination. Le calcul fait est très simple, il se base sur l'utilisation de la formule suivante :

déplacement (*unité mobile*) = vitesse_ moyenne (*unité mobile*) * temps.

Le nœud *i* limite la distribution des requêtes de route en insérant une valeur normalisée [AGG 99a, AGG 99b] de la nouvelle distance relative (RDM_Radius), dans le champ TTL [STE 97] de la tête du paquet requête de route RREQ. Cette procédure est appelée *la Micro-découverte de distance relative* (RDM).

Notons que la nouvelle distance qui existe entre la source et la destination après l'écoulement d'un certain temps, ne s'obtient pas toujours en rajoutant les déplacements calculés à l'ancienne valeur de la distance. Comme montre la figure 3.12, durant l'intervalle de temps *t*, le nœud source (respectivement destination) peut être trouvé à n'importe quel point du cercle ayant comme rayon la valeur déplacement_SRC (respectivement déplacement_DST). Par conséquent, la nouvelle distance relative

maximale, est égale à l'ancienne distance relative plus deux fois le déplacement calculé. La distance relative minimale est égale à la valeur absolue de la différence entre l'ancienne distance relative, et deux fois le nouveau déplacement.

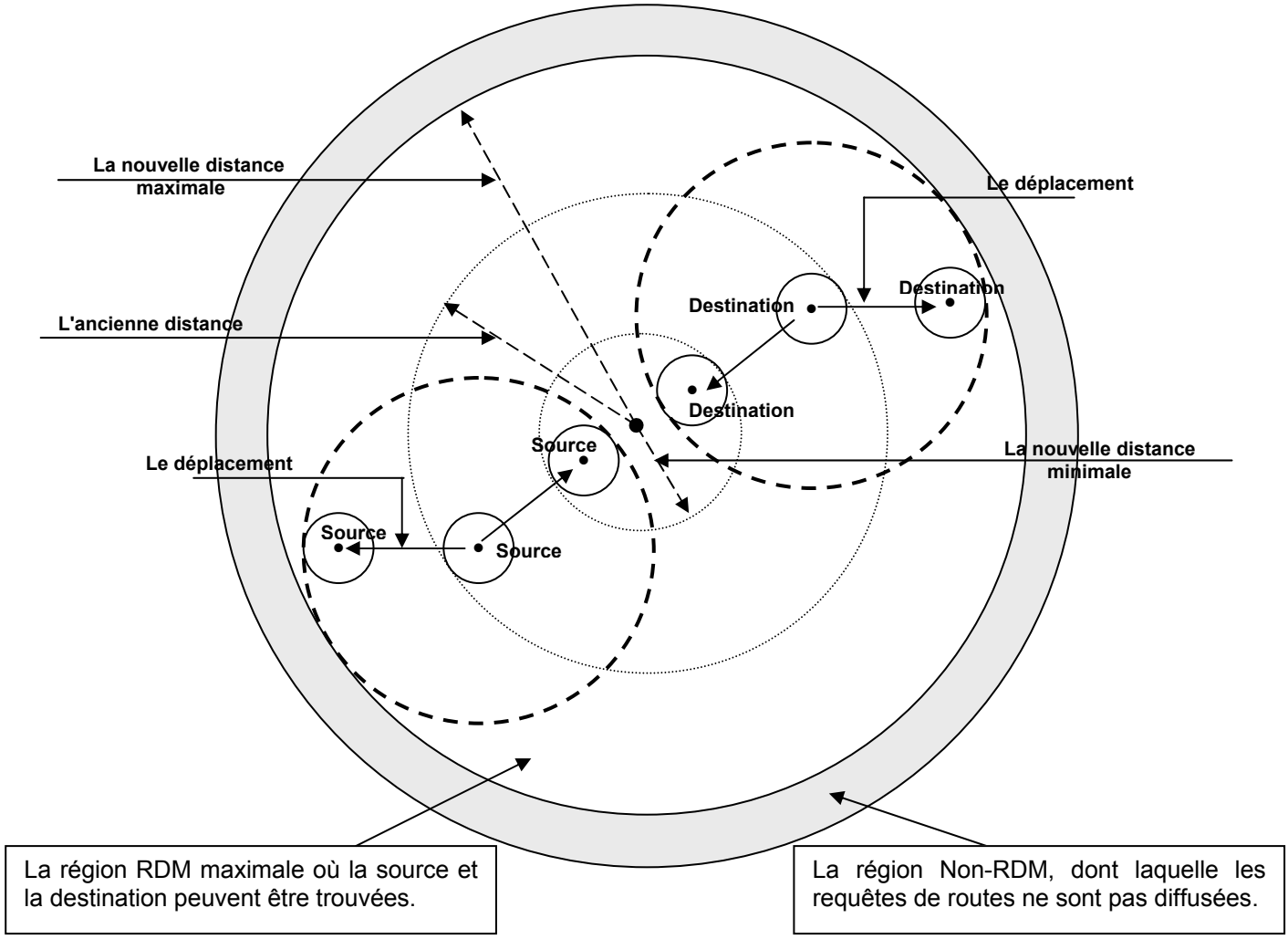


Figure 3.12 : La procédure RDM du protocole RDMAR.

Dans le protocole RDMAR, la décision du choix de chemin est prise au niveau du nœud destination. Seulement le meilleur chemin choisi sera valide, les autres chemins reste passifs. Quand un nœud intermédiaire *i* reçoit un paquet de données, il traite d'abord la tête du paquet et envoie par la suite le paquet au nœud suivant. En plus de cela, le nœud *i* envoie un message explicite au nœud précédent afin de tester si le lien de communication qui existe entre les deux nœuds, est bidirectionnel ou non. De cette manière, les nœuds qui envoient le paquet de données, peuvent avoir les informations de routage nécessaires pour l'envoi des acquittements au nœud source. Si un nœud *i* est incapable d'envoyer le paquet de données à cause de l'indisponibilité de routes, ou à cause des erreurs rencontrées le long du chemin (défaillance de liens ou de nœuds); le nœud essaie de retransmettre le paquet, un certain nombre de fois. La raison de ces multiples essais, est que la défaillance pouvait être causée par des facteurs temporaires

(par exemple, des bruits de signal). Cependant si le problème persiste, deux cas peuvent exister et cela suivant la distance relative qui existe entre le nœud *i* et la destination.

Si à l'instant de la défaillance, le nœud *i* trouve - en utilisant sa table de routage - qu'il est proche du nœud destination, il initie une procédure RDM telle qu'elle est décrite précédemment. Dans le cas contraire, i.e. le nœud *i* est proche de la source du paquet de donnée, le nœud avertit la source de la défaillance en lui délivrant le paquet. Ce dernier cas est appelé : *la phase d'avertissement de défaillance*.

Durant la phase d'avertissement de défaillance, chaque nœud *i* qui reçoit un paquet à renvoyer, maintient une liste de tous les voisins pour lesquels le nœud *i* représente le routeur par défaut correspondant à la destination *j*. Cette liste, appelée *la liste dépendante*, est utilisée pour envoyer les avertissements de défaillance seulement aux nœuds qui utilisent le chemin défaillant. De cette manière, les nœuds qui ont besoin de routes, peuvent chercher de nouveaux chemins. Le message d'avertissement de défaillance traverse les nœuds de transit jusqu'à ce qu'il arrive au nœud source. Un nœud qui reçoit le message d'avertissement, doit supprimer de sa table de routage, la route correspondante à la destination, dans le cas où le message est reçu à partir du nœud suivant associé à la destination.

Le RDMAR propose deux optimisations dans le cas de défaillances. La première optimisation consiste à utiliser une technique appelée "*la technique Cranback de défaillance*". Cette technique est similaire à celle proposée dans [ATM 96]. Dans cette technique, un nœud *i* qui reçoit un paquet de donnée, garde d'abord une copie temporaire du paquet avant de l'envoyer. si *i* reçoit un message d'erreur concernant le même paquet, *i* transmet le paquet en suivant un autre chemin - s'il existe - au lieu de transmettre un message d'erreur au nœud source.

La deuxième optimisation, concerne le cas où un nœud *i*, reçoit un paquet destiné à *j* sachant que le nœud suivant , par exemple *k*, associé à *j* ne peut pas être atteint. Dans ce cas, le nœud *i* peut envoyer un avertissement d'erreur à tous les nœuds sources, dont le nœud *i* est participant dans leurs chemins actifs vers une destination *l*; sachant que *l* ne peut pas être atteinte à cause de la défaillance du lien (*i*,*k*). Cette deuxième optimisation est en cours de test.

3.4 Conclusion

Dans ce chapitre nous avons présenté plusieurs protocoles de routage qui ont été proposé pour assurer le service de routage dans les réseaux mobiles ad hoc. Nous avons décrit leurs principales caractéristiques et fonctionnalités afin de comprendre les stratégies utilisées dans l'acheminement des données entre les différentes unités mobiles.

Assurer la connexion de tous les nœuds d'un réseau ad hoc est un problème très complexe vu la dynamicité et l'évolution rapide de la topologie, en effet les unités mobiles sont dynamiquement et arbitrairement éparpillés d'une manière où l'interconnexion peut changer à tout moment. Le but d'un protocole de routage est

donc, l'établissement de routes qui soient correctes et efficaces entre une paire quelconque d'unités.

Comme nous avons vu, les protocoles proposés sont classés en deux catégories : les protocoles pro-actifs et les protocoles réactifs. Les protocoles des deux catégories essaient de s'adapter aux contraintes imposées par le réseau ad hoc, et cela en proposant une méthode qui soit de moindre coût en capacités et ressources, et qui garantit la survabilité du routage en cas de n'importe quelle panne de lien ou de nœuds. Les protocoles de routage étudiés offrent différents avantages qui sont en réalité complémentaires et préférables pour différents types d'applications.

La conclusion générale qu'on peut tirer de l'étude des différentes stratégies, est que la conception d'un protocole de routage pour les réseaux ad hoc doit tenir compte de tous les facteurs et limitations physiques imposés par l'environnement afin que la stratégie de routage ne dégrade pas les performances du système.