Les protocoles de routage dans les réseaux mobiles Ad Hoc

Nadjib Badache[1], Djamel Djenouri[2], Abdelouahid Derhab[3], Tayeb Lemlouma[4]
Laboratoire des logiciels de base CERIST
***E-mail:** 1badache@wissal.dz , 2 djenouri@hotmail.com, 3 derhabos@hotmail.com, 4 lemlouma@inrialpes.fr*

1- Introduction

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 à 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 ou 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.

A cause des limitations des réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante et de l'énergie. 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.

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.

2- Les protocoles de routage pro-actifs

Les protocoles de routage pro-actifs pour les réseaux mobiles ad-hoc, sont basés 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 méthode *Etat de Lien (Link State)* et la méthode du *Vecteur de Distance (Distance Vector)*. 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.

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 le sépare d'une destination donnée, en se basant sur l'image complète du réseau formé 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 Shortest Path First), 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.

Dans l'approche de routage "Distance Vector", chaque nœud de routage diffuse à ses nœuds de routage voisins, sa vision des distances qui le séparent de tous les hôtes du réseau. En se basant sur les informations reçues depuis 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].

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 considération, les caractéristiques du nouvel environnement.

2.1 Destination Sequenced Distance Vector (DSDV) [PER 94]

Le protocole de routage de vecteur de distance ordonnancé par destination est dérivé d'un algorithme classique de vecteur de distance, l'algorithme distribué de Bellman-Ford (DBF). Des perfectionnements sont faits afin d'éviter le problème des boucles présentes dans DBF. Ceci est évité en étiquetant chaque entrée de table de routage avec un numéro de séquence pour commander l'information de routage.

Dans DSDV, chaque nœud maintient une table de routage qui a une entrée pour chaque destination dans le réseau. Les attributs pour chaque destination sont le prochain saut, le nombre de sauts, et un numéro de séquence qui est envoyé par le nœud destinataire. Pour maintenir l'uniformité des tables de routage, DSDV utilise deux types de mise à jour: des mises à jour périodiques et d'autres basées sur les évènements afin de propager l'information de routage aussi rapidement que possible quand il y a n'importe quel changement topologique. Les paquets de mise à jour incluent les destinations accessibles de chaque nœud et le nombre de sauts exigés pour atteindre chaque destination avec le numéro de séquence lié à chaque route.

Lors de réception d'un paquet de mise à jour, chaque nœud le compare avec l'information existante concernant la route. Des routes avec des anciens numéros de séquence sont simplement ignorées. En cas de route

avec le numéro de séquence égal au numéro de séquence de la route annoncée, on remplace l'ancienne si elle a une meilleure métrique. La métrique est alors incrémenté par un saut puisque le paquet exige un saut de plus pour atteindre la destination. Des routes nouvellement enregistrées sont immédiatement annoncées à ses voisins.

Quand un lien au prochain saut est invalide, toute route contenant ce prochain saut est immédiatement assigné à une métrique infinie et son numéro de séquence est mis à jour. C'est le seul cas où les numéros de séquence sont assignés par un nœud autre que la destination. Quand un nœud reçoit une métrique infinie, et il a un numéro de séquence supérieur ou égal avec une métrique finie, un envoi de mise à jour de route est déclenché. Par conséquent, des routes avec une métrique infinie seront rapidement remplacées par des routes réelles propagées.

DSDV utilise également un mécanisme pour amortir des fluctuations dans des mises à jour de table de routage. Dans un environnement où beaucoup de nœuds indépendants transmettent l'information de routage asynchrone, une certaine fluctuation pourrait se développer. Par exemple, un nœud pourrait recevoir deux routes à la même destination avec le même numéro d'ordre, cependant, celui avec la métrique la plus mauvaise arrive toujours en premier. Ceci a pu mener aux excès des mises à jour de route et des fluctuations des tables de routage. DSDV résout ce problème en employant des données " de temps de stabilisation ".
Spécifiquement, la durée de temps jusqu'à ce que la route devienne stable (nommé temps de stabilisation) est prévue, et le temps de stabilisation doit être fini avant d'annoncer n'importe quelle nouvelle information de route au réseau.
En d'autres termes, le temps de stabilisation est utilisé comme moyen de décider combien de temps il faut attendre avant d'annoncer de nouvelles routes.

Un des avantages principaux de DSDV est qu'il fournit les routes sans boucles (loop free) à tout instant. Cependant, il a un certain nombre d'inconvénients.
Il est difficile à déterminer des valeurs optimales pour les paramètres comme le temps de stabilisation maximum pour une destination particulière. Ceci

pourrait conduire à des fluctuations et de fausses annonces ayant pour résultat le gaspillage de bande passante. DSDV emploie également des mises à jour périodiques et évènementielles, qui pourraient causer des messages de contrôle excessifs de communication. En outre, dans DSDV, un nœud doit attendre jusqu'à ce qu'il reçoive la prochaine mise à jour de route lancée par la destination avant qu'il puisse mettre à jour son entrée de la table de routage pour cette destination.. En outre DSDV ne supporte pas le routage multichemin.

2.2 Wireless Routing Protocol (WRP) [MUR 96]

Le protocole de routage sans fil (Wireless routing protocol) est également basé sur l'algorithme de vecteur de distance. Pour éviter le problème de boucle présent dans l'algorithme de vecteur de distance, WRP inclut l'avant dernier nœud (prédécesseur) pour chaque destination. Cette information de prédécesseur est incluse dans quatre tables maintenues au niveau de chaque nœud i:

1)-une table de distance pour chaque nœud j qui contient les champs suivants :

D_{jk} : distance entre les nœuds i et j, tel que k est le prochain saut pour atteindre j.

P_{jk} :prédécesseur de j dans le chemin entre i et j, tel que k est le prochain saut pour atteindre j.

2)-table de routage qui contient les champs suivants:

- l'adresse de la destination
- D_j :distance entre i et j
- P_j : prédécesseur qui correspond au chemin minimum pour atteindre j
- S_j : successeur qui correspond au chemin minimum pour atteindre j

-une marque tag_j =correct :chemin simple
=error :boucle
=null :destination n'est pas marquée

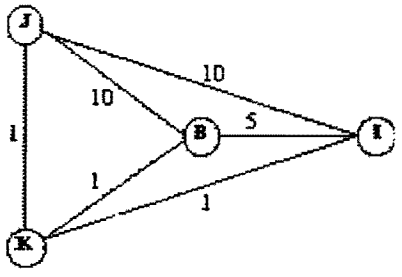
3)-table des coûts de liens qui contient les champs suivants:

- l_k :le coût pour chaque voisin k
- la fréquence de mise à jour

Le coût d'un lien défaillant est considéré comme étant infini.

4)- liste des retransmissions : permet de connaître l'ensemble des voisins qui n'ont pas acquitté leur message de mise à jour et de retransmettre ce message à cet ensemble.

Les tableaux b) et c) de la figure 1 montrent la table de distance et la table de routage au nœud I, respectivement, pour le réseau représenté sur la même figure



a)

Destination/ Voisin	B	K	J
B	(5,I)	(2,k)	(12, k)
K	(6,b)	(1,i)	(11, j)
J	(7,k)	(2,k)	(10, i)

b) Tableau de distance au nœud I

Destination	Prochain saut	Distance	Prédécesseur
B	K	2	K
J	K	1	I
J	K	2	K

c) Tableau de routage au nœud I

Figure 1

Pour garder une vue consistante du réseau et pour répondre aux changements topologiques, chaque nœud échange des paquets de mise à jour avec ses voisins. Une entrée de mise à jour indique une destination, une distance à la destination, et un prédécesseur à la destination. Lors de la réception d'un paquet de mise à jour, chaque nœud met à jour la distance et les entrées de prédécesseur dans la table de distance. Par exemple quand un nœud i reçoit une mise à jour de son voisin k concernant la destination j, 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.

Notez que par le retour en arrière de l'information de prédécesseur maintenue dans la table de distance, le chemin complet peut être trouvé. Par exemple, en regardant le deuxième tableau, on peut indiquer le chemin complet du nœud I au nœud J dont la distance est 2 est I-k-j. Après mise à jour de la table de distance, chaque nœud met à jour également sa table de routage en conséquence en choisissant le voisin qui offre le plus petit coût à la destination donnée comme prochain saut pour arriver à cette destination.

L'avantage principal de WRP est qu'il réduit la formation des boucles provisoires en employant l'information de prédécesseur pour identifier la route. Ceci réduit alternativement le temps de convergence. Cependant, comme dans tous les protocoles proactive, dans WRP, chaque nœud maintient constamment des informations de routage complètes à toutes les destinations dans le réseau, exigeant des messages de contrôle importants.

2.3 Global State Routing (GSR) [CHE 98]

Il est similaire au DSDV, en plus de ça, il utilise les principes de routage de LS (Link State). Il améliore ces deux algorithmes en évitant l'inondation des messages de routage, et il utilise aussi une vue globale de la topologie et la méthode de dissémination de DBF, qui implique l'absence d'inondation. Chaque nœud i contient les informations suivantes:

- une liste des voisins A_i
- une table de topologie TT_i
- une table des nœuds suivants pour chaque destination $NEXT_i$
- table de distance : qui contient pour chaque destination la distance minimale.

Lors de la réception d'un message de routage, le nœud met à jour sa TT_i ; si le numéro de séquence (*NS*) du message est supérieur au *NS* de la table, il diffuse une mise à jour à ses voisins.

GSR utilise les algorithmes de recherche de plus court chemin et maintient la table la plus récente reçue à partir des voisins d'une façon périodique. Il assure plus de précision concernant les données qui s'échangent dans le réseau.

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 des états de liens sont générés et diffusés par inondation dans tout le réseau. Par contre, GSR maintient la table la plus récente des états de 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.
Cependant GSR a toujours les mêmes inconvénients que tous les protocoles basés sur LS.

2.4 Fisheye State Routing (FSR) [PEI 00]

Le routage d'état d'œil de poisson (FSR) est une amélioration de GSR, il est basé sur l'utilisation de la technique «œil de poisson ». La grande taille des messages de mise à jour dans GSR gaspille une quantité considérable de la bande passante du réseau. Dans FSR, un message de mise à jour ne contient pas l'information sur tous les nœuds. Au lieu de cela, il échange des informations sur les nœuds les plus proches plus fréquemment qu'il le fait sur les nœuds les plus lointains, réduisant ainsi la taille de message de mise à jour. Donc chaque nœud obtient l'information précise concernant les voisins. Le détail et l'exactitude de l'information diminuent quand la distance du nœud augmente. La figure 2 définit la portée du fisheye pour le nœud (11) central. La portée est définie en termes des nœuds qui peuvent être atteints en un certain nombre de sauts. Le nœud central a l'information la plus précise sur tous les nœuds dans le cercle blanc et ainsi de suite. Quoiqu'un nœud n'ait pas l'information précise au sujet des nœuds éloignés, les paquets sont conduits correctement parce que l'information de route devient de plus en plus précise quand le paquet se rapproche de la destination.

Puisque la technique "œil de poisson" est réalisée, dans le protocole FSR, en se basant sur le changement des périodes. Les paquets de mise à jour, arrivent lentement aux nœuds qui sont loin de la source. Cependant, la connaissance imprécise du meilleur chemin vers une destination distante, est compensée par le fait que la route devient progressivement plus précise quand les paquets s'approchent peu à peu de la destination.

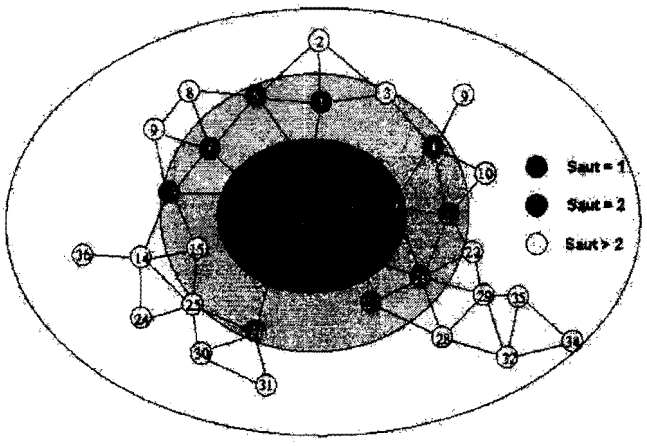
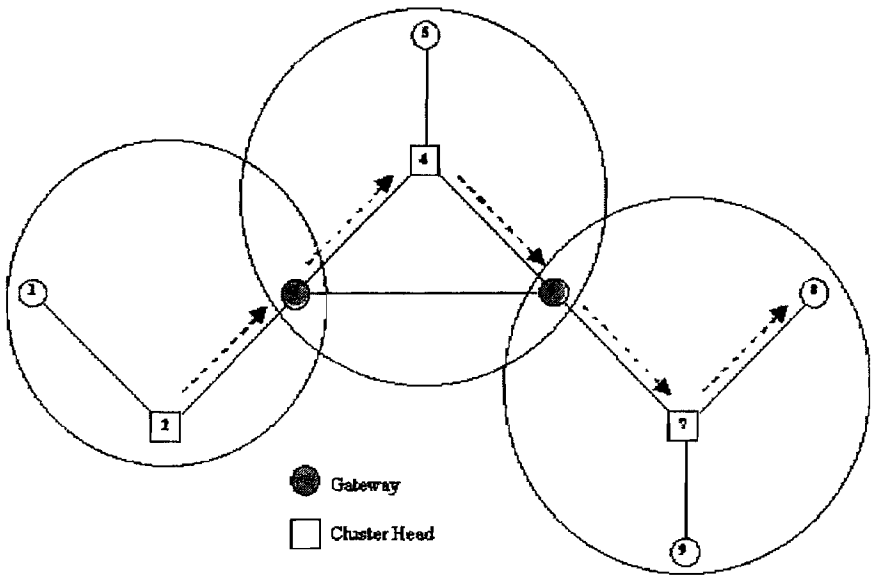


***Figure 2** : Exactitude d'information dans FSR*

2.5 Cluster-Gateway Switching Routing (CGSR) [CHI 97]

CGSR est basé sur une architecture de réseau basée sur les groupes. Dans CGSR, des nœuds spécifiques sont indiqués pour être des *clusterheads* (*chefs de groupe*), qui commandent l'accès au canal dans un groupe au niveau de la couche MAC. Dans la couche MAC, les clusterheads sont plus prioritaires, et ils ont ainsi plus de chance de transmettre que d'autres nœuds Tous les nœuds dans un groupe peuvent communiquer avec leur clusterhead et probablement entre eux.

Pour empêcher les changements fréquents de clusterhead, « *Least Clusterhead Change Algorithm*» (*LLC*) est employé. Dans cet algorithme, les clusterheads peuvent changer seulement aux deux conditions suivantes: (1) deux clusterheads deviennent des voisins, et (2) un nœud devient déconnecté de n'importe quel groupe.



a)

Groupe De Destination	Prochain Saut	Métrique	#ordre
7	3	4	71

b) **Exemple de Tableau de routage**

Destination	Clusterhead Destination	#ordre
8	7	100

c) **Exemple de Tableau de membre de groupe**

Figure 3 : Exemple de routage (de nœud 2 à nœud 8)

Dans CGSR, le protocole DSDV est modifié de sorte qu'il puisse profiter de l'architecture de groupe. Spécifiquement, le routage hiérarchique est employé pour acheminer des paquets. Chaque nœud maintient deux tables: une table des membres de groupe qui associe à chaque nœud destination son clusterhead, et une table de routage qui indique le prochain saut pour atteindre le groupe de destination. Les deux tables contiennent des numéros de séquence pour purger les routes périmées et pour empêcher la formation de boucles.

Dans CGSR, des paquets sont cheminés alternativement par des clusterheads et des passerelles. En d'autres termes, la route typique ressemble à $C_1G_1C_2G_2...\ C_iG_i$, où C_i est un clusterhead et G_i est une passerelle (gateway). Une passerelle est un nœud qui appartient à plus d'un groupe, un chemin

direct entre deux passerelles n'est pas utilisé. Cette voie de déroutement stricte a comme conséquence l'incrémentation de la longueur de chemin. La présence d'un clusterhead entre deux passerelles est avantageuse puisque les clusterheads ont plus de chance de transmettre que d'autres nœuds. La figure 3 montre un exemple concernant CGSR. L'avantage principal de CGSR est que seulement les routes aux clusterheads sont maintenues dus au routage hiérarchique. Cependant, le maintien des groupes cause une charge supplémentaire. Spécifiquement, chaque nœud doit périodiquement annoncer sa table de membre de groupe et mettre à jour sa table basée sur les mises à jour reçues.

2.6 Hierarchical State Routing (HSR) [IWA 99]

La caractéristique du routage hiérarchique d'état (HSR) est le groupement multiniveaux et la division logique des nœuds mobiles. Le réseau est divisé en groupes, et un représentant pour chaque groupe est élu.
Dans HSR, les représentants de groupes s'organisent encore en groupes et ainsi de suite. Les nœuds d'un groupe physique diffusent leurs informations de liens entre eux. Le chef de groupe récapitule l'information de son groupe et l'envoie aux chefs de groupe voisins par l'intermédiaire d'une passerelle.

Les chefs de groupes sont membres du groupe sur un plus haut niveau et ils échangent leur information de liens aussi bien que les informations les plus élémentaires des niveaux inférieurs et ainsi de suite. Un nœud à chaque niveau inonde au niveau le plus bas l'information qu'il obtient après que l'algorithme soit exécuté à ce niveau. Ainsi le niveau le plus bas a une information hiérarchique de topologie. Chaque nœud a une adresse hiérarchique composée des numéros des groupes sur le chemin de la racine au nœud. Une passerelle peut être atteinte de la racine par l'intermédiaire de plus d'une voie d'accès. Ainsi, la passerelle peut avoir plus qu'une adresse hiérarchique. Une adresse hiérarchique a pour but d'assurer la livraison de n'importe où dans le réseau à un hôte.

En outre, des nœuds sont également divisés dans des sous-réseaux logiques et à chaque nœud est assigné une adresse logique < subnet, host >. Chaque sous-réseau a un serveur de gestion d'emplacement dit LMS (Location Management Server). Tous les nœuds enregistrent leur adresse logique dans le

LMS. Les LMS diffusent leur adresse hiérarchique aux niveaux supérieurs et l'information est envoyée vers le bas à tous les LMS. La couche transport envoie un paquet à la couche réseau avec l'adresse logique de la destination. La couche réseau trouve l'adresse hiérarchique de la destination de son LMS et lui envoie alors le paquet. Une fois que la source et la destination savent les adresses hiérarchiques d'autres nœuds, ils peuvent ignorer le LMS et communiquer directement.

Les principaux inconvénients de HSR sont :

- si une route devient invalide, tous les paquets envoyés seront perdus jusqu'à ce qu'une nouvelle route sera établie.
- HSR maintient des adresses hiérarchiques longues, et le changement fréquent des adresses rend la localisation et la garde de trace des nœuds une tache difficile.

2.7 Zone Based Hierarchical Link State (ZHLS) [JOA 99]

Dans le protocole Zone-based Hierarchical Link State Routing Protocol (ZHLS) [LEM 00], le réseau est divisé en zones non-recouvertes (figure 4).

À la différence d'autres protocoles hiérarchiques, il n'y a aucun chef de zone. ZHLS définit deux niveaux de topologie: niveau de nœud et niveau de zone. Une topologie de niveau de nœud indique que les nœuds d'une zone sont reliés entre eux physiquement. Un lien virtuel entre deux zones existe si au moins un nœud d'une zone est physiquement relié à un certain nœud de l'autre zone. La topologie de niveau de zone indique comment les zones sont reliées ensemble.

Il y a deux types de paquet d'état de lien : *LSP orienté nœud* et *LSP orienté zone*.

- *LSP orienté nœud* : contient les informations des nœuds voisins, et il est propagé à l'intérieur de la zone.
- *LSP orienté zone* : contient les informations de zone et il est propagé globalement. Chaque nœud a la connaissance complète de connectivité des nœuds dans sa zone, et connais seulement les informations de connectivité de zone d'autre zone dans le réseau. Alors en donnant l'identification de zone et l'identification de nœud d'une destination, le paquet est conduit en se basant sur l'identification de zone jusqu'à ce qu'il atteigne la zone correcte. Puis dans cette zone, il est conduit en se basant sur l'identification de nœud.

Une < identification de zone, le nœud identification > de la destination est suffisante pour que le routage soit adaptatif au changement topologique.
Parmi les inconvénients de ZHLS on peut citer:
- Si la zone devient large, une quantité importante d'information doit être sauvegardée pour chaque nœud.
Les chemins ne sont pas optimaux, les paquets doivent passer par les nœuds qui relient les zones de la source et la destination même si ces derniers sont physiquement proches.

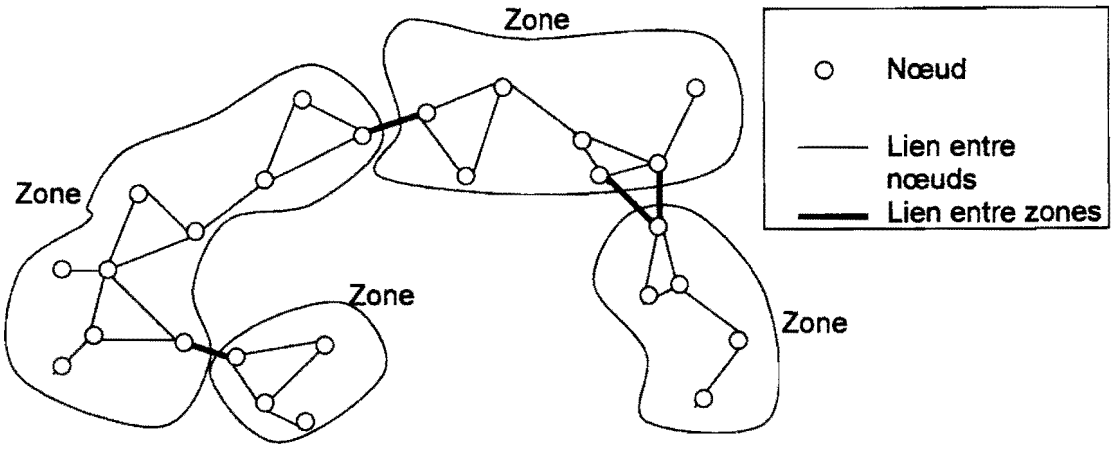


Figure 4 *: La décomposition du réseau en zones.*

2.8 Distance Routing Effect Algorithm for Mobility (DREAM) [BAS98]

Le protocole appelé "Algorithme d'Effet de Routage basé sur la Distance, pour la Mobilité" ou DREAM (Distance Routing Effect Algorithm for Mobility) est un protocole pro-actif basé sur les informations de localisation 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 précédemment).
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 [LEM 00].

3 Les protocoles de routage réactifs

Comme nous l'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 même si elles ne sont pas utilisées. La sauvegarde permanente des chemins de routage est assurée par un échange continu 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), représentent 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), appartiennent à cette classe de protocoles de routage.

Les protocoles de routage appartenant 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.

3.1 Dynamic Source Routing (DSR) [JOH 96]

DSR est basé sur le routage de source, où la source indique le chemin complet à la destination dans l'en-tête du paquet et chaque noeud le long de ce chemin expédie simplement le paquet au prochain saut indiqué dans le chemin (voir figure 5). Il utilise un cache de routes. Par conséquent, une source contrôle d'abord son cache de routes pour déterminer la route à la destination. Si une route est trouvée, la source utilise cette route. Autrement, la source emploie un protocole de découverte de route pour construire une route.

Dans la découverte de route, la source diffuse par inondation un paquet de requête dans le réseau, et la réponse est retournée par la destination ou un autre nœud qui peut accomplir la requête de son cache de routes. Chaque paquet de requête a une identification unique ID et une liste initialement vide. En recevant un paquet de requête, si un nœud a déjà vu cette identification (duplication) ou il trouve sa propre adresse déjà enregistrée dans la liste, il ignore la copie et arrête l'inondation, autrement, il appose sa propre adresse dans la liste et envoie la requête à ses voisins, ceci dans le cas où il n'a pas de chemin caché vers la destination. Si un nœud peut accomplir la requête de son cache de routes, il peut envoyer un paquet de réponse à la source sans propager le paquet de requête plus loin.
En outre, n'importe quel nœud participant à la découverte de route peut apprendre des routes à travers des paquets de données en transit, et recueillir cette information de routage dans son cache de routes. Ce protocole de découverte de route est semblable au protocole « Address Resolution Protocol »de l'Internet (ARP), sauf que les demandes d'ARP ne se propagent pas au-delà d'un routeur. Il est également semblable au protocole de découverte de route utilisé dans des passerelles de routage de source dans IEEE 802 LANs.

Un échec de route peut se produire puisque les nœuds mobiles se déplacent d'un endroit à l'autre. Un échec de route peut être détecté par le protocole de la couche liaison, ou il peut être implicite quand aucune émission n'a été reçue pendant un moment d'un ancien voisin. Quand un échec de route est détecté, le nœud détectant l'échec envoie un paquet d'erreur à la source, qui emploie alors le protocole de découverte de route encore pour découvrir une nouvelle route.
Il faut noter que dans DSR, aucun message périodique de commande n'est employé pour l'entretien de route.

L'avantage principal de DSR est qu'il y a peu ou pas de messages de contrôle quand peu de sources communiquent avec les destinations rarement accédées. Mais peut avoir le problème de grand échelle (scalability). Pendant que le réseau devient plus grand, les paquets de commande et les paquets de messages deviennent également plus grands puisqu'ils doivent véhiculer des adresses pour chaque nœud dans le chemin.

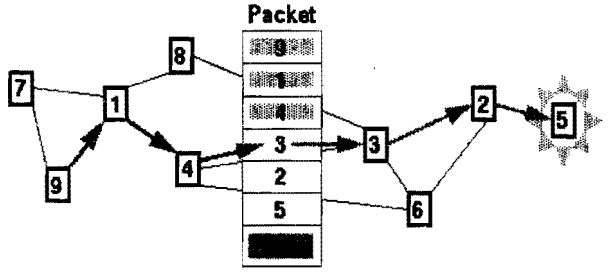


***Figure 5** : transfert du paquet du nœud 9 au nœud 5*

3.2 Ad Hoc On-Demand Distance Vector (AODV) [PER 99]

AODV est essentiellement une combinaison de DSR et de DSDV. Il combine le mécanisme de demande de base de la découverte et de l'entretien de route de DSR et de l'utilisation du routage de saut-par- saut, des numéros de séquence et des échanges périodiques de DSDV.

Quand un nœud *S* a besoin d'une route à une certaine destination *D*, il diffuse un message de demande de route à ses voisins, y compris le dernier numéro de séquence pour cette destination. La demande de route est inondée en quelque sorte par le réseau jusqu'à ce qu'elle atteigne un nœud

qui a une route à la destination. Chaque nœud qui expédie la demande de route crée une route inversée au nœud *S*.
Quand la demande de route atteint un nœud qui a une route à *D*, ce nœud produit une réponse de route qui contient le nombre de sauts nécessaires pour atteindre *D* et le numéro de séquence pour *D* le plus récemment vu par le nœud produisant la réponse.
Chaque nœud qui participe à l'expédition de cette réponse vers le créateur de la demande de route (le nœud *S*), crée une route vers *D*. L'état créé dans chaque nœud se rappelle seulement du prochain saut et pas la route entière, comme il serait fait dans DSR.

Afin de mettre à jour des routes, une version de AODV exige que chaque nœud transmette périodiquement un message *HELLO*, à une cadence d'une fois par seconde. L'absence de réception de trois messages *HELLO* consécutifs d'un voisin est prise comme indication que le lien vers ce dernier est invalide. AODV suggère brièvement qu'un nœud puisse employer des méthodes de la couche physique ou de la couche de liaison pour détecter des ruptures de lien aux nœuds qu'il considère voisins.
Quand un lien devient invalide, tout nœud ascendant qui a récemment expédié des paquets à une destination en utilisant ce lien est signalé via une réponse non sollicitée de route contenant une métrique infinie pour cette destination. À la réception d'une telle réponse de route, un nœud doit établir une nouvelle route à la destination en utilisant la découverte de route comme décrit ci-dessus. AODV ne peut fonctionner qu'avec les liens bidirectionnels ce qui représente un inconvénient.

3.3 Cluster Based Routing Protocol (CBRP) [JIA 99]

Dans le protocole de routage basé sur les groupes (CBRP), et comme son nom l'indique, les nœuds sont divisés en groupes (voir figure 6). Pour former des groupes, l'algorithme suivant est utilisé:

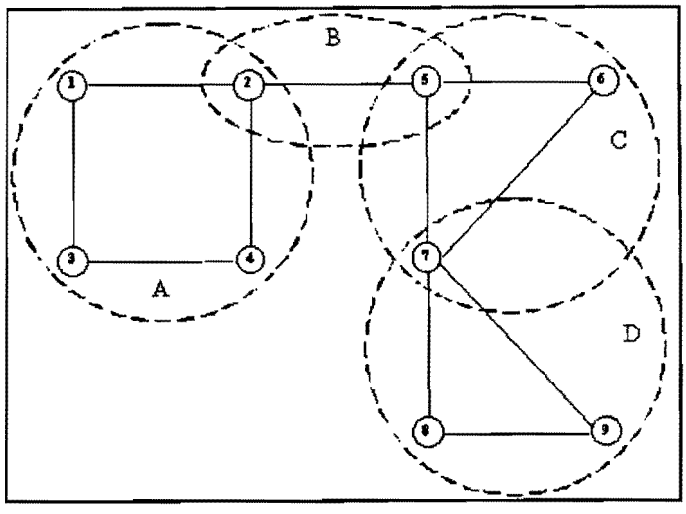
Quand un nœud entre dans le réseau, il entre dans l'état " indécis ", déclenche un timer et envoie le message *HELLO*. Quand un chef de groupe reçoit ce message il répond avec un message *HELLO* immédiatement. Quand le nœud indécis reçoit ce message il passe son état en "membre" (c'est à dire, il devient un membre de groupe). Si le délai de garde du nœud indécis

termine, alors il devient un chef de groupe s'il a au moins un lien bi-directionnel à un certain voisin autrement il reste dans l'état indécis et répète la procédure encore.

Les chefs de groupe (Clusterheads) sont changés aussi rarement que possible. Chaque nœud met à jour une table des voisins. Pour chaque voisin, la table des voisins d'un nœud contient le mode du lien (uni- ou bi-directionnel) et l'état du voisin (chef du groupe ou membre). Un chef de groupe garde des informations sur les membres de son groupe et également met à jour une table d'adjacence de groupe qui contient des informations sur les groupes voisins. Pour chaque groupe voisin, une entrée dans cette table contient la passerelle par laquelle le groupe peut être accédé et le chef de groupe (voir la figure 6 b).

Quand une source doit envoyer des données à la destination, elle diffuse par inondation des paquets de demande de route (mais seulement aux chefs de groupes voisins).
À la réception de la demande un chef de groupe vérifie si la destination est dans son groupe. Si oui, alors il envoie la demande directement à la destination autrement il l'envoie à tous ses chefs de groupe adjacents.
L'adresse de chefs de groupe est enregistrée dans un paquet ainsi un chef de groupe supprime un paquet de demande qu'il a déjà vu.
Quand la destination reçoit le paquet de demande, elle répond avec la route qui a été enregistrée dans le paquet de demande. Si la source ne reçoit pas de réponse au cours d'une période de temps donnée, elle essaye d'envoyer la demande de route de nouveau.
Dans CBRP, le routage est fait en utilisant le routage de source(source routing). Il utilise également le raccourcissement de route, en recevant un paquet de route source, le nœud essaye de trouver le nœud le plus lointain dans la route qui est son voisin (ceci pourrait avoir lieu à cause du changement de topologie) et envoie le paquet à ce nœud réduisant de ce fait la route.
En envoyant le paquet, si un nœud détecte un lien défaillant, il renvoie un message d'erreur à la source et utilise alors un mécanisme local de réparation.
Dans le mécanisme local de réparation, quand un nœud trouve que le prochain saut est inaccessible, il voit si le prochain saut peut être accédé

par un de ses voisins ou si le saut qui suit le prochain saut peut être accédé par n'importe quel autre voisin. Si un de ces deux cas est vérifié, le paquet peut être envoyé sur le chemin réparé.



a)

5, 6, 7	**c) Liste de membres du groupe C**

GROUPE	NŒUD DE LIAISON	REPRÉSENTANT
B	5	2
D	7	8

b) Table des groupes adjacents pour le nœud 6

Voisin	Statut	Etat de lien
8	Représentant de groupe	Uni ou bidirectionnel
7	Nœud de liaison	Uni ou bidirectionnel

d) Table des voisins pour le nœud 9

***Figure 6** : exemple de décomposition des groupes dans CBRP*

3.4 Lightweight Mobile Routing (LMR) [COR 95]

LMR (Le routage léger mobile) utilise la méthode source-initiated, qui construit des routes seulement une fois qu'un un nœud source en a décidé.

Pour construire une route vers une destination donnée, le nœud source diffuse par inondation dans le réseau un paquet de requête *QRY*. Un nœud donné n'envoie un paquet *QRY* qu'une seule fois (les paquets de requête dupliqués peuvent être détectés puisque chaque paquet a un numéro de séquence unique).

Un paquet de réponse *RPY* est retourné vers le nœud source par un ou plusieurs nœuds qui ont une route à la destination. Un nœud donné reçoit le paquet RPY à travers des liens appelés *upstream links* ou *liens ascendants*. On dit qu'un nœud a une route s'il connaît le prochain saut dans la route vers la destination, c'est-à-dire il a au moins un lien qui peut mener à la destination, appelé *downstream link* ou *lien descendant*.
Le paquet *RPY* passe seulement par des liens non dirigés, transformant ces liens non dirigés en liens dirigés vers l'origine de la réponse. Après que la propagation de réponse est terminée, LMR construit un ensemble de multiples routes sans boucle enracinées à la destination (construction d'un DAG ou Direct Acyclic Graph) comme représenté dans la figure 7.

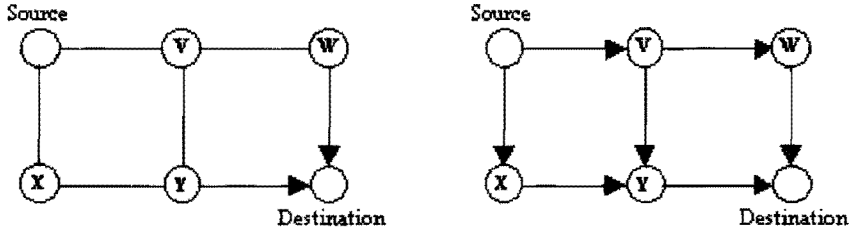
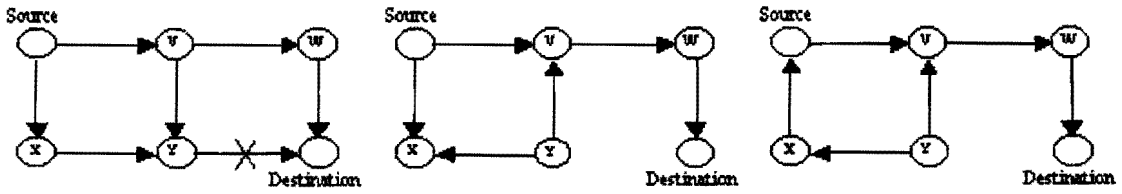


Figure 7 : *Le processus de requete-réponse*

Le protocole garantit que tous les nœuds participant à l'inondation de réponse obtiennent une ou plusieurs routes. Les routes supplémentaires augmentent la fiabilité. En outre, si l'information de nombre de sauts est ajoutée dans le paquet de réponse, chaque nœud peut améliorer sa décision de routage en choisissant le chemin le plus court. L'utilisation des routes optimales a une importance secondaire, l'importance primaire est de trouver une route rapidement afin qu'elle puisse être utilisée avant que la topologie ne change.
Dans la figure 8(a),le nœud Y perd son lien descendant à la destination. Ceci déclenche le mécanisme d'inversion de lien. Dans la figure 8(b), le nœud Y renverse la direction de tous ses liens ascendants par un paquet d'échec *FQ*. L'inversion du nœud y fait perdre au nœud X son lien descendant, ainsi il renverse plus tard son unique lien ascendant par l' expédition d'un paquet *FQ* au nœud ascendant (la figure 8(c)). D'autre part, l'inversion du nœud Y ne

cause pas le renversement de lien par V puisqu'il a toujours un lien descendant à la destination. Dans ce cas, le nœud Y arrête la transmission du paquet *FQ*, et diffuse un paquet *RPY*.



(a) (b) (c)

Figure 8 *: Processus d'inversement de liens*

Les avantages important de LMR sont :

1- LMR n 'a pas besoin des informations globales de connectivité.
2- Chaque nœud a besoin seulement de connaître les liens vers ses voisins adjacents.
3- Les nœuds n'ont pas besoin de garder l'information de routage complète (c-à-d chemin complet à la destination); au lieu de cela, ils doivent savoir seulement les nœuds de prochain-saut.
4- LMR a la capacité de maintenir les chemins multiples sans boucle à une destination donnée, où les chemins supplémentaires augmentent la fiabilité.

L'inconvénient de LMR est qu'il peut prendre beaucoup de temps pour converger si un échec de lien (ou le changement topologique) découpe le réseau. Dans une telle situation, LMR peut construire temporairement des routes invalides.

3.5 Temporally-Ordered Routing Algorithm (TORA) [PAR 97]

TORA est un protocole de routage distribué basé sur un algorithme d'inversion de liens et dérivé du protocole LMR. Il est conçu pour:

- Découvrir les routes à la demande.
- Fournir plusieurs routes pour une même destination.
- Minimiser l'effet des changements de la topologie.

L'optimisation de route est considérée d'importance secondaire, et les plus longues routes sont souvent utilisées pour éviter le temps de découvrir de nouvelles routes.
Le protocole exécute trois fonctions : (a) création des routes, (b) maintenance des routes et (c) suppression des routes.

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*), les 3 premiers champs sont nommés niveau de référence (reference level), et les 2 autres delta.

Le processus de création (ou de découverte) de routes pour une destination donnée, commence quand un 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 n'a pas un lien sortant (downstream link) et qui reçoit le paquet *QRY*, rediffuse le paquet à ses voisins. Un nœud qui a un lien sortant, met à jour sa taille et répond par l'envoi d'un paquet *UPD (update*) qui contient sa nouvelle taille. Lors de la réception du paquet *UPD*, le nœud récepteur affecte la valeur de taille contenue 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. De cette façon, un DAG est crée du nœud source vers le nœud destination (figure 9).

A cause de la mobilité des nœuds, 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* perd son dernier lien sortant à cause d'une défaillance, (s'il existe toujours au moins un lien sortant après cette défaillance, aucune réaction n'est effectuée (figure 9(1)), sinon 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 orientés 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 d'un ensemble de liens qui participent dans les

chemins, où une défaillance est détectée; ce qui reconstruit le DAG ou indique à la source l'invalidité des chemins rompus. (figure 9(2,3,4))

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.

TORA suppose que tous les nœuds ont des horloges synchronisées par une source de temps externe comme Global Positioning System (GPS), ce qui limite son application. Si la source de temps externe tombe en panne cet algorithme cesse de fonctionner.

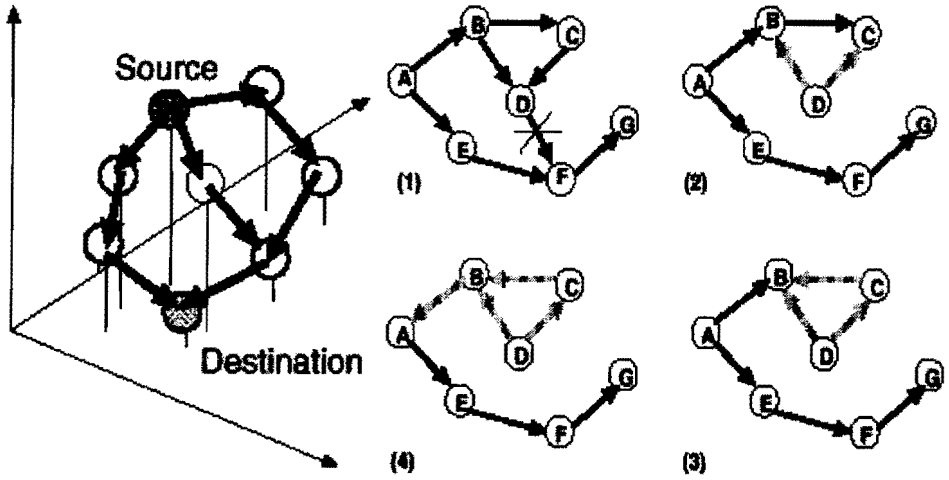


Figure 9 : *Création et maintenance des routes dans TORA*

3.6 Associativity Based Routing (ABR) [TOH 96]

Le protocole ABR utilise une nouvelle approche de routage, il définit une nouvelle métrique connue sous le nom de degré de stabilité d'association.
Dans l'ABR, le chois des routes est basé sur *les états d'associativité des nœuds*.
Les routes choisies ainsi auront probablement une durée de vie longue.
Chaque nœud produit des « *beacons* » (message de contrôle) périodiques pour signaler son existence. Quand un nœud voisin reçoit un *beacon*, il met à jour ses tables d'associativité. Pour chaque *beacon* reçu, un nœud incrémente sa valeur d'associativité qui correspond au nœud émetteur.

La stabilité d'association signifie la stabilité de la connexion d'un nœud avec un autre. Une valeur d'associativité élevée avec un nœud indique un bas état de mobilité du nœud, alors qu'une valeur basse de valeur d'associativité peut indiquer un état élevé de mobilité.
L'objectif principal du protocole ABR est de trouver des chemins de longue durée de vie. ABR consiste en trois fonctions : (a) découverte des routes, (b) reconstruction des routes (RRC), et (c) 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*) (figure 10(a))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 envoie 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 causé le mouvement d'une route. 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*) (figure 10(a)), 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]*) (figure 10(b)), 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 son timeout, et relance par la suite le même processus pour le voisin suivant. Si le processus de localisation *LQ* échoue pour tous les voisins, 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.

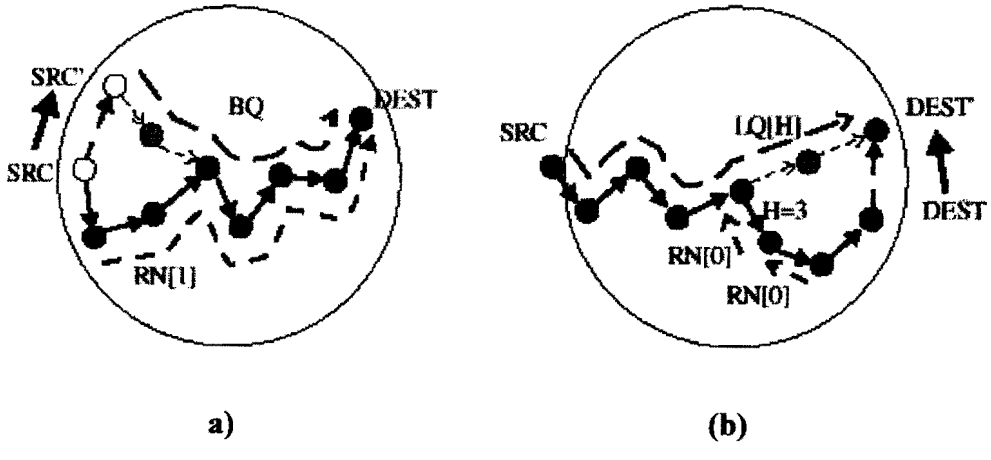


Figure 10 : *Les phases de ABR*

3.7 Signal Stability Routing (SSR) [DUB 97]

Le protocole de routage adaptatif SSR choisit les routes en se basant sur la puissance de signal entre les nœuds, et la stabilité de l'emplacement des nœuds.

Ce critère de sélection de route a l'effet de choisir les routes qui ont une connectivité plus forte.

SSR comporte deux protocoles coopératifs: le protocole dynamique de routage (DRP) et le protocole de routage statique (SRP).
Le DRP maintient la table de stabilité de signal (SST) et la Table de routage (RT). Le SST enregistre la puissance de signal des nœuds voisins obtenus par des messages périodiques de la couche de lien de chaque nœud voisin.
La puissance de signal a deux valeurs: canal fort ou faible. Toutes les transmissions sont reçues et traitées par DRP. Après la mise à jour des entrées appropriées de la table, le DRP passe le paquet vers le SRP. Le SRP passe le paquet à la pile si c'est le récepteur ciblé. Sinon, il recherche la destination dans la table de routage (RT) et expédie le paquet. S' il n'y a aucune entrée pour la destination dans RT, il lance le processus de recherche de route pour trouver une route. Les paquets de demandes de routes sont expédiés au prochain saut seulement s'ils sont reçus sur des canaux puissants et n'ont pas été précédemment traités (pour éviter de former une boucle). La destination choisit le premier paquet de demande de route arrivé pour le renvoyer car il est fortement probable que le paquet soit arrivé via le chemin le plus court et/ ou le moins encombré.
Le DRP renverse la route choisie et envoie le message de réponse « route-reply » à l'initiateur de la demande de route. Les DRP des nœuds tout au long du chemin mettent à jour leur RTs en conséquence.
Les paquets de recherche de chemin arrivant à la destination sont nécessairement arrivés sur le chemin de stabilité de signal la plus puissante, parce que les paquets arrivant sur un canal faible sont abandonnés aux nœuds intermédiaires. Si le délai de garde de la source termine avant de recevoir une réponse alors elle change le champ PREF de l'entête pour indiquer que les canaux faibles sont acceptables, (peut être seulement en utilisant ces liens que le paquet peut être propagé).
Quand une panne de lien est détectée dans le réseau, les nœuds intermédiaires envoient un message d'erreur à la source indiquant quel canal est défaillant. La source envoie alors un message d'effacement pour informer tous les nœuds que ce lien est devenu invalide, ensuite il initie une nouvelle recherche de route pour trouver un nouveau chemin à la destination.

3.8 Location Aided Routing (LAR) [KO 98]

Le protocole LAR est similaire au DSR mais la différence réside dans le fait qu'il utilise les informations de localisation, 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.

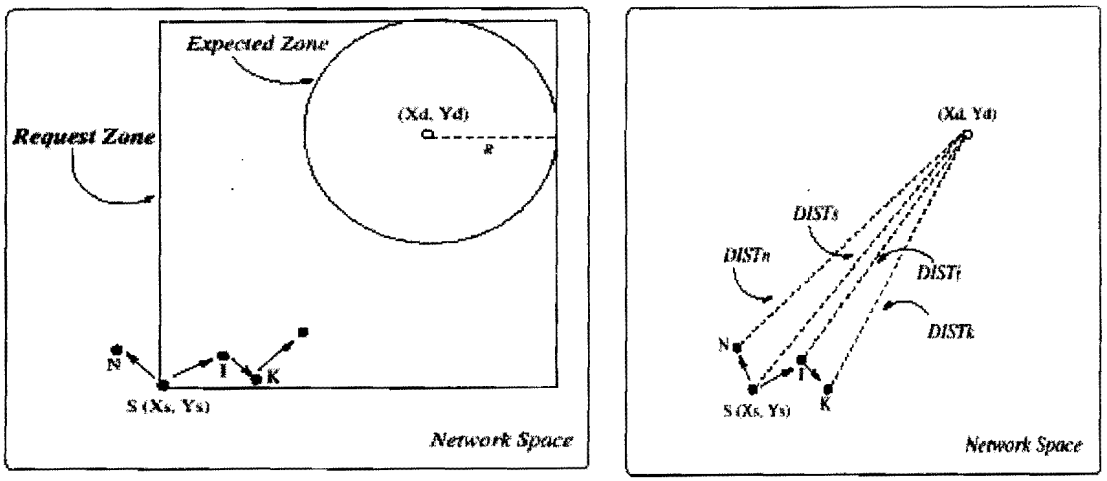
Dans la première approche (figure 11(a)), un nœud source définit une région circulaire dans laquelle la destination peut être localisée. L'estimation de position et la taille de la région se fait en se basant sur :

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 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 cette zone.

Dans la deuxième approche (figure 11(b)), le nœud source calcule la distance qui le 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 le 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 le 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).



(a) (b)

Figure 11 : *Les deux approches de LAR*

3.9 Relative Distance Micro-discovery Ad hoc Routing (RDMAR) [AGG99]

Le protocole de Routage basé sur la Micro découverte des Distances Relatives ou RDMAR (Relative Distance Micro-discovery Ad hoc Routing) est un protocole réactif conçu principalement pour s'adapter aux changements rapides des réseaux ad hoc en réduisant le contrôle utilisé.

Une 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 l'avons déjà vu, les

protocoles LAR et DREAM (sections 3.8 et 2.8) 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 destinataires qui peuvent être atteints. Une entrée qui est associée à une destination donnée, contient les informations suivantes :
Le routeur par défaut : qui est un champ indiquant le nœud suivant à travers lequel le nœud courant peut atteindre la destination.
Un champ RD : qui donne la distance estimée entre le nœud et la destination.
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.
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.
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 le 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 de la nouvelle distance relative (RDM_Radius), dans le champ TTL 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. 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.

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 restent passifs. Quand un nœud intermédiaire *i* reçoit un paquet de données, il traite d'abord l'en-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 données, 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"*. 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*) [LEM 00].
zone, ZRP est plus réactif.

3.10 Zone Routing Protocol [HAA97]

Dans le protocole de routage de zone, chaque nœud a sa propre zone de routage qui inclut les nœuds dont la distance (en nombre de sauts) ne dépasse pas un certain nombre prédéfini (rayon de la zone). Chaque nœud

doit savoir la topologie du réseau localement dans sa zone de routage, et des mises à jour sont propagées seulement dans cette zone. Un protocole proactive tel que DSDV est employé dans une zone de routage pour se renseigner sur sa topologie. Pour découvrir un chemin aux nœuds d'une zone extérieur, un protocole réactif tel que DSR est employé. Notez que ZRP montre le comportement hybride proactif et réactif par l'utilisation du rayon de zone. Pour un grand rayon de zone, ZRP est plus proactif, et pour un petit rayon de zone, ZRP est plus réactif.

La découverte de route utilisée dans ZRP est illustrée dans la figure 12. Supposons que la source A veut découvrir une route à la destination E. A vérifie d'abord que E est hors de sa zone. Il envoie alors un paquet de requête à tous les nœuds sur la périphérie de sa zone, c.-à-d., B et F. En recevant un paquet de requête, chacun de ces nœuds ajoute son adresse au paquet de requête et lui fait suivre ses nœuds périphériques puisque E n'est pas dans sa zone de routage. En particulier, B expédie le paquet de requête à C, qui identifie E en tant que son membre de zone. D envoie alors un paquet de réponse qui inclut la route A-B-C-E.

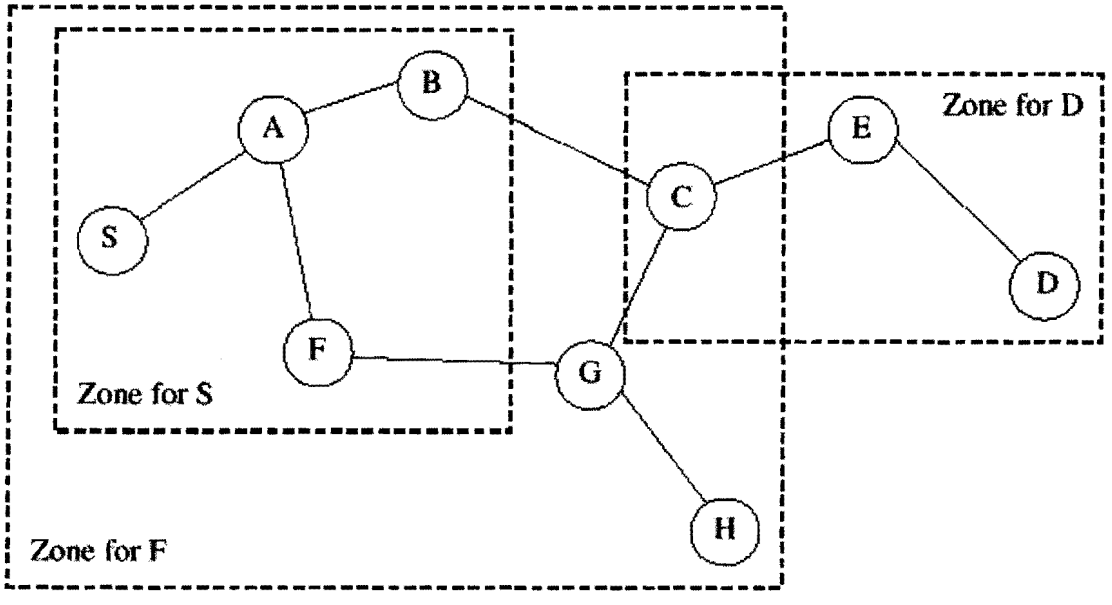


Figure 12 : *Décomposition en zones dans ZRP*

4- Conclusion

Dans cet article, nous avons présenté plusieurs protocoles de routage qui ont été proposés pour assurer le service de routage dans les réseaux mobiles ad hoc.

Assurer la connexion de tous les nœuds d'un réseau ad hoc est un problème très complexe à cause de la dynamicité et l'évolution rapide de la topologie. En effet, les unités mobiles sont dynamiquement et arbitrairement éparpillées 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.

Les protocoles proposés sont classés en deux catégories principales: les protocoles pro-actifs et les protocoles réactifs. Les protocoles des deux catégories essaient de s'adapter aux contraintes imposées par les réseaux ad hoc, et cela en proposant une méthode qui soit de moindre coût en ressources, et qui garantit la survivabilité du routage en cas de panne de liens ou de nœuds.

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.

5– Bibliographie

[AGG 98]: George Aggelou and Rahim Tafazolli. "RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks". Proceedings of the ACM International Workshop on Wireless Mobile Multimedia (WoWMoM), Seattle, WA, August 1999, pp. 26-33.

[BAS 98]: S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. "A distance routing effect algorithm for mobility (DREAM)". In Proceeding of ACM/IEEE MOBICOM'98, Dallas, Texas, pp 76-84, october 1998.

[BER 92] : D. Bertsekas and R. Gallager. "Data Networks". Prentice Hall Inc, pp 297-333, 1992.

[CHE 98]: Tsu-Wei Chen and Mario Gerla. "Global State Routing: A new routing scheme for ad-hoc wireless networks". Proceedings of the IEEE International Conference on Communications (ICC), Atlanta, GA, June 1998, pp. 171-175.

[CHI 97]: C. C. Chiang, H-K Wu, Winston Liu, and Mario Gerla. "Routing in clustered multihop, mobile wireless networks". The IEEE Singapore International Conference on Networks, pp 197-211, 1997.

[COR 95]: Corson, M.S.; Ephremides, A. "A distributed routing algorithm for mobile wireless networks.", Wireless Networks, vol.1, (no.1), p.61-81. 1995.

[DUB 97]: R. Dube et al. "Signal stability based adaptive routing for ad hoc mobile networks". IEEE Pers. Comm., pp 36-45, February 1997.

[HAA 97]: Z. Haas. "A new routing protocol for the reconfigurable wireless networks". In Proc of the IEEE Int. Conf. on Universal Personal Communications., Oct 1997.

[IWA 99]: Iwata, C. C. Chiang, G. Pei, M. Gerla, and T.-W. Chen. "Scalable routing strategies for ad hoc wireless networks". IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, pp.1369-1379, August 1999.

[JIA 99] : Mingliang Jiang, Jinyang Li, Y.C. Tay. "Cluster based routing protocol", IETF Draft, August 1999. http://www.ietf.org/internet- drafts/draft-ietf-manet-cbrp-spec-01.tx t.

[JOA 99]: M. Joa-Ng and I.-T. Lu. "A Peer-to-Peer zone-based two-level link state routing for mobile ad hoc networks". IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, pp 1415-1425, August 1999.

[JOH 96] : David.B Jhonson, David.A Maltz. "Dynamic Source Routing in Ad Hoc wireless" in Mobile Computing, edited by T. Imielinski and H. Korth, chapter 5, pp. 153-181, Kluwer, 1996.

[KO 98]: Y.-B. Ko and N.H. Vaidya. "Location-aided routing (LAR) in mobile ad hoc networks". In Proceedings of ACM/ IEEE MOBICOM'98, Dallas, Texas, pp 66-75, October 1998.

[LEM 00]: Tayeb Lemlouma. "Le routage dans les réseaux mobiles Ad Hoc", Mini projet, USTHB, 2000.

[MUR 96]: S. Murthy and J.J. Garcia-Luna-Aceves. "An efficient routing protocol for wireless networks". ACM Mobile Networks and Application Journal, Special Issue on routing in mobile Communication Networks, pp183-197, October 1996

[PAR 97]: V. D. Park and M. S. Corson. "A highly adaptive distributed routing algorithm for mobile wireless networks". Proceeding INFOCOM '97, pp 1405-1413, April 1997.

[PEI 00]: Guangyu Pei, Mario Gerla, Tsu-Wei Chen. "Fisheye State Routing in Mobile Ad Hoc Networks". Proceedings of the IEEE International Conference on Communications (ICC), New Orleans, LA, June 2000, pp. 70-74.

[PER 94]: Charles. E. Perkins and Pravin Bhagwat. "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer". ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications, pp. 234-244, 1994.

[PER 99] : Charles. E.Perkins, Elizabeth M.Royer. "Ad hoc on demand distance vector (AODV) algorithm". In Systems and Applications (WMCSA'99), page 90-100, 1999.

[TOH 96]: Chai-Keong Toh. "A novel distributed routing protocol to support ad hoc mobile computing". Proceeding 1996 IEEE 15th Annual Int'l. Phoenix Conf. Comp. And Commun, pp 480-86, March 1996.