République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de constantine
Faculté des sciences et science de l'ingénieur
Département d'informatique

N° d'ordre :
Série :

Mémoire

Présenté en vue de l'obtention du diplôme de

Magister en informatique

Option : GL & IA**.**

ROUTAGE DANS LES RESEAUX MOBILES AD HOC PAR UNE APPROCHE A BASE D'AGENTS

Présenté par : Mr BOUKHECHEM Nadhir

Soutenue-le : 25 / 06 / 2008 devant le jury composé de :

Président :

Mr BENMOHAMED Mohamed (Prof. Université Mentouri de Constantine)

Rapporteur :

Mr SAHNOUN Zaidi (Prof. Université Mentouri de Constantine)

Examinateurs :

Mr MAAMRI Ramdane (M.C. Université Mentouri de Constantine)

Mme BELALA Faiza (M.C. Université Mentouri de Constantine)

Promotion 2007-2008

Table des matières

Liste des Figures

Liste des Tableaux

Remerciements

Si ce mémoire a pu voir le jour, c'est certainement grâce au soutien et l'aide de plusieurs personnes qui m'ont permis d'accomplir ce travail dans des conditions idéales. Je profite de cet espace pour les remercier tous.

Je voudrais tout d'abord exprimer mes profonds remerciements a ma famille pour leur soutien inconditionnel, merci pour m'avoir encouragé, supporté et pour avoir accepté tant de sacrifices durant cette période.

Je tiens à remercier Mr SAHNOUN Zaidi Professeur à l'Université Mentouri de constantine pour son encadrement. Son expérience et ses qualités scientifiques ont toujours été sources d'enrichissement me permettant de mener à bien ce travail.

Enfin, je remerci Mr MAAMRI Ramdane , M.C. A l'Université Mentouri de Constantine, et Mme BELALA Faiza , M.C. A l'Université Mentouri de Constantine pour avoir accépté d'être les examinateurs de ce mémoire et Mr BENMOHAMED Mohamed Professeur A l'Université Mentouri de Constantine pour avoir accépté d'être le président.

Résumé

Les réseaux mobiles ad hoc appelés généralement MANET (Mobile Ad hoc NETwork) sont un nouveau type de réseaux basés sur la technologie sans fil. Un réseau ad hoc est constitué d'un ensemble d'unités de calcul portables comme les PDA (Personal Digital Assistant) et les laptops qui se déplacent librement et forment ensemble d'une manière dynamique un réseau interconnecté. Les réseaux ad hoc ne dépendent d'aucune infrastructure préétablie, les noeuds mobiles doivent coopérer ensemble pour pouvoir gérer leurs communications.

Le routage est le problème le plus important dans les réseaux ad hoc en effet à cause de la mobilité des noeuds il est très difficile de localiser une destination à un instant donné. Plusieurs protocoles de routage pour les réseaux ad hoc on été proposés ces protocoles essayent de maximiser les performances en minimisant le délai de livraison des paquets, l'utilisation de la bande passante et la consommation d'énergie.

Les systèmes multiagents sont une nouvelle technologie issue du domaine de l'intelligence artificielle cette technologie est particulièrement intéréssante à utiliser dans des environnements de nature distribué et dynamique comme les réseaux ad hoc. Dans ce mémoire on propose un protocole de routage pour les réseaux mobiles ad hoc basé sur les agents.

Mots clés :
Réseaux Mobiles Ad Hoc, Protocoles de routages Ad Hoc, Systèmes Multiagents.

Abstract

The ad hoc mobile networks generally called MANET (Mobile Ad hoc NETwork) are a new type of networks based on wireless technology. An ad hoc network consists of a set of portable computing units such as PDA (Personal Digital Assistant) and laptops who move freely and form together in a dynamic way an interconnected network. The ad hoc networks not depend on any predetermined infrastructure, mobile nodes must cooperate together to be able to manage their communications.

The routing is the most important problem in ad hoc networks in effect because of the mobility of nodes it is very difficult to locate a destination at a given moment. Several routing protocols for ad hoc networks have been proposed these protocols are trying to maximize performance and minimize the time for delivery of packets, bandwidth usage and consumption of energy.

The multiagents systems are a new technology resulting from the field of the artificial intelligence this technology is particularly interesting to use in environments of nature distributed and dynamic like the ad hoc networks. In this report proposes a routing protocol for mobile ad hoc networks based agents.

Keywords:
Mobile Ad Hoc Networks, Ad Hoc Routing Protocols, Multiagents Systems.

Introduction générale

Ces dernières années le développement de la technologie sans fil a ouvert de nouvelles perspectives dans le domaine des télécommunications, les réseaux mobiles basés sur la technologie sans fil connaissent aujourd'hui une forte expansion. Les réseaux mobiles offrent une grande flexibilité d'emploi, ils permettent aux utilisateurs de se déplacer librement tout en continuant normalement leurs communications. Il existe deux types de réseaux mobiles, les réseaux mobiles avec infrastructure et les réseaux mobiles ad hoc. Les réseaux mobiles avec infrastructure sont basés sur un ensemble de sites fixes appelés stations de base, ces sites vont relier les différents noeuds mobiles pour former un réseau interconnecté. L'inconvénient de ce type de réseau c'est qu'il requière le déploiement d'une importante infrastructure fixe. Les réseaux ad hoc en contrepartie n'ont besoin d'aucune infrastructure fixe préexistante. Un réseau ad hoc est constitué d'un ensemble d'unités de calcul portables comme les PDA (Personal Digital Assistant) et les laptops qui sont munis d'une interface de communications sans fil. Ces unités se déplacent librement dans une certaine zone géographique et forment ensemble d'une manière dynamique un réseau interconnecté. Pour pouvoir communiquer entre eux chaque unité mobile doit jouer le rôle d'un routeur et d'un terminal, et doit retransmettre les paquets des autres unités mobiles. Les réseaux ad hoc offrent une grande flexibilité d'emploi et une grande robustesse et peuvent se déployer très rapidement.

Les recherches actuelles dans les réseaux ad-hoc sont dirigées vers les algorithmes de routage. En effet à cause de la mobilité des noeuds il est très difficile de localiser une

destination à un instant donné, les protocoles de routage conçus pour des réseaux statiques sont donc inadaptés pour ce type de réseaux. Plusieurs protocoles de routage pour les réseaux ad hoc ont été développés, chaque protocole essaye de maximiser les performances du réseau en minimisant le délai de livraison des paquets, l'utilisation de la bande passante et la consommation d'énergie. Les algorithmes de routage pour les réseux ad hoc peuvent se classer en trois catégories, les protocoles Table-Driven, les protocoles On-Demande et les protocoles Hybrides. Les protocole Tables-Driven maintiennent à jour des tables de routage qui indiquent les routes vers chaque destination du réseau, les routes sont donc calculées même si elles ne sont pas utilisées. L'avantage de ces protocoles c'est que la connexion entre les noeuds est immédiate puisque les routes sont calculées à l'avance, l'inconvénient c'est qu'ils utilisent beaucoups de paquets de contrôle pour maintenir à jour les tables de routage. Les protocoles On-Demand en contrepartie calculent les routes selon les besoins, la route est calculée quand elle est demandée, l'avantage de ces protocoles c'est qu'ils utilisent moins de paquets de contrôle que les protocoles Table-Driven, l'inconvénient est qu'ils ont un délai inital avant de commencer la transmition des données, c'est le délai nécéssaire pour déterminer la route. Les protocoles hybrides essaient de combiner les deux approches précédentes pour bénéficier de leurs avantages mais ils cumulent aussi leurs inconvénients. Le problèlme est donc de trouver un compromis et essayer d'avoir un délai initial court tout en utilisant un minimum de paquets de controle.

L'agent et les systèmes multiagents sont une nouvelle technologie issus du domaine de l'intelligence artificielle. Les systèmes multiagents sont appliqués dans beaucoup de domaines tels que les systèmes distribués, les interfaces homme machine et les réseaux. Les systèmes multiagents prennent en compte les aspects de coopération, d'autonomie, de distribution et d'intelligence, ils sont particulièrement intéréssants à utiliser dans des environnements de nature distribué et dynamique comme les réseaux ad hoc. Un système basé sur cette technologie semble donc approprié pour gérer le routage dans les réseaux ad hoc et obtenir de meilleurs performances.

Ce travail rentre dans le cadre de l'étude du problème de routage dans les réseaux mobiles ad hoc. Son objectif principal est de proposer un protocole de routage pour les réseaux ad hoc basé sur la technologie agent pour minimiser le délai initial de transmissions des paquets de données et minimiser l'utilisation des paquets de contrôle. Le protocole proposé se base sur la coopération d'agents statiques et d'agents mobiles pour découvrir la topologie du réseau et calculer les routes entre les noeuds. Les agents mobiles sont

responsables de la collecte des informations de routage et de la découverte des routes, les agents statiques utilisent les informations fournies par les agents mobiles pour déterminer les meilleurs routes pour transmettre les paquets de données.

Ce mémoire se divise en quatre chapitres :

Dans le premier chapitre on présente les réseaux mobiles ad hoc, leurs caractéristiques, leurs applications et le problème de routage.

Au deuxième chapitre on présente une classification des différents protocoles de routage pour les réseaux ad hoc et on decrit en détail quelques protocoles tout en indiquant leurs avantages et leurs inconvénients.

Dans le troisième chapitre est présenté l'agent et les systèmes multiagents et leurs avantages dans les réseaux ad hoc, ainsi que quelques protocoles de routage basés sur les agents.

Au quatrième chapitre on propose un protocole de routage pour les réseaux ad hoc basé sur les agents, on donne le modèle du système, le role des différents agents et leurs intéractions possibles, on compare notre protocole au protocole de routage AODV en utilisant le simulateur de réseau NS2 et on présente les résultats obtenus. Enfin on termine ce mémoire par une conclusion générale.

Chapitre 1
Les réseaux mobiles Ad Hoc

1.1. Introduction

Ces dernières années le développement de la technologie de transmission sans fil a offert de nouvelles perspectives dans le domaine des télécommunications. Les réseaux mobiles ad hoc appelés généralement MANET (Mobile Ad hoc Network) constituent un

nouveau type de réseaux basés sur cette technologie. Un réseau mobile ad hoc permet à ses utilisateurs de se déplacer librement tout en continuant normalement leurs communications, il peut se déployer rapidement et n'a besoin d'aucune infrastructure fixe préexistante. Dans se chapitre nous allons présenté les réseaux mobiles ad hoc leurs caractéristiques, leurs applications et le problème de routage qui est le plus important dans ce type de réseaux.

1.2. Les réseaux Mobiles

Un réseau est dit mobile s' il permet à ses utilisateurs d'accéder à l'information indépendamment de leurs position géographique. Pour communiquer entre eux les noeuds du réseau mobile utilisent une interface de communication sans fil (médium radio ou infrarouge) qui permet de propager les signaux sur une certaine distance. Les réseaux mobiles offrent une plus grande flexibilité d'emplois et un plus grand confort par rapport au réseaux statiques.

Nous pouvons distinguer deux classes de réseaux mobiles, les réseaux mobiles avec infrastructure de communication, et les réseaux mobiles sans infrastructure de communication ou les réseaux Ad Hoc.

1.2.1. Les réseaux mobiles avec infrastructure

Un réseau mobile avec infrastructure est basé sur un ensemble de sites fixes appelés stations de base qui sont interconnectés entre eux à travers un réseau de communication filaire, chaque station de base peut communiquer directement en utilisant une interface sans fil avec les noeuds mobiles se trouvant dans une zone géographique limitée comme le montre la figure suivante :

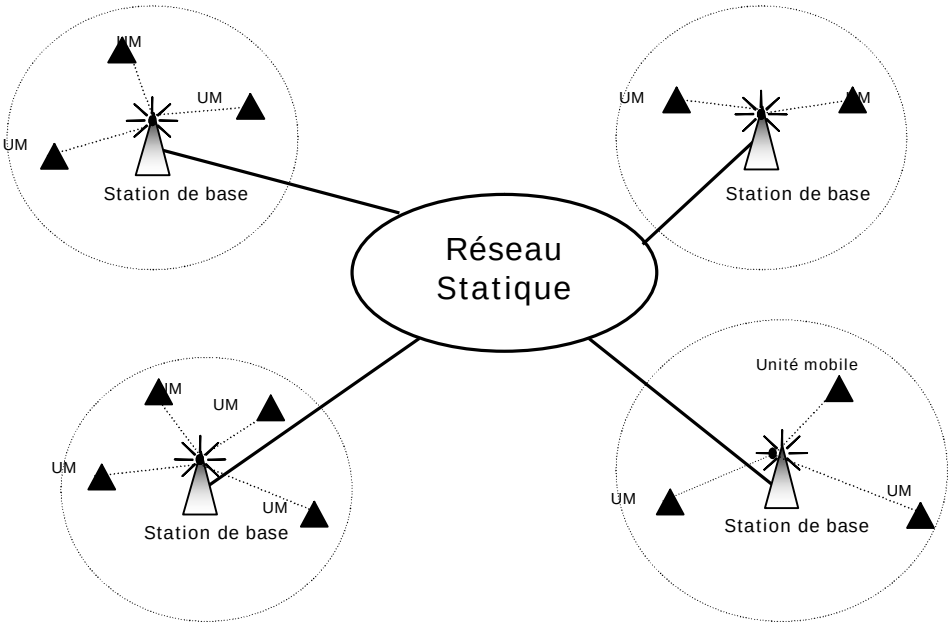


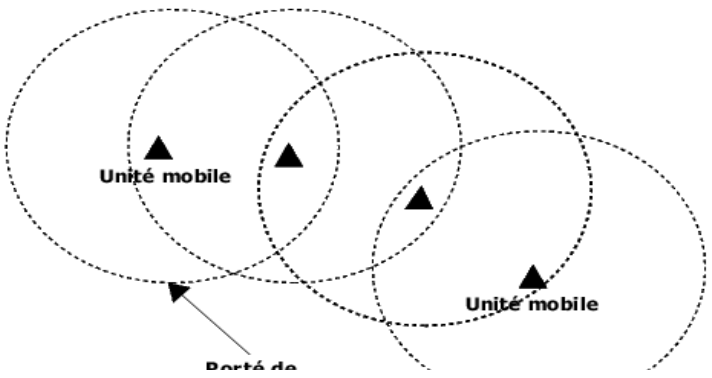
Figure 1.1 : Réseau mobile avec infrastructure

Un noeud dans les réseaux mobiles avec infrastructures se connecte et communique avec les autres noeuds du réseau a travers la station de base la plus proche dans sa porté de communication, si le noeud mobile sort de la porté de cette station il doit trouver une autre station de base pour continuer la communication. Les réseaux mobiles avec infrastructures coûtent chère car ils demandent le déploiement d'une importante infrastructure fixe.

1.2.2. Les réseaux Ad Hoc

Un réseaux Ad Hoc appelé généralement MANET (Mobile Ad hoc Network), est composé d'un ensemble relativement danse de noeuds mobiles qui se déplacent librement dans une certaine zone géographique sans aucune infrastructure fixe préexistante. Un noeud dans le réseau ad hoc communique avec un autre noeud directement (en utilisant son interface sans file), si ce dernier est dans sa porté de transmission, ou indirectement par l'intermédiaire d'autres noeuds du réseau dans le cas contraire. Chaque noeud dans le réseau ad hoc doit se comporter comme un terminal, et aussi comme un routeur, et participer à la découverte et la maintenance des routes entre les noeuds du réseau.

Il y a aucune limitation de taille dans un réseau ad hoc , il peut contenir des dizaines ou



des milliers de noeuds, la figure suivante illustre un réseau ad hoc composé de quatre noeuds mobiles.

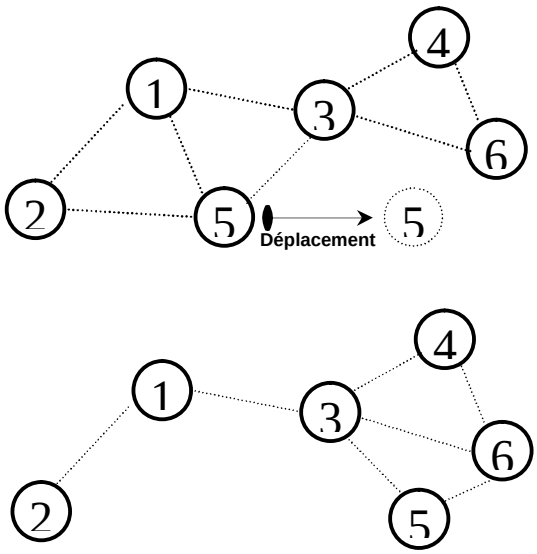
Figure 1.2 : Un réseaux ad hoc composé de quatre noeuds

Les réseaux Ad Hoc ont en théorie une très grande robustesse puisque pour que le réseau cesse de fonctionner, il faudrait qu'un nombre important de noeuds qui le compose soit hors service. En effet si un des noeuds du réseau devient indisponible pour cause de défaillance ou de manque d'énergie, cela ne change rien ou presque pour les autres noeuds qui vont se réorganiser et continuer leurs communications. Contrairement aux réseaux mobiles sans infrastructure où tout dépend de l'état des stations de base pour communiquer.

1.2.2.1. Les caractéristiques des réseaux Ad Hoc

Les réseaux ad hoc sont caractérisés principalement par :

- **Une topologie dynamique** : La topologie des réseaux ah hoc changent rapidement, et aléatoirement , ceci est causée par la mobilité arbitraire des noeuds du réseau. Le changement de la topologie change les routes entre les noeuds et provoque la perte de paquets. La (figure 1.3) illustre la topologie d'un réseau ad hoc avant et aprés le déplacement d'un noeud.



(Topologie avant le déplacement du noeud 5)

(Topologie après le déplacement du noeud 5)

Figure 1.3 : Changement de la topologie d'un réseau ad hoc.

1. **Une Bande passante limitée :** Les noeuds dans les réseaux ad hoc utilisent une

technologie de communication sans fil, malgré des progrès très importants la bande passante reste modeste comparée aux technologies des réseaux filaires.

2. **Des contraintes d'énergie** : Les noeuds mobiles dans les réseaux ad hoc sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables, la consommation d'énergie devient alors un problème important.

3. **Absence d'infrastructure :** Les réseaux ad hoc ne dépendent d'aucune infrastructure préétablie, ceci rend la gestion du réseau plus complexe.

1.2.2.2. Les applications des réseaux Ad Hoc

Les réseaux ad hoc sont rapides et faciles à déployer, ils sont particulièrement intéressants pour les applications militaires ou l'installation d'infrastructure fixe est souvent impossible, ils peuvent être aussi utilisées dans :

- **Les opérations de recherche et de secours** : en cas de tremblement de terre, de feux ou d'inondation, dans le but de remplacer rapidement l'infrastructure détruite.

- **L'informatique embarquée** : dans des véhicules communiquant par exemple.

- **Les entreprises** : dans le cadre d'une réunion ou d'une conférence.

- **Les gares et aéroports** : pour la communication et la collaboration entre les membres du personnel.

D'une façon générale, les réseaux ad hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau fixe est trop contraignant, soit parce qu'il est difficile à mettre en place, soit parce que la durée d'installation du réseau ne le justifie pas.[14].

1.3. La communication dans les réseaux Ad Hoc

Les signaux envoyés par les interfaces sans fil s'atténuent au fur et à mesure qu'ils s'éloignent de leur émetteur, un noeud dans les réseaux ad hoc ne peut donc pas communiquer avec un autre s'il est situé trop loin de lui, il doit alors passer par des noeuds intermédiaires

pour atteindre la destination désiré . La figure 1.4 illustre la topologie d'un réseau ad hoc composé de cinq noeuds *(A,B,C,D,E)* , si le noeud *A* veut communiquer avec le noeud *D* qui n'est pas dans sa porté de transmission, il doit passer par les noeuds intermédiaires *B* et *E*, on parle alors de communication « Multi-saut » ou « Multi-hop ».

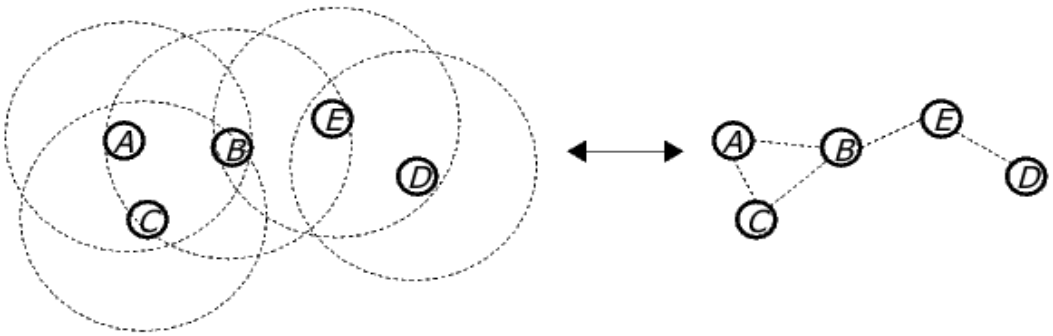


Figure 1.4 : Communication entre deux noeuds dans un réseau ad hoc

Dans la figure (1.4) le noeud B peut communiquer directement avec les noeuds A, C et E, ces noeuds sont dit : « **Voisins** ». En générale un paquet dans le réseau ad hoc doit traversé plusieurs noeuds avant d'atteindre sa destination.

La technique la plus basique pour réaliser une communication entre un noeud source et un noeud destination dans les réseaux ad hoc est l'inondation, ou le noeud source va transmettre les paquets à tous les noeuds voisins, chaque noeud mobile ré-émet a son tour les paquets qu'il reçoit à ses voisins jusqu'à ce qu'ils arrivent à la destination. L'inondation consomme beaucoup de ressources (bande passante et énergie) et n'est pas adapté pour les réseaux ad hoc.

1.3.1. Le routage dans les réseaux Ad Hoc

Généralement, le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné [14]. Dans les réseaux ad hoc il existe aucune infrastructure fixe ou élément centrale qui peut gérer le routage. Chaque noeud doit donc participer à un protocole de routage qui lui permet de communiquer avec les autres noeuds du réseau.

La stratégie (ou le protocole) de routage est utilisée dans le but de découvrir les chemins qui existent entre les noeuds d'un réseau. 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. [15].

A cause de la mobilité des noeuds dans les réseaux ad hoc il est très difficile de localiser une destination à un instant donné, les protocoles de routage conçus pour des réseaux statiques sont inadaptés pour ce type de réseaux. Un protocole de routage pour les réseaux ad hoc doit organisé le réseau et prendre en compte les limitations existantes, il doit construire et maintenir les routes entre les différents noeuds et s'adapter à la topologie changeante et imprévisible.

Un protocole de routage conçu pour les réseaux ad hoc :

- **Doit offrir un support de communication fiable** : le protocole doit permettre d'effectuer des échange de données fiables entre les différents noeuds du réseau,
- **Nécessite une distribution des opérations** : le protocole doit être entièrement distribué,
- **Doit avoir un minimum de paquets de contrôle** : les paquets de contrôles doivent être garder le minimum possible car elles consomment une précieuse bande passante et causent des collisions avec les paquets de données,
- **Doit offrir de bonne performance** : en terme de délai de livraison des paquets de données,
- **Doit Conserver les ressources** : le protocole doit optimiser l'utilisation des ressources comme la bande passante, l'énergie et le processeur.

1.4. Le routage dans le modèle OSI

Le modèle de référence de connexion entre systèmes ouverts OSI [9] (reference model of Open Systems Interconnection) a pour objectif de permettre la création de logiciel réseau modulaires et réutilisables indépendamment des techniques mises en oeuvre. Le modèle OSI propose une architecture composée de sept couches (figure 1.5). Chaque couche propose un service aux couches qui lui sont adjacentes.

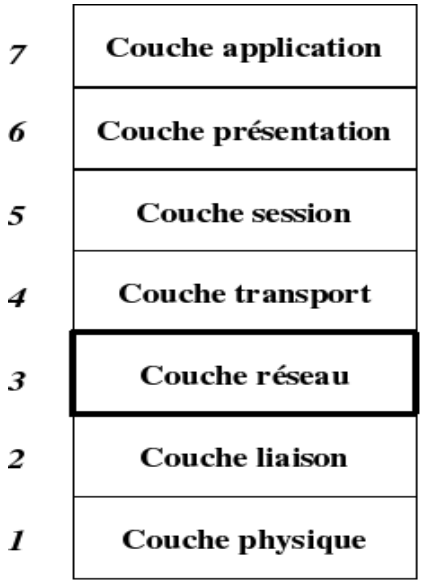


Figure1.5 : Le modèle OSI.

- La ***couche physique*** sert à transmettre les données via un canal de communication et comprend le matériel nécessaire à la réalisation de cette fonction,
- La ***couche liaison*** a pour fonction initiale la détection d'erreurs dans la couche physique,
- La ***couche réseau*** permet l'échange de données entre les noeuds du réseau,
- La ***couche transport*** permet de faire parvenir les données aux applications exécutées par les noeuds,
- La ***couche session*** est l'interface utilisateur du réseau,
- La ***couche présentation*** s'occupe des détails relatifs aux interfaces comme, par exemple, les imprimantes, les formats de fichiers et les écrans,
- La ***couche application*** contient les informations sur les applications réparties sur le réseau.

Les protocoles de routage sont associés a la couche réseau du modèle OSI car c'est dans cette couche qu'on détermine les routes qui doivent être emprunter par les paquets de données.

1.5. Conclusion

On a présenté dans ce chapitre les réseaux mobiles ad hoc. Ils se distinguent des autres réseaux par leur grande flexibilité d'emploi et leur grande robustesse mais aussi par des

contraintes supplémentaires tel que la limitation d'énergie et l'absence d'élément central pour gérer le réseau. On a aussi présenté le problème de routage qui est le plus important dans ce type de réseau, et les difficultés posées par la mobilité des noeuds. Dans le chapitre suivant nous allons présenter en détail quelques protocoles de routage développés pour les réseaux ad hoc.

Chapitre 2
Les protocoles de routage dans les réseaux Ad Hoc

2.1. Introduction

Ces dernières années plusieurs protocoles de routage pour les réseaux ad hoc on été développés, ces protocoles essayent de maximiser les performances en minimisant le délai de livraison des paquets, l'utilisation de la bande passante et la consommation d'énergie. Dans ce chapitre nous allons présenter une classification des protocoles de rouage existants et présenter en détail le fonctionnement de quelques protocoles toute en indiquant leurs avantages et leurs inconvénients.

2.2. Classification des protocole de routage Ad Hoc

Selon la manière de création et de maintenance des routes on peut les classer les protocoles de routage pour les réseaux ad hoc comme suit : protocoles Table-driven, protocoles On-demand et protocoles hybrides.

2.2.1. Les protocoles Table-driven

Le principe des protocoles Tabledriven est de maintenir à jour des tables de routage qui

indiquent les routes vers chaque destination du réseau. Les protocoles de routage Table-driven sont basées sur deux méthodes utilisées dans les réseaux filaires, la méthode État de Lien (Link State) et la méthode Vecteur de Distance (Distance Vector).

La méthode Link State : Dans la méthode Link State chaque noeud diffuse périodiquement (par inondation) l'état des liens avec ses voisins a tous les noeuds du réseau, chaque noeud maintien alors une vue globale de la topologie du réseau ce qui lui permet de calculer les routes pour atteindre chaque destination. On inonde aussi le réseau quand il y a un changement dans l'état des liens. Cette méthode permet de trouver rapidement des alternatives pour transmettre les paquets en cas de coupure d'une route, on peut aussi utiliser simultanément plusieurs routes pour atteindre la même destination. Le problème avec cette méthode est que la quantité d'informations à stocker et diffuser peut devenir considérable si le réseau contient un grand nombre de noeuds.

La méthode Distance Vector : Dans la méthode Distance Vector chaque noeud transmet à ses voisins la distance (nombre de noeuds) qui le sépare de chaque destination dans le réseau et le noeud voisin à utiliser pour atteindre cette destination. En se basant sur les informations reçues depuis tous ses voisins, chaque noeud calcul le chemin le plus court vers n'importe quel destination dans le réseau. Si la distance séparant deux noeuds change on répète le processus de calcul. La méthode Distance Vector évite l'inondation, mais elles est moins précise que la méthode Link State il est aussi difficile de trouver des routes alternatives en cas de coupure d'une route.

Les liens entre les noeuds dans les réseaux ad hoc changent rapidement. Les deux méthodes précédentes vont engendrer énormément de paquets de contrôle (inondation des états des liens, et transmission des vecteurs de distance) ce qui les rend inadaptés pour les réseaux ad hoc.

Les protocoles Table-driven calculent les routes à l'avance ils disposent donc des routes immédiatement vers les destinations du réseau. Le problème avec ces protocoles c'est qu'ils chargent le réseau avec les paquets de mise à jour des tables de routage même si le réseau n'est pas utilisé. Parmi les protocoles Table-driven les plus connus on peut cité : DSDV, GSR et FSR.

2.2.2. Les protocoles on-demand

A l'opposé des protocoles Table-driven les protocoles On-demand créent et maintiennent les routes selon les besoins, si un noeud veut envoyer un paquet à une destination à la quelle il ne connaît aucune route, il lance un procédure de découverte de route globale qui va inonder le réseau avec un paquet de requêtes et lui fournir les informations nécessaires pour atteindre cette destination. Si la route devient invalide une autre procédure de découverte de route est lancée.

Les protocoles On-demand réduise la charge des paquets de contrôle comparés aux protocoles Table-driven, surtout si le réseau est très dynamique. Le problème avec ces protocoles c'est qu'ils ont un délai initial avant de commencer la transmission des paquets provoquer par la procédure de découverte de route, aussi la redécouverte de route en cas de coupure génère une charge supplémentaire. Parmi les protocoles On-demand on peut cité : DSR et AODV .

2.2.3. Les protocoles Hybrides

Les protocoles hybrides essaient de combiner les deux approches précédentes pour bénéficier de leurs avantages, ils utilisent un protocole Table-driven, pour connaître les voisins les plus proches, dans le but de réduire le délai et un protocole On-demand au- delà de cette zone prédéfinie dans le bute de réduire la charge des paquets de contrôle.

Les protocoles hybrides cumulent aussi les inconvénients des protocoles Table-driven et des protocoles On-demand à savoir les paquets de contrôle périodique, et le délai de découverte de route. Parmi les protocole hybrides on peut cité le protocole CBRP .

2.3. Quelques protocoles de routage pour les réseau ad hoc

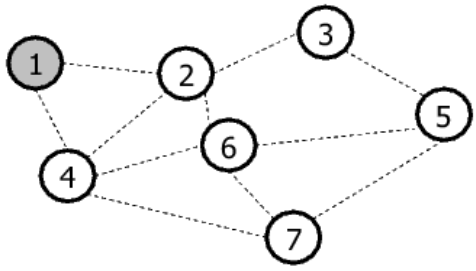
2.3.1. Le protocole DSDV

DSDV (*Destination-Sequenced Distance-Vector*) est un protocole de routage Table-driven. Chaque noeud dans DSDV garde une table de routage qui donne pour chaque destination accessible dans le réseau :

- Le noeud voisin à utilisé pour atteindre cette destination,
- Un numéro de séquence qui est envoyé par le noeud destinataire et qui permet de distinguer les nouvelles routes des anciennes,

- Le nombre de sauts (noeuds intermédiaires) pour atteindre cette destination.

Périodiquement chaque noeud dans le réseau diffuse par inondation un paquet de mise à jour des tables de routage qui inclue les destinations accessibles et le nombre de sauts exigés pour atteindre chaque destination avec le numéro de séquence lié à chaque route. Des paquets de mise à jour sont aussi diffusés immédiatement si il y a un changement dans la topologie du réseau afin de propager les informations de routage aussi rapidement que possible.



A la réception d'un paquet de mise à jour, chaque noeud le compare avec les informations existantes dans sa table de routage. les routes les plus récentes (qui ont le plus grand numéro de séquence) avec la distance la plus courte sont gardées, les autres sont simplement ignorées.

La figure (2.1) illustre la topologie d'un réseau ad hoc a un instant donné et la table de routage correspondant au noeud (1) dans le protocole DSDV.

Destination	**Prochain saut**	**Distance**	**Numéro de séquence**
4	*4*	*1*	*10*
3	*2*	*2*	*32*
6	*2*	*2*	*88*
5	*4*	*3*	*19*
2	*2*	*1*	*12*
7	*4*	*2*	*57*

Figure 2.1 : A gauche topologie du réseau, a droite table de routage du noeud (1) dans le protocole DSDV.

DSDV fourni a tout moment des routes valables vers toutes les destinations du réseau, mais l'inondation des paquets de mise à jour (périodique et en cas de changement de topologie) cause une charge de contrôle importante au réseau.

2.3.2. Le protocole GSR

Dans le protocole Table-driven GSR (*Global State Routing*) chaque noeud maintient une table de la topologie qui l'informe sur la topologie globale du réseau et lui permet de calculer les routes pour atteindre chaque destination. GSR utilise la méthode Link State des réseaux filaires et l'améliore en supprimant le mécanisme d'inondation des paquets de contrôle.

Un noeud dans GSR maintient :

- Une liste de voisins,
- Une table de topologie qui contient les informations sur les liens du réseau,
- Une table des noeuds suivants qui indique le noeud à utiliser pour atteindre chaque destination,
- Une table de distance qui contient la plus courte distance pour chaque destination.

Comme dans la méthode Link State chaque noeud dans GSR construit sa table de topologie basé sur les informations de liens reçus, et l'utilise pour calculer les distances minimales qui le sépare des autre noeuds du réseau. Dans GSR la table de topologie entière de chaque noeud est échangé périodiquement uniquement avec les voisins au lieu de la diffusé par inondation dans tout le réseau.

GSR réduit la charge des paquets de contrôle en évitant l'inondation et assure plus de précision, concernant les données de routage. Le problème de GSR est la taille des ses paquets de mise à jour (Table de topologie) qui peut devenir considérable si le réseau contient un grand nombre de noeuds.

2.3.3. Le protocole FSR

Une caractéristique observé dans l'oeil de poisson (Fish eye) est qu'il distingue les choses en détail au centre, et que sa précision se dégrade en s'éloignant du point central.

FSR (*Fisheye State Routing*) est un protocole Table-diriven, il minimise la charge des paquets de mise à jour des tables de routage du protocole GSR en utilisant la technique de l'oeil de poisson. Les paquets de mise à jour dans FSR, ne contiennent pas l'information sur tous les noeuds du réseau, il échange les informations sur les noeuds les plus proches plus fréquemment qu'il le fait sur les noeuds les plus lointains, il réduit ainsi la taille des paquets de mise à jour.

Un noeud dans FSR a donc des informations précises sur les noeuds proches (figure 2.2 : Zone 1), la précision des informations diminue quand la distance augmente (Figure 1.5: Zone 2 et 3).

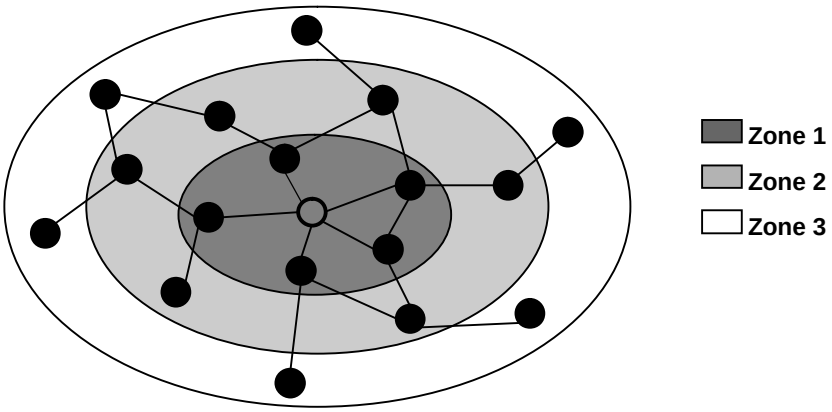


Figure 2.2 : Précision des informations d'un noeud dans FSR
en utilisant la technique oeil de poisson.

Malgré qu'un noeud dans le réseau n'a pas des informations précises sur les noeuds éloignés (surtout si le réseau est très dynamique), les paquets peuvent être transmis correctement car l'information sur les routes devient de plus en plus précise quand les paquets se rapprochent de leurs destinations.

2.3.4. Le protocole AODV

Le protocole AODV (*Ad hoc On-demand Distance Vector*) appartient a la famille des protocoles On-Demand, il est basé sur deux mécanismes, la *découverte de route* et la

maintenance de route. La découverte de route permet de trouver une route pour atteindre une destination et cela en inondant un paquet de requête dans tout le réseau. La maintenance de route permet de détecter et signaler les coupures de routes provoquées éventuellement par la mobilité des noeuds. AODV n'utilise pas des mises à jour périodique, les routes sont découvertes et maintenues selon les besoins.

Chaque noeud intermédiaire qui se trouve dans la route entre un noeud source et un noeud destination doit garder une table de routage qui contient :

- L'adresse de la destination.
- Le noeud suivant à utiliser pour atteindre la destination.
- La distance en nombre de noeud : C'est le nombre de noeud nécessaire pour atteindre la destination.
- Le numéro de séquence destination : Il permet de distinguer les nouvelles routes des anciennes.
- Le temps d'expiration de l'entrée de la table : C'est le temps au bout du quel l'entrée est valide.

AODV utilise trois types de messages pour créer et maintenir les routes, le RREQ (Route Request) pour demander une route, le RREP (Route Reply) pour répondre à une requête de demande de route, et le RERR (Route Error) pour signaler une coupure de route.

2.3.4.1 La découverte de route de route

Si un noeud dans AODV veut communiquer avec une destination à la quelle il ne possède pas de route valide, il inonde le réseau avec une requête de demande route (RREQ : Route REQuest) qui contient : son adresse, son numéro de séquence, l'adresse de la destination, la dernière valeur connue du numéro de séquence de la destination, et le numéro de la requête. La réponse à la requête (RREP) est retournée par la destination ou par un autre noeud qui possède une route à la destination. La RREP contient l'adresse du noeud source, l'adresse de la destination, le numéro de séquence de la destination et le nombre de sauts pour atteindre la destination. La figure suivante illustre l'inondation de la requête de demande de route par le noeud source et le renvoie de la réponse par le noeud destination dans AODV.

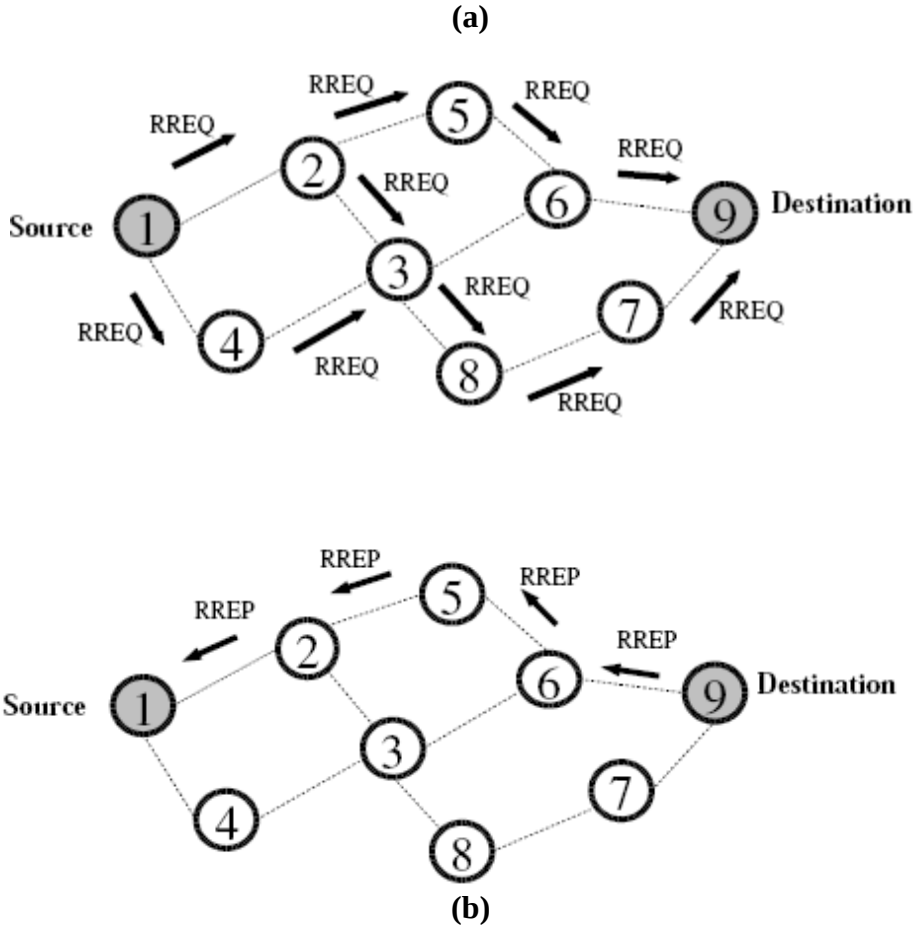


Figure 2.3 : (a) Inondation de RREQ, (b) revoie du RREP dans AODV.

Si le noeud émetteur de la requêtte ne reçoit pas de réponse RREP pendant une certaine période (appelé RREP_WAIT_TIME) il rediffuse une nouvelle requêtte. Si la requête RREQ est rediffusée un certain nombre de fois (RREQ_RETRIES) sans recevoir de réponse une erreur est déclenchée.

Un noeud intermédiaire (se trouvant entre un noeud source et un noeud destination) qui rediffuse une requête de route (figure 2.3 (a)), sauvegarde l'adresse du noeud source qui a envoyé la requêtte la première fois et l'adresse du noeud voisin qui lui a transmit la requêtte, cette information est utilisée pour reconstruire la route inverse qui sera traversée par la réponse de route (RREP) (figure 2.3 (b)).

2.3.4.2 La maintenance de route

Chaque noeud dans AODV maintient une liste des ses voisins. Avec une cadence d'une fois par seconde chaque noeud va transmettre un message HELLO, Si un noeud ne reçoit pas d'un voisin trois messages HELLO consécutifs (pas de messages pendant trois secondes) le lien avec le voisin est considéré invalide.

Si un lien entre deux noeuds est invalide (à cause de la mobilité ou la défaillance d'un noeud), les noeuds utilisant se lien sont prévenus par un message d'erreur (RERR) , ils vont alors diffusés une autre requêtte. La figure suivante illustre la coupure d'un lient entre deux noeuds et l'envoi du RERR dans AODV.

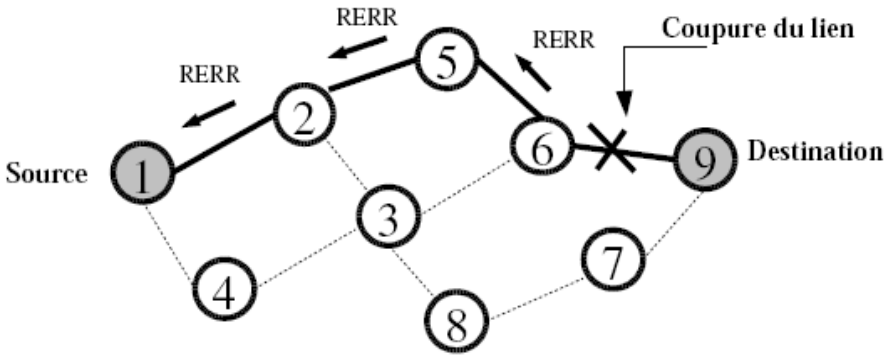


Figure 2.4 : Coupure de route et envoie du RERR dans AODV.

Le protocole AODV utilise l'inondation pour découvrir les routes, si le réseau est très utilisé il peut générer un trafic de contrôle énorme. Aussi AODV étant un protocole On-Demand il a un délai initial avant de commencer la transmission des paquets.

2.3.5. Le protocole DSR

DSR (*Dynamic Source Routing*) est un protocole On-demand semblable au protocole AODV, il utilise une technique appelé « Source Routing » dans laquelle l'émetteur (la source) indique la route complète par lequelle un paquet doit passer pour atteindre sa destination, cette route est insérée dans l'entête du paquet. Les noeuds intermédiaires entre le noeud source et le noeud destination n'ont pas besoins de maintenir à jour les informations sur la route traversé puisque la route complète est insérée dans l'entête du paquet.

Si un noeud dans DSR veut communiquer avec une destination à laquelle il ne possède pas de route, il inonde le réseau avec un paquet de requête (RREQ) similaire a celui de AODV. Chaque noeud qui reçoit la requête et qui ne possède pas de route à la destination demandée insère son adresse dans le paquet RREQ et le diffuse à ses voisins. La réponse à la

requête (RREP) est retournée par la destination ou par un autre noeud qui possède une route à la destination (figure 2.5).

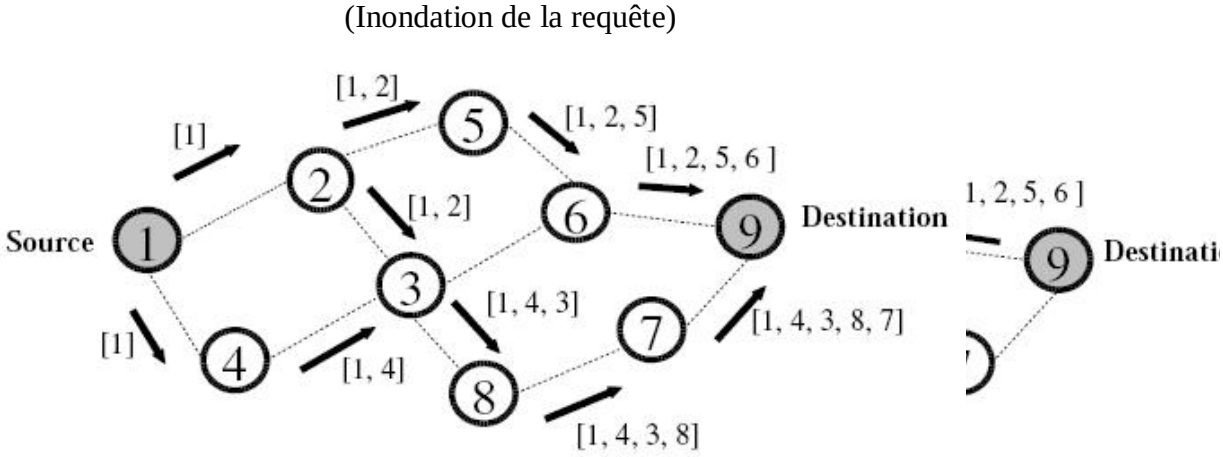


Figure 2.5 : Découverte de route dans DSR.

Les routes dans AODV sont construites en traversant la route inverse ver la source (de la destination à la source), dans DSR les routes sont construites quand la requête traverse le réseau vers la destination (de la source à la destination).

Si un noeud reçoit un paquet de données, et le lien a utiliser pour retransmettre ce paquet est coupé (coupure de route) le noeud envoie un message d'erreur de route (RERR) semblable à celui de AODV au noeud source. Le noeud source va lancer une autre requête de découverte de route pour atteindre la destination.

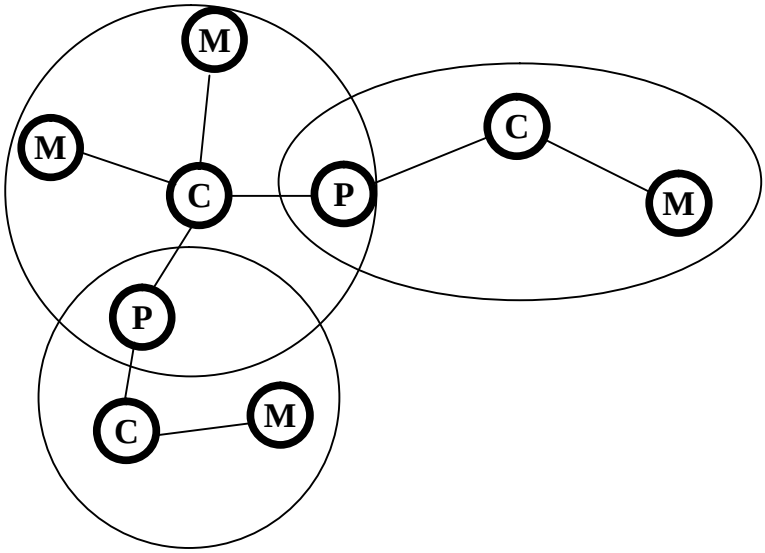
Le protocole DSR comme le protocole AODV utilise l'inondation pour découvrir les routes ce qui généré un trafic de contrôle énorme quand le réseau est très utilisé. Aussi la taille des paquets de données dans DSR devient très grande quand le nombre de noeuds dans réseau est grand, puisque les paquets doivent porter les adresses de chaque noeud dans la route traversée. DSR a aussi un délai avant de commencer la transmission des paquets provoqués par la procédure de découverte de route.

2.3.6. Le protocole CBRP

Le protocole de routage CBRP (*Cluster Based Routing Protocol*) est un protocole Hybride. Il utilise deux niveaux hiérarchiques pour effectuer le routage. Les noeuds dans

CBRP sont rassemblés en groupes appelés « Clusters », avec un chef au centre de chaque groupe appelé « Clusterhead ». Les clusters sont relié entre eux par des noeuds passerelles qui se trouvent à l'extrémité des clusters. CBRP utilise un protocole Table-driven pour maintenir les membres du cluster, et un protocole On-demand pour découvrir les noeuds en dehors du cluster.

Le diamètre d'un cluster dans CBRP est de deux sauts, un exemple d'assemblage en cluster est illustré dans la figure suivante :



C : Chef M : Membre P : Passerelle.

Figure 2.6 : Assemblage en clusters dans CBRP.

Le principe de formation des clusters dans CBRP est le suivant :

1. Les noeuds s'échangent des messages «Hellos» pour connaître leurs voisinages.
2. Les noeuds vont élire le noeud avec le plus petit identifiant chef de cluster.
3. Un noeud qui n'a pas de chef de cluster comme voisin devient chef de cluster.
4. Le chef de cluster prend tous ses voisins comme membres de son cluster.

Pour éviter des changements fréquents dans les structures des clusters, les chefs sont maintenus le plus longtemps possible. Un chef de cluster n'est pas changé même si un nouveau membre (non chef) de sont cluster a un identifiant plus petit. Si par la suite d'un changement de topologie deux chefs de clusters ont un lien direct entre eux, il faut reconstruire les clusters et le noeud possédant l'identifiant le plus faible est élu comme chef.

Chaque noeud dans CBRP maintient une table de voisinage qui indique le statut (membre, ou chef) de ses voisins. Un chef de cluster maintient aussi une liste des noeuds membres de son cluster, et une liste des chefs de clusters voisins avec les adresses des noeuds passerelles pour les atteindre.

Quand un noeud source dans CBRP veut envoyer un paquet de données à un noeud destination, il transmet au chef de son cluster un paquet de requête qui contient l'adresse de la destination désirée, le chef de cluster va insérer son adresse dans le paquet et le diffuse aux chefs de clusters voisins. Quand un chef de cluster reçoit un paquet de requête de route, il vérifie dans sa liste des noeuds membre si le noeud destination appartient a son cluster. Si c'est le cas il répond au noeud source en inversant la route enregistré dans le paquet de requête, si non il enregistre son adresse dans le paquet de requête et le diffuse aux chefs de clusters voisins. Si le noeud source ne reçoit pas de réponse au cours d'une certaine période de temps, il envoi une autre requête à son chef.

CBR utilise la technique « Source Routing ». Comme dans DSR la route complète vers la destination est insérée dans l'entête du paquet . Le routage par source permet à CBRP de faire des des raccourcissements de routes s' il y a un changement de topologie. Quand un noeud reçoit un paquet il essaye de l'envoyer au noeud voisin le plus lointain dans la route, ce qui raccourci la route utilisée. Si un noeud dans CBRP reçoit un paquet de données et le lien ver le noeud suivant est coupé (coupure de route) il envoie un message d'erreur à la source et essaye de réparer la route localement. Il vérifie s' il peut atteindre le noeud suivant dans la route, ou le noeud qui vient après le noeud suivant par un autre voisin, si c'est le cas le paquet est envoyé sur la route réparée.

CBRP diffuse les requêtes de route seulement au chefs de cluster. il réduit ainsi la charge provoquée par l'inondation des requêtes de route. Mais la formation et la maintenance des clusters engendrent une charge supplémentaire au réseau. Aussi les chef de clusters sont responsables de l'acheminement des données se qui peut provoquer des goulets d'étranglement.

2.3.7. Le protocole ARA

ARA *(Ant-Colony-Based Routing Algorithm)* est un protocole On-demand. Il se base sur les techniques d'optimisation par colonie de fourmis (ACO). ARA utilise des agents

fourmis pour découvrir et maintenir les routes entre les noeuds du réseau.

L'idée de base des algorithme d'optimisation par colonie de fourmis est inspiré du comportement des fourmis réel quand elles recherchent la nourriture. Les fourmis réussissent toujours à trouver le chemin le plus court pour atteindre la nourriture.

La figure 2.7 illustre un scénario avec deux chemins possibles pour atteindre la nourriture. Quant les premières fourmis arrivent a l'intersection des deux chemins il vont choisir aléatoirement le chemin à prendre. Les fourmis en parcourant le chemin vont déposé une matière appelée phéromone, cette matière indique aux autres fourmis le chemin à utiliser. Le phéromone s'évapore avec le temps. la quantité de phéromone déposée sur le deuxième chemin va devenir plus importante que celle déposée sur le premier chemin puisque les fourmis parcourent le deuxième chemin plus vite. Avec le temps toutes les fourmis vont utiliser le deuxième chemin qui est le chemin le plus court.

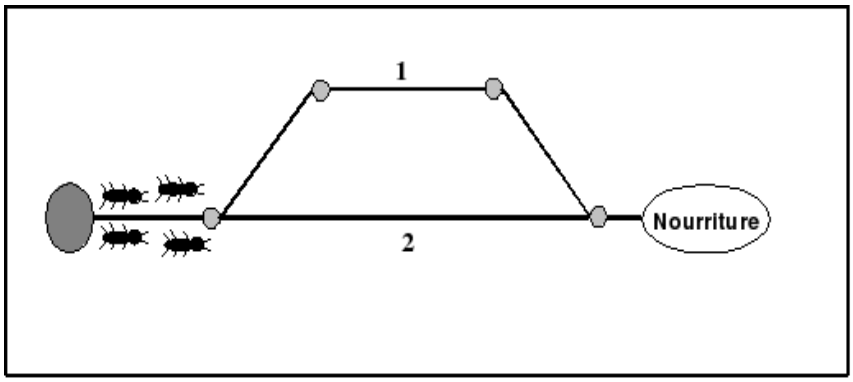


Figure 2.7 : Recherche de nourriture par une colonie de fourmis

Ce comportement est utilisé dans ARA pour trouver la route la plus courte entre un noeud source et un noeud destination dans le réseau**.**

ARA se compose de trois phases la découverte de routes, la maintenance de routes et le traitement des coupure de routes.

2.3.7.1. La découverte de route dans ARA

Dans la phase de découverte de routes de nouvelles routes sont crées. Dans cette phase deux types de fourmis sont utilisées, les *forward ant (FANT)* pour maintenir le phéromone vers le noeud destination et a *backward ant (BANT)* pour maintenir le phéromone vers le

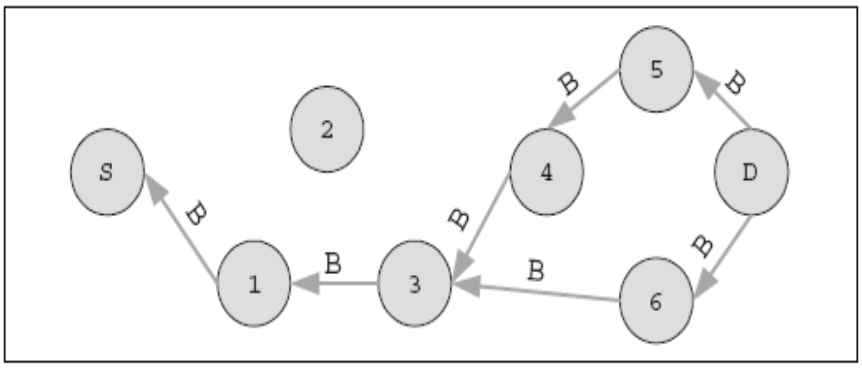
noeud source. Les fourmis sont en réalité de simples paquets avec un numéro de séquences uniques qui permet de les distinguer.

Un noeuds source qui cherche une route pour atteindre un noeud destination va diffuser un forward ant (FANT) à ses voisins. Un noeud qui reçoit un FANT pour la première fois crée un enregistrement dans sa table de routage, qui contient les champs suivant :

- **Destination address** : C'est l'adresse du noeud source.
- **Next hop** : C'es l'adresse du voisin par lequel il a reçu le FANT.
- **Pheromone value** : C'est la valeur du phéromone, qui dépend du nombre de noeuds que le FANT a traversé. Elle représente la trace du phéromone laissé par les fourmis.

Après avoir créé l'enregistrement, le noeud transmet le FANT à ses voisins (figure 2.8 (a)). Les FANT dupliqués sont identifiés et détruits grasse à leurs numéros de séquences uniques. Quand le FANT atteint la destination un *backward ant (BANT)* est créé et envoyé au noeud source (figure 2.8 (b)), le BANT effectue les mêmes taches que le FANT. Quand le noeud source reçoit le BANT de la destination, la route est établie et les paquets de données peuvent être envoyer.

(a) Envoi du backward ant par le noeud source.



(b) Envoi du forward ant par le noeud destination.

Figure 2.8 : Découverte de route dans ARA.

Les deux figures précédentes illustrent la phase de découverte de route dans ARA, le noeud source a une seule trace de phéromone pour atteindre la destination (en utilisant le noeud 1), mais le noeud destination a deux traces de phéromone pour atteindre la source (en utilisant les noeuds 5 et 6). ARA peut donc créer plusieurs routes dans une seule phase de découverte de route (multi-path routing).

2.3.7.2. La maintenance de routes

Les routes dans ARA sont maintenues durant la communication. Après que les fourmis FANT et BANT ont établi les traces de phéromones entre la source et la destination, les paquets de données transmis sont utilisés pour maintenir ces traces.

Quand un noeud intermédiaire *Ni* retransmet un paquet de données pour une destination *D*, en passant par le voisin *Nj*, il augmente la quantité de phéromone dans l'enregistrement *(D, Nj, Ph)*, la route vers la destination est donc renforcée par les paquets de données. Aussi le voisin Nj augmente la quantité de phéromone dans l'enregistrement *(S, Ni, ph)*, Donc la route vers la source est aussi renforcée. L'évaporation du phéromone est simulée en diminuant régulièrement la quantité de phéromone dans tous les enregistrements. Avec le temps le chemin le plus court entre la source et la destination va être utilisé.

2.3.7.3. Traitement des coupure de routes.

Les coupures de route sont provoquées par la mobilité des noeuds dans le réseau, Si un noeud ne peut pas retransmettre un paquet vers sa destination (à cause d'une coupure d'un lien) , le paquet est retourné vers le noeud source avec un message d'erreurs de route *(ROUTE ERROR)*. Le voisin en recevant le message d'erreurs va désactiver ce lien en donnant la valeur 0 a la quantité de phéromone dans la table de routage. Il cherche ensuite dans sa table de routage une autre alternative pour retransmettre le paquet. Si le paquet ne peut pas atteindre sa destination, le noeud source lance une autre phase de découverte de route.

Le protocole ARA permet la découverte de plusieurs routes pour la même destination, s'a permet de trouver rapidement des alternatives en cas des coupures d'une route. Aussi le protocole s'adapte au changement de topologie puisque la route la plus courte entre la source et la destination va émerger progressivement. Mais la phase de découverte de route engendre un trafique considérable puisque les FANT sont inondée dans le réseau pour chercher les routes vers la destination, en plus ARA a un délai avant de comancer la transmission des paquets provoqués par la phase de découverte de route.

2.3.8. Le protocole Ant-AODV

Le protocole Ant-AODV est une combinaison du protocole AODV et des capacités de découverte de routes des agents fourmis. chaque noeud dans ce protocole maintient une table de routage qui indique pour chaque destination le noeud voisin à utiliser pour atteindre la

destination. Cette table est maintenu par les agents fourmis qui se déplacent dans le réseau et découvre les routes entre les noeuds (figure 2.9). Les agents fourmis sont de simples paquets avec un numéro de séquence unique qui permet de les distinguer.

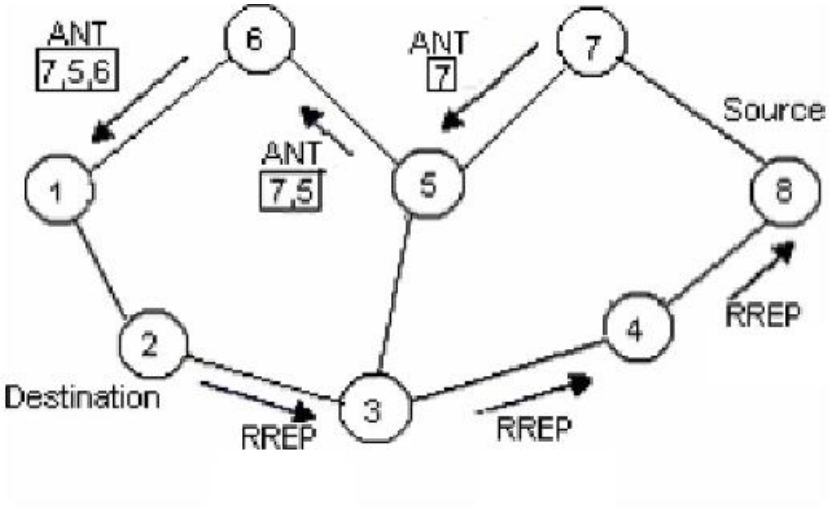


Figure 2.9 : Découverte de route dans Ant-AODV.

Si un noeud veut envoyer des paquets a une destination pour la quelle ils n'a pas de routes récentes dans sa table de routage, il lance une découverte de route en utilisant le protocole AODV. Le noeud inonde le réseau avec la requête de demande route RREQ qui contient son adresse et l'adresse de la destination. La réponse à la requête (RREP) est retournée par la destination ou par un autre noeuds qui possède une route à la destination.

L'utilisation des agents fourmis avec AODV permet de fournir plus de routes aux noeuds ce qui par la suite réduit le coup de la découverte de routes, même si un noeud lance une requête pour une destination pour laquelle il n'a pas de routes récentes la probabilité de recevoir une réponse rapidement de ses voisins est grande ce qui a pour résultat la réduction du délai de découverte de routes.

2.4. Tableaux récapitulatifs

Nous décrivons dans les deux tableux suivants les différentes classes de protocoles de routage pour les réseaux ad hoc ainsi que les protocoles de routage présentés dans ce chapitre.

Classes	*Caractéristiques*	*Avantages*	*Inconvénients*
Table-Driven	- Calculer les routes a l'avance.	- Transmission immédiate des données.	- Utiliser beaucoup de paquets de contrôles.
On-Demand	- Calculer les route a la demande.	- Utiliser moins de paquets de contrôles.	- Délai initial avant de commencer la transmission des données.
Hybride	- Combinaison des deux approches précédentes.	- Bénéficier des avantages des deux approches précédentes.	- Cumuler les inconvénients des deux approches précédentes.

Tableau 2.1 : Les classes des protocoles de routage pour les réseaux ad hoc.

Protocoles	*Classe*	*Avantages*	*Inconvénients*
DSDV	Table-Driven	- Fournit a tout moment des routes valables ver toutes les destinations du réseau.	- L'inondation des paquets de mise a jour cause une charge de contrôle importante au réseau.
GSR	Table-Driven	- Évite l'inondation en transméttant les paquets de mise a jour seulement aux voisins.	- Taille considérable des paquets de mise a jour.
FSR	Table-Driven	- Minimise la charge des paquets de mise a jour des table de routages du protocole GSR en utilisant la technique de l'oeil de poisson.	- Taille considérable des paquets de mise a jour.
AODV	On-Demand	- Découvre les route a la demande en inondant le réseau avec un paquets de requête.	- Délai initial avant de commencer la transmission des données.

Protocoles	*Classe*	*Avantages*	*Inconvénients*
DSR	On-Demand	- Découvre les route a la demande en inondant le réseau avec un paquets de requête. - Les paquets de données peuvent être redirigé pendant leurs transmission.	- Délai initial avant de commencer la transmission des données. - La taille des paquets de données très grande quand le nombre de noeud dans réseau est grand.
CBRP	Hybride	- Diffuse les requête de route seulement au chefs de cluster se qui permet de réduire la charge provoqué par l'inondation des requête de route.	- La formation et la maintenance des cluster engendre une charge supplémentaire.
ARA	On-Demand	- Découvre les route a la demande en utilisant les techniques d'optimisation par colonie de fourmis. - Permet de trouver rapidement des alternative en cas des coupure d'une route.	- La phase de découverte de route engendre un trafique considérable.
Ant-AODV	Table-Driven	- Réduit le coup de la découverte de routes en combinant les fourmis et AODV.	- Les fourmis peuvent engendrer un trafic considérable.

Tableau 2.2 : Les protocoles de routage pour les réseaux ad hoc.

2.5. Conclusion

Nous avons présenté dans ce chapitre une classification des protocoles de routage pour les réseaux ad hoc. Ces protocoles peuvent se classer en trois grandes catégories. Les protocoles Tables-Driven qui réduisent le délai de livraison de paquets mais utilisent beaucoup de paquets de contrôle, les protocoles On-Demend qui réduisent l'utilisation des paquets de contrôle mais qui ont un délai de livraison des paquets élevé, et les protocoles Hybrides qui essayent de combiner les deux approches précédentes pour avoir de meilleurs

performances. Le problème est donc de trouvé un compromis entre le délai de livraison des paquets de données et l'utilisation des paquets de contrôle. Dans le chapitre suivant nous allons présenter la technologie agent ainsi que quelques protocoles de routage basé sur cette technologie.

Chapitre 3
Le routage à base d'agents dans les réseaux Ad Hoc

3.1. Introduction

L'agent et les systèmes multiagent sont une nouvelle technologie issue du domaine de l'intelligence artificielle, cette technologie est appliquée dans beaucoup de domaine tel que les systèmes distribués, les interfaces homme machine et les réseaux. Cette technologie est particulièrement intéréssante à utiliser dans des environnements de nature distribué et dynamique comme les réseaux ad hoc. Dans ce chapitre nous allons présenter la technologie des agents et des systèmes multiagent ainsi que quelques protocoles de routage pour les réseaux ad hoc basés sur les agents.

3.2. L'agent et les systèmes multiagents

Pour Jennings et Wooldridge [11],[18] un agent est une entité informatique *située* dans un environnement, qui est capable *d'agir* de manière *autonome* et *flexible* de façon à remplir ses objectifs, c'est à dire ceux qui lui ont été assignés lors de sa conception.

- L'aspect **situé** signifie que l'agent perçoit son environnement et qu'il est capable de le modifier.

- L**'autonomie**, signifie que l'agent est capable d'agir sans l'intervention directe d'êtres humains ou d'autres agents et qu'il est capable de contrôler ses propres actions et son état interne.

- La **flexibilité** caractérise un *agent intelligent*. Être *flexible* signifie que l'agent est capable :

de percevoir son environnement et répondre en temps voulu aux changements qui s'y produisent (*réactivité*); d'avoir un comportement dirigé par son but, d'être opportuniste et capable d'initiative quand c'est approprié (*proactivité*); et d'interagir, quand c'est approprié, avec d'autres agents artificiels ou humains de manière à compléter leur résolution de problème et d'aider les autres par leurs activités (*sociabilité*).

Un Système multiagent (SMA) est une application où les agents travaillent ensemble pour résoudre un problème commun, les possibilités de chaque agent prises séparément ne permet pas de le résoudre. Chaque agent possède donc des connaissances et savoir-faire limités, ce qui l'oblige à interagir avec d'autres pour mener à bien le projet commun.

Les SMA mettent en oeuvre des interactions complexes, comme la *coopération*, la *coordination* et la *négociation* [12]:

- Il y a **coopération** lorsque les agents travaillent ensemble vers un but commun.

- La **coordination** consiste à organiser les activités liées à la résolution d'un problème de manière à éviter les interactions néfastes et à exploiter les interactions bénéfiques.
- La **négociation** a pour objectif la gestion des conflits entre agents, c'est à dire qu'elle consiste à aboutir à un accord acceptable par l'ensemble des agents impliqués.

3.2.1 Les SMA et le routage dans les réseaux ad hoc

Le système de gestion de routage dans les réseaux ad hoc nécessite une distribution des opérations, car il n'existe aucun élément central fixe qui peut organisé le réseau, il nécessite la coopération entre les différents noeuds mobiles car un noeud mobile ne peut communiquer qu'avec les noeuds proches de lui , il a donc besoin de coopérer avec les autres noeuds mobiles qui vont l'aider à transmettre les informations. Le système de gestion de routage dans les réseaux ad hoc doit aussi être robuste, il doit pouvoir continuer a fonctinner normalement en cas de défaillance d'un ou plusieurs noeuds, il doit être intelligent, il doit choisir les meilleures routes pour transmettre les paquets et réagir rapidement au évènements qui peuvent intervenir. Les systèmes multiagent prennent en compte les aspects de coopération, d'autonomie, de distribution et d'intelligence ils nous semblent donc appropriés pour gérer le routage dans les réseaux ad hoc.

3.3. L'agent mobile

Un agent mobile est un agent possédant la capacité de se déplacer entre les machines du réseau. Un agent mobile qui s'exécute sur une machine du réseau peut suspendre sont exécution sur cette machine , se transférer à une autre machine du réseau et continuer son exécution. A un instant données un agent mobile est constitué de son code, de son contexte d'exécution et de ses données.

Les agents mobiles héritent de deux technologies, la technologie des agents, et la technologie du code mobile qui consiste à définir un code tel que le programme compilé dans ce code puissent être porter et exécuter sur n'importe quelle machine.

3.3.1. L'environnement d'exécution d'agents mobiles

Un environnement d'exécution d'agents mobiles fournit des primitives pour créer, lancer et exécuter un agent mobile [25]. Cet environnement est constitué d'un ensemble de programmes statiques qui s'exécutent sur les sites du système susceptibles d'accueillir des agents.

Les environnements d'exécution d'agents mobiles offrent plusieurs services de base permettant l'exécution d'un agent mobile sur un site, tel que :

- **La création d'un agent mobile** : Un agent mobile peut être créé localement sur la machine locale ou sur une machine distante. A sa création, un nom globalement unique doit être attribué à l'agent ce qui va permettre de localiser l'agent et de communiquer avec lui.

- **La migration d'un agent mobile :** La migration permet le transfert d'un agent en cours d'exécution d'un site à un autre à travers le réseau. Il existe deux types de migration, *la migration forte* permet à un agent de se déplacer quelque soit l'état d'exécution dans lequel il se trouve, dans ce type de migration l'agent se déplace avec son code, son contexte d'exécution et ses données, dans ce cas, l'agent reprend son exécution après la migration exactement là où elle était avant son déplacement. Et *la migration faible* qui ne fait que transférer avec l'agent son code et ses données, sur le site de destination, l'agent redémarre son exécution depuis le début en appelant la méthode qui représente le point d'entrée de l'exécution de l'agent, et le contexte d'exécution de l'agent est réinitialisée.

- **L'accès aux ressources locales :** L'agent mobile doit pouvoir accéder aux ressources de la

machine ou il se trouve. cela pose le problème du piratage des sites visités par les agents mobiles.

- **La communication entre les agents mobiles** : Un agent mobile doit pouvoir communiquer avec d'autres agents résidents dans la même machine ou avec des agents résidents dans d'autres machine distante.

3.3.2. Avantages des agents mobiles

Dans [10] on donne un ensembles de raisons qui justifient le choix d'utiliser les agents mobiles :

- **Réduire la charge du réseau** : les agents mobiles sont envoyés sur la machine où résident les données. Ainsi, c'est le calcul qui va aux données et non les données au calcul. Cela va permettre de réduire significativement, voir supprimer, les communications distantes et se contenter des interactions locales, ce qui réduit la charge du réseau.

- **Surmonter le temps de latence sur le réseau** : Les agents mobiles sont envoyés dans la zone d'activité, afin d'entreprendre des actions sur place. Cela leur permet de répondre immédiatement à des changements dans leur environnement et de surmonter le temps de latence du réseau.

- **S'exécuter de manière asynchrone et autonome** : Les agents mobiles s'exécutent indépendamment de leur plate-forme d'origine, c'est à dire un agent peut continuer à parcourir le réseau même sans interaction avec l'utilisateur qui la envoyer, cela permet a l'utilisateur qui a envoyé l'agent de déconnecter sa machine et préservé les ressources pendant le travail de l'agent. Cette caractéristique rend les agents mobiles très intéressants pour les réseaux ad hoc, ou la connexion entre les machines du réseau est très fragile et les ressource en bande passante et énergie sont limitées.

- **S'adapter dynamiquement aux environnements d'exécution** : Les agents mobiles ont la capacité de percevoir et de s'adapter dynamiquement a leur environnement.

- **Hétérogènes** : la multiplicité des systèmes augmente l'incompatibilité des langages de commandes et des formats de données. Les agents mobiles peuvent résoudre ce problème sous contrainte que les primitives du langage de l'agent soit exécutable sur tout système.

- **Robustes et tolérants** : Un agent mobile peut se déplacer pour éviter une erreur matérielle ou logicielle, ou tout simplement un arrêt de la machine. Contrairement a un programme classique qui est lié à une machine.

3.3.3. Les agents mobiles et le routage dans les réseaux Ad hoc

Les protocoles de routage Table-Driven exigent de connaître la topologie du réseau entier. Pour que les noeuds restent au courant des changements dans la topologie des informations de mis à jour doivent être fréquemment propager dans tout le réseau. Les protocoles Table-Driven utilisent donc en permanence une grande partie de la capacité du réseau pour garder les informations de routage à jour ce qui les rend inadaptés pour les réseaux ad hoc. Dans l'autre coté les protocole de routage On-demand utilisent une procédure de découverte de route qui génère un grand volume de trafic et produit la congestion du réseau et provoque la perte de paquets, ce qui les rend aussi ces protocoles inadaptés pour les réseaux ad hoc.

Les agents mobiles sont une solution pour découvrir la topologie du réseau et mettre en oeuvre un protocole de routage sans générer trop de trafic dans le réseau [1]. les agents mobiles se déplacent dans le réseau et recueillent les informations nécessaires pour le routage, arrivent à un noeud il vont mettre à jour sa table de routage par les dernières informations recueillies. Un noeud dans le réseau reçoit des agents des informations sur la topologie du réseau ce qui lui permet de calculer des routes correctes pour communiquer avec les autres noeuds.

3.4. Travaux relatives

L'idée d'utiliser les agents pour le routage dans les réseau ad hoc a été explorer récemment dans quelques travaux. Nous allons présenter dans la suite de ce chapitre quelques protocoles de routage ad hoc basé sur les agents.

3.4.1 Le protocole MARP

MARP *(Multi-Agent Routing Protocol*) [1] est un protocole de routage pour les réseaux ad hoc basé sur les agents. Les routes dans MARP sont découvertes à l'avance en utilisant des agents mobiles. Les agent mobiles explorent le réseau et recueillent les informations de topologie. Les noeuds dans le réseau reçoivent les informations de topologie des agents et peuvent calculer les routes ver les différentes destinations.

Dans son article l'auteur définit quelques notions qui serons utilisées dans le protocole :

- **L'affinité :** L'affinité *Anm* d'un lien entre un noeud *n* et un noeud *m* dans le réseau est la durée de vie prévue de ce lien. Pour calculer le *Anm* le noeud *n* envoie périodiquement un signale au noeud *m*, le noeud *m* va calculer périodiquement la force de ce signal et peut déterminer la distance qui le sépare du noeud m. Il peut aussi savoir s' il s'éloigne ou se rapproche de lui et faire des prévisions sur la durée de vie du lien.

- **La stabilité :** La stabilité d'une route R entre un noeud source et un noeud destination à un instant donné est définit comme l'affinité la plus basse des liens entre le noeud de cette route.

- **La Recency:** La topologie des réseaux ad hoc change rapidement , les informations concernant la topologie deviennent rapidement incorrectes. Ceci signifie que n'importe quelle information qu'un noeud *A* reçoit concernant un autre noeud *B* est seulement partiellement correcte puisque il y a une différence de topologie entre l'instant ou les informations concernant le noeud *B* on été recueillis , et l'instant ou ces informations ont été livrées au noeud *A*. Si un noeud reçoit des agents des informations différentes concernant un autre noeud dans le réseau, il doit garder celles qui soient plus correctes; qui sont les informations les plus récentes. Pour pouvoir déterminer degré d'exactitude d'une information (La recency), chaque noeud dans le réseau maintient un compteur de récency qui est initialisé à 0. Quand un agent arrive à un noeud il incrémente son compteur et enregistre sa nouvelle valeur avec les informations recueillis à partir du noeud. La valeur de récency d'un noeud représente le nombre de fois que le noeud a été visité par des agents. Si deux agents ont des information différentes au sujet du même noeud alors, l'agent portant la plus grande valeur de receny a des informations plus récentes.

- **Le Temps de Migration des agents :** On ne permet pas à un agent visitant un noeud d'émigrer immédiatement à un autre noeud. Un agent sera forcé de rester dans un noeud pendant une période préspécifiée appelée Time-to-Migrate (TtM), avant de pouvoir migrer. Le temps de migration TtM permet de contrôler la congestion du réseau due au trafic des agents. Par exemple, si TtM = 100 millisecondes, ca veut dire que la médium sans fil transmettra l'agent toutes les 100 millisecondes, Si on suppose qu'un agent prendrait

approximativement 3 millisecondes. pour migrer physiquement d'un noeud à l'autre, alors le médium sans fil serait libre du trafic d'agent 97 pour cent du temps. La réduction du trafic d'agent (en augmentant TtM) réduit la fréquence à laquelle les agents peuvent visiter les noeuds du réseau. Ceci peut s'avérer peu convenable si le réseau est fortement mobile et la topologie change rapidement car les informations de topologie deviennent rapidement incorrectes. Le compromis est ainsi entre la congestion et la précision des informations.

Un agent comprend les trois composants suivants :

- L'identifiant de l'agent.
  Le programme de l'agent.
  La mémoire de l'agent.

La mémoire de l'agent contient un ensemble de variables d'état du réseau telle que l'affinité et la recency, elle contient aussi des informations concernant l'état des liens entre les noeuds.

La taille de la mémoire est un paramètre important qui affecte le performances du protocole, une taille de mémoire plus grande permet aux agents de fournir plus d'informations aux noeuds, mais en même temps elle augmente les frais de migration des agents. Le nombre d'agents utilisé est aussi un paramètre important , plus il y a d'agents pour le routage plus les frais augmentent. La taille de la mémoire et le nombre d'agents doivent être choisis afin d'avoir un équilibre entre les frais et l'efficacité du routage.

Quand un agent arrive à un noeud N il exécute les étapes suivantes :

1. Il enregistre dans le noeud les dernières informations recueillis qui se trouvent dans sa mémoire.
2. l'agent détermine tous les noeuds qui sont des voisins de N.
3. Il détermine en suite le noeud voisin qui a la plus petite valeur de récency, c'est le noeud qui a été le moins rendu visite comme perçu par le noeud N. La récency d'un noeud est le nombre de visites d'agents que le noeud a reçu.
4. Si ce voisin de N n'a pas été visité dans les 3 visites précédentes par d'autres agents à partir du noeud N, l'agent choisit ce voisin comme prochaine destination, Si non l'agent choisit le deuxième voisin le moins visité. Ceci assurera que les agents arrivant à un même noeud ne choisissent pas la même destination consécutivement.
5. Après le choix de la bonne destination l'agent Incréments la valeur du recency du noeud N et l'enregistre dans sa mémoire, il enregistre aussi les informations d'états des liens du

noeud. L'agent migre ensuite vers sa prochaine destination.

L'information qu'un noeud a obtenu d'un agent devient de plus en plus imprécise avec le temps, ceci est due à la mobilité des noeuds, cependant le noeud obtient à chaque fois des agents une mise à jour de ces informations. Chaque noeud a donc une vue plus ou moins précise sur l'affinité des liens dans le réseau.

MARP réduit les valeurs d'affinité de chaque lien avec le temps. ceci suppose que l'affinité des liens est toujours en diminution. Après certain temps, si un noeud constate qu'une route utilisée a atteint une basse stabilité (indiquant qu'une erreur d'itinéraire est imminente), le noeud calcul une nouvelle route plus stable en utilisant les informations de topologie reçu des agents et l'utilise à la place de la première.

En réduisant l'affinité un noeud fait en quelque sorte des prévisions pessimistes sur l'affinité des liens, ce qui permet d'éviter les coupures de routes et d'avoir des communications continuent entres les noeuds. Il est probable qu'un lien du réseau existe en réalité, mais que le noeud l'a supprimé, mais il est peu probable, qu'un noeud considère qu'un lien existe alors qu'il n'existe pas en réalité.

Les agents mobiles dans MARP sont utilisés pour recueillir et diffuser les informations de routage dans le réseau, un agent mobile se déplace seulement à un noeud à la fois au lieu d'inonder le réseau avec les paquets de mise à jour. Le nombre d'agents peut aussi être utiliser pour contrôler le trafic dans le réseau.

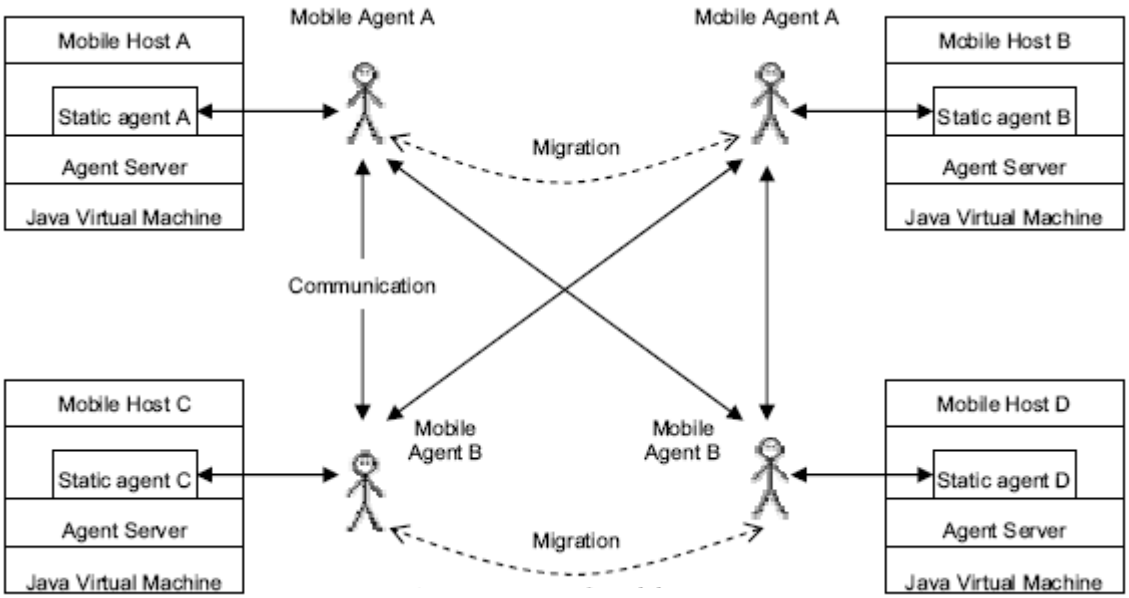
L'établissement des routes dans MARP est dépendant des agents qui visitent les noeuds et fournissent les informations, si un noeud veut envoyer des paquets de données à une destination pour la quelle il n'a pas de routes assez récentes, il doit enregistrer les paquets dans son buffer et attendre jusqu'à ce qu'un agent lui fournissent les routes nécessaires, ce qui augmente le délai initial, en plus dans certains cas un noeud transportant des agents se déconnecte du réseau ce qui diminue le nombre d'agents participant au routage ce qui a son tour diminue les performances.

3.4.2 Un modèle générique a base d'agents

Dans [2] un modèle générique a base d'agent pour effectuer le routage des données dans les réseaux ad hoc a été proposé. Dans ce modèle on utilise des agents mobiles et des agents statiques pour déterminer les routes entres les différents noeuds. Chaque noeud dans le réseau

contient un agent statique qui est chargé de la gestion des ressources locales tel que l'énergie et capacité mémoire. Les agents mobiles explorent le réseau et fournissent les informations de routage au agents statiques, les agents statiques vont utiliser ces informations pour calculer la meilleure route pour transmettre les paquets.

Figure 3.1 : Le modèle d'agents pour le routage.



Chaque noeud dans le modèle exécute un agent serveur qui fournit les fonctionnalités nécessaires aux agents statiques et aux agents mobiles tel que la migration et la communication.

Chaque noeud contient un agent statique. La tache de l'agent statique est de maintenir la table de routage, de décider quelles routes utilisées pour le routage et de gérer les ressources locales. De l'autre coté les agents mobiles sont responsables de la collecte des informations a partir des agents statiques, de mettre à jour les tables de routage et de découvrir de nouvelles routes. Ils informent aussi les agents statiques et les autres agents mobiles des changements dans le réseau.

3.4.3 Protocole MWAC

Dans [9] on propose un protocole de routage à base d'agent pour les réseaux mobiles ad hoc, appelé MWAC (*Multi-Wireless-Agent Communication*) dans ce protocole on associe à chaque noeud un agent. Les agents vont s'organiser d'une manière dynamique en groupes ayant une structure hiérarchique afin de localiser au mieux l'inondation et ainsi

obtenir un gain en ressources. L'organisation des agents va émergé de leurs interactions et de l'évolution de leurs états.

Un groupe dans ce protocole contient :

- Un ***agent représentant*** qui va gérer les communications au sein de son groupe,
- Un ou plusieurs ***agents de liaison*** : ils appartiennent à plusieurs groupes et permettent aux différents représentants de communiquer entre eux,
- Aucun ou plusieurs ***simples membres*** qui n'ont aucun rôle particulier dans le groupe si ce n'est recevoir et traiter les paquets qui leurs sont destinés et transmettre leurs propres paquets.

La figure suivante est un exemple de l'organisation des agents en groupes, un agent est soit un représentant (R), soit un agent liaison (L), soit un simple membre (S).

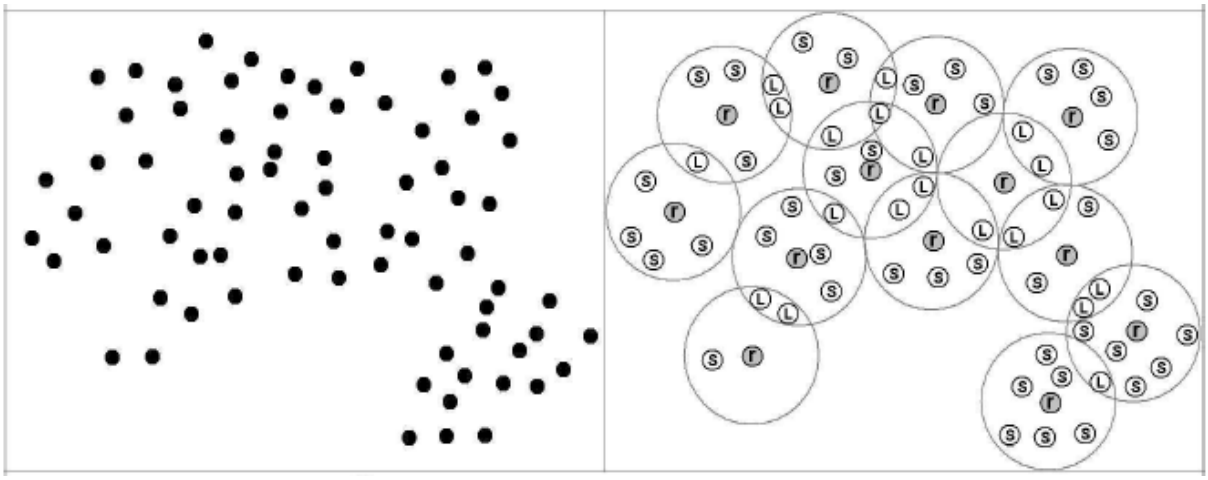


Figure 3.2. : Organisation des agents dans MWAC

Un agent représentant devra traiter toutes les requêtes d'envois de paquets des membres de son groupe, quand le destinataire d'un paquet n'est pas voisin de l'émetteur, c'est à lui que revient le rôle de trouver une route et de vérifier le bon acheminement du paquet. C'est aussi lui qui est inondé par les requêtes de recherche de route des autres représentants. Il est donc plus sollicité que les autres agents de son groupe. Un agent de liaison est chargé d'assurer la communication entre les groupes c'est-à-dire ses voisins qui sont représentants. Ce sont eux qui distribuent les requêtes de recherche de route aux représentants. Les agents simples

membres n'ont aucun rôle particulier dans la gestion des paquets.

Les données que doivent contenir les agents sont variables. Elles dépendent du statut de l'agent dans le groupe :

- **Données sur soi :** Tout agent connaît son identifiant et son statut dans le groupe représentant (R), agent liaison (L) ou simple membre (S).
- **Données sur les autres :** Les données qu'un agent possède sur les autres dépendent du rôle de l'agent. Chaque agent possède une table de voisins. Cette table contient pour chaque voisin un triplet <identifiant, rôle, groupe> auquel il appartient. Si l'agent est un représentant il aura en plus une liste des groupes adjacents qui lui permet de connaître l'identifiant des groupes et donc de leur représentant voisin. Si l'agent est un simple membre (agent SM) alors il ne contient que la table des voisins définie précédemment. Si l'agent est un agent de liaison il ne possède, tout comme l'agent simple membre, qu'une seule table, celle des voisins. Il sait qui sont les groupes voisins grâce à l'identifiant des représentants qu'il perçoit.

Au début un agent va envoyer le message *QuiSontMesVoisins* pour connaître ses voisins et leur demander de se présenter. ses voisins répondent avec le message *JeSuisUnDeTesVoisins* précisant le triplet <id,rôle,groupe> qui leur est associé. L'agent détermine, en fonction de son voisinage, le rôle qu'il doit prendre. Il le signalera à ses voisins via le message *JeChangeDeRôle*. Le protocole d'interaction entre un agent et ses voisin pour déterminé son rôle est illustré dans la figure suivante.

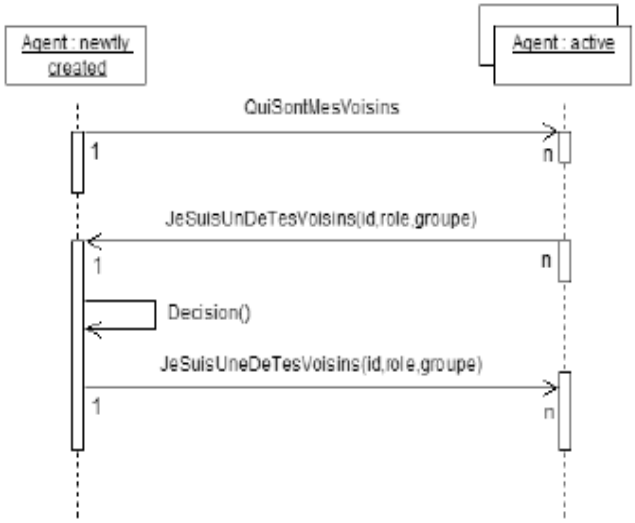


Figure 3.4 : Interaction entre un agent et ses voisin pour déterminer son rôle.

Chaque agent calcul une fonction score qui va permettre d'élire les représentants déterminer le rôle de chaque agent. La fonction sore est calculée en multipliant le nombre de voisins par le niveau d'énergie de l'agent. L'algorithme suivant est ensuite utilisé :

SI monNombreDeVoisin= 0 **ALORS**
// On a des voisins
SI nbRepresentantsVoisin = 0 **ALORS**
// Aucun de nos voisins n'est représentant : on décide de le devenir. Ce cas intervient quand on vient
// juste de créer l'agent ou quand il vient de se déplacer.
monRôle = _;
SINON SI nbRepresentantsVoisin = 1 **ET** monRôle=REPRESNTANT **ALORS**
// Un de nos voisins est représentant : on se soumet à son autorité et ce même s'il est moins efficace
// que nous : on privilégie (pour le moment) la stabilité dans l'organisation à sa performance.
MonRôle = _;
SINON
//Il existe, dans notre voisinage, plusieurs représentants.
SI monRôle = R **ALORS**
// Je suis moi-même représentant : j'entre en conflit avec les autres prétendants à ce rôle. Une
// élection va avoir lieu et l'agent au meilleur score restera en place
ProcedureElectionRepresentant()
SIN
// On n'est pas représentant : on devient agent de liaison pour ces représentants

monRôle =
FINSI
FINSI
SINON
// On n'a pas de voisin : on n'a plus aucun rôle
monRôle =
FINSI

Dans l'algorithme précédant une procédure d'élection du représentant est lancée quand au moins deux agents voisins sont représentants. Le protocole d'élection du représentant est

illustré dans la figure suivante :

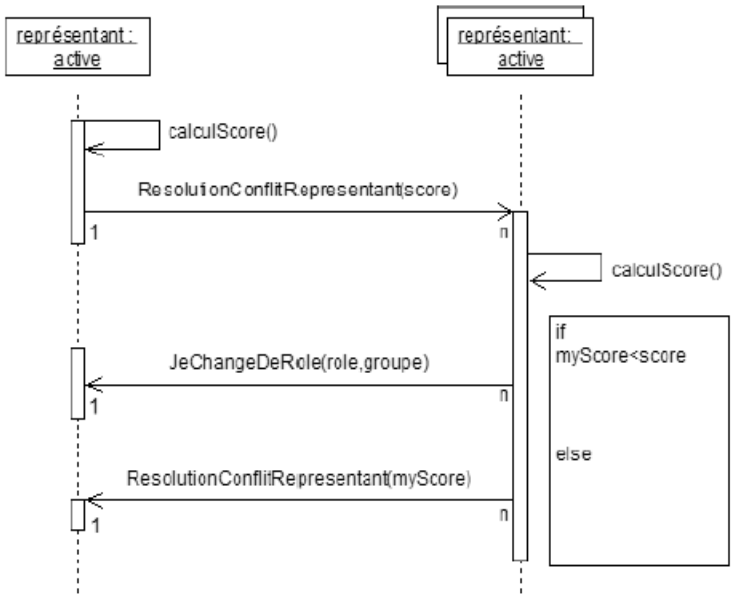


Figure 3.5 : Protocole d'élection du meilleur représentant.

Le message *ResolutionConflitRepresntant* est un message utilisé par un représentant en conflit avec un ou plusieurs autres représentants pour communiquer son score, Le message

JeChangeDeRole est utilisé par l'agent qui a changé de rôle pour en avertir ses voisins. Ce message contient comme information l'identifiant de l'agent, son rôle et son groupe.

Dans ce protocole on diffuse les requêtes de recherche de route aux agents représentants, ce qui permet de mieux localisé l'inondation et ainsi de réduire le trafic engendré par l'inondation. Mai la formation et la maintenance de la structure des groupes engendrent un trafic supplémentaire dans le réseau. Aussi les agents représentants sont responsables de la transmission des paquets de données il sont donc plus sollicité que les autres agents.

3.5. Conclusion

Dans ce chapitre on a présenté la technologie des agents et des systèmes multiagents. Nous avons vu que l'agent intègre des aspects qui sont nécessaires pour gérer les réseaux ad hoc comme la coopération l'autonomie et l'intelligence. Nous avons aussi présenté la technologie des agents mobiles, nous avons vu que les agents mobiles sont une alternative pour découvrir la topologie du réseau et éviter d'utiliser l'inondation. Nous avons aussi présenté quelques protocoles de routage a base d'agents. Dans le chapitre suivant nous allons proposé un protocole de routage pour les réseaux mobiles ad hoc basé sur les agents.

Chapitre 3
Le routage à base d'agents dans les réseaux Ad Hoc

3.1. Introduction

L'agent et les systèmes multiagent sont une nouvelle technologie issue du domaine de l'intelligence artificielle, cette technologie est appliquée dans beaucoup de domaine tel que les systèmes distribués, les interfaces homme machine et les réseaux. Cette technologie est particulièrement intéréssante à utiliser dans des environnements de nature distribué et dynamique comme les réseaux ad hoc. Dans ce chapitre nous allons présenter la technologie

des agents et des systèmes multiagent ainsi que quelques protocoles de routage pour les réseaux ad hoc basés sur les agents.

3.2. L'agent et les systèmes multiagents

Pour Jennings et Wooldridge [11],[18] un agent est une entité informatique *située* dans un environnement, qui est capable *d'agir* de manière *autonome* et *flexible* de façon à remplir ses objectifs, c'est à dire ceux qui lui ont été assignés lors de sa conception.

- L'aspect **situé** signifie que l'agent perçoit son environnement et qu'il est capable de le modifier.

- L**'autonomie**, signifie que l'agent est capable d'agir sans l'intervention directe d'êtres humains ou d'autres agents et qu'il est capable de contrôler ses propres actions et son état interne.

- La **flexibilité** caractérise un *agent intelligent*. Être *flexible* signifie que l'agent est capable : de percevoir son environnement et répondre en temps voulu aux changements qui s'y produisent (*réactivité*); d'avoir un comportement dirigé par son but, d'être opportuniste et capable d'initiative quand c'est approprié (*proactivité*); et d'interagir, quand c'est approprié, avec d'autres agents artificiels ou humains de manière à compléter leur résolution de problème et d'aider les autres par leurs activités (*sociabilité*).

Un Système multiagent (SMA) est une application où les agents travaillent ensemble pour résoudre un problème commun, les possibilités de chaque agent prises séparément ne permet pas de le résoudre. Chaque agent possède donc des connaissances et savoir-faire limités, ce qui l'oblige à interagir avec d'autres pour mener à bien le projet commun.

Les SMA mettent en oeuvre des interactions complexes, comme la *coopération*, la *coordination* et la *négociation* [12]:

- Il y a **coopération** lorsque les agents travaillent ensemble vers un but commun.

- La **coordination** consiste à organiser les activités liées à la résolution d'un problème de manière à éviter les interactions néfastes et à exploiter les interactions bénéfiques.
- La **négociation** a pour objectif la gestion des conflits entre agents, c'est à dire qu'elle

consiste à aboutir à un accord acceptable par l'ensemble des agents impliqués.

3.2.1 Les SMA et le routage dans les réseaux ad hoc

Le système de gestion de routage dans les réseaux ad hoc nécessite une distribution des opérations, car il n'existe aucun élément central fixe qui peut organisé le réseau, il nécessite la coopération entre les différents noeuds mobiles car un noeud mobile ne peut communiquer qu'avec les noeuds proches de lui , il a donc besoin de coopérer avec les autres noeuds mobiles qui vont l'aider à transmettre les informations. Le système de gestion de routage dans les réseaux ad hoc doit aussi être robuste, il doit pouvoir continuer a fonctinner normalement en cas de défaillance d'un ou plusieurs noeuds, il doit être intelligent, il doit choisir les meilleures routes pour transmettre les paquets et réagir rapidement au évènements qui peuvent intervenir. Les systèmes multiagent prennent en compte les aspects de coopération, d'autonomie, de distribution et d'intelligence ils nous semblent donc appropriés pour gérer le routage dans les réseaux ad hoc.

3.3. L'agent mobile

Un agent mobile est un agent possédant la capacité de se déplacer entre les machines du réseau. Un agent mobile qui s'exécute sur une machine du réseau peut suspendre sont exécution sur cette machine , se transférer à une autre machine du réseau et continuer son exécution. A un instant données un agent mobile est constitué de son code, de son contexte d'exécution et de ses données.

Les agents mobiles héritent de deux technologies, la technologie des agents, et la technologie du code mobile qui consiste à définir un code tel que le programme compilé dans ce code puissent être porter et exécuter sur n'importe quelle machine.

3.3.1. L'environnement d'exécution d'agents mobiles

Un environnement d'exécution d'agents mobiles fournit des primitives pour créer, lancer et exécuter un agent mobile [25]. Cet environnement est constitué d'un ensemble de programmes statiques qui s'exécutent sur les sites du système susceptibles d'accueillir des agents.

Les environnements d'exécution d'agents mobiles offrent plusieurs services de base permettant l'exécution d'un agent mobile sur un site, tel que :

- **La création d'un agent mobile** : Un agent mobile peut être créé localement sur la machine locale ou sur une machine distante. A sa création, un nom globalement unique doit être attribué à l'agent ce qui va permettre de localiser l'agent et de communiquer avec lui.

- **La migration d'un agent mobile :** La migration permet le transfert d'un agent en cours d'exécution d'un site à un autre à travers le réseau. Il existe deux types de migration, *la migration forte* permet à un agent de se déplacer quelque soit l'état d'exécution dans lequel il se trouve, dans ce type de migration l'agent se déplace avec son code, son contexte d'exécution et ses données, dans ce cas, l'agent reprend son exécution après la migration exactement là où elle était avant son déplacement. Et *la migration faible* qui ne fait que transférer avec l'agent son code et ses données, sur le site de destination, l'agent redémarre son exécution depuis le début en appelant la méthode qui représente le point d'entrée de l'exécution de l'agent, et le contexte d'exécution de l'agent est réinitialisée.

- **L'accès aux ressources locales :** L'agent mobile doit pouvoir accéder aux ressources de la machine ou il se trouve. cela pose le problème du piratage des sites visités par les agents mobiles.
- **La communication entre les agents mobiles** : Un agent mobile doit pouvoir communiquer avec d'autres agents résidents dans la même machine ou avec des agents résidents dans d'autres machine distante.

3.3.2. Avantages des agents mobiles

Dans [10] on donne un ensembles de raisons qui justifient le choix d'utiliser les agents mobiles :

- **Réduire la charge du réseau** : les agents mobiles sont envoyés sur la machine où résident les données. Ainsi, c'est le calcul qui va aux données et non les données au calcul. Cela va permettre de réduire significativement, voir supprimer, les communications distantes et se contenter des interactions locales, ce qui réduit la charge du réseau.

- **Surmonter le temps de latence sur le réseau** : Les agents mobiles sont envoyés dans la zone d'activité, afin d'entreprendre des actions sur place. Cela leur permet de répondre immédiatement à des changements dans leur environnement et de surmonter le temps de latence du réseau.

- **S'exécuter de manière asynchrone et autonome** : Les agents mobiles s'exécutent indépendamment de leur plate-forme d'origine, c'est à dire un agent peut continuer à parcourir le réseau même sans interaction avec l'utilisateur qui la envoyer, cela permet a l'utilisateur qui a envoyé l'agent de déconnecter sa machine et préservé les ressources pendant le travail de l'agent. Cette caractéristique rend les agents mobiles très intéressants pour les réseaux ad hoc, ou la connexion entre les machines du réseau est très fragile et les ressource en bande passante et énergie sont limitées.

- **S'adapter dynamiquement aux environnements d'exécution** : Les agents mobiles ont la capacité de percevoir et de s'adapter dynamiquement a leur environnement.

- **Hétérogènes** : la multiplicité des systèmes augmente l'incompatibilité des langages de commandes et des formats de données. Les agents mobiles peuvent résoudre ce problème sous contrainte que les primitives du langage de l'agent soit exécutable sur tout système.

- **Robustes et tolérants** : Un agent mobile peut se déplacer pour éviter une erreur matérielle ou logicielle, ou tout simplement un arrêt de la machine. Contrairement a un programme classique qui est lié à une machine.

3.3.3. Les agents mobiles et le routage dans les réseaux Ad hoc

Les protocoles de routage Table-Driven exigent de connaître la topologie du réseau entier. Pour que les noeuds restent au courant des changements dans la topologie des informations de mis à jour doivent être fréquemment propager dans tout le réseau. Les protocoles Table-Driven utilisent donc en permanence une grande partie de la capacité du réseau pour garder les informations de routage à jour ce qui les rend inadaptés pour les réseaux ad hoc. Dans l'autre coté les protocole de routage On-demand utilisent une procédure de découverte de route qui génère un grand volume de trafic et produit la congestion du réseau et provoque la perte de paquets, ce qui les rend aussi ces protocoles inadaptés pour les réseaux ad hoc.

Les agents mobiles sont une solution pour découvrir la topologie du réseau et mettre en oeuvre un protocole de routage sans générer trop de trafic dans le réseau [1]. les agents mobiles se déplacent dans le réseau et recueillent les informations nécessaires pour le routage,

arrivent à un noeud il vont mettre à jour sa table de routage par les dernières informations recueillies. Un noeud dans le réseau reçoit des agents des informations sur la topologie du réseau ce qui lui permet de calculer des routes correctes pour communiquer avec les autres noeuds.

3.4. Travaux relatives

L'idée d'utiliser les agents pour le routage dans les réseau ad hoc a été explorer récemment dans quelques travaux. Nous allons présenter dans la suite de ce chapitre quelques protocoles de routage ad hoc basé sur les agents.

3.4.1 Le protocole MARP

MARP *(Multi-Agent Routing Protocol*) [1] est un protocole de routage pour les réseaux ad hoc basé sur les agents. Les routes dans MARP sont découvertes à l'avance en utilisant des agents mobiles. Les agent mobiles explorent le réseau et recueillent les informations de topologie. Les noeuds dans le réseau reçoivent les informations de topologie des agents et peuvent calculer les routes ver les différentes destinations.

Dans son article l'auteur définit quelques notions qui serons utilisées dans le protocole :

- **L'affinité :** L'affinité *Anm* d'un lien entre un noeud *n* et un noeud *m* dans le réseau est la durée de vie prévue de ce lien. Pour calculer le *Anm* le noeud *n* envoie périodiquement un signale au noeud *m*, le noeud *m* va calculer périodiquement la force de ce signal et peut déterminer la distance qui le sépare du noeud m. Il peut aussi savoir s' il s'éloigne ou se rapproche de lui et faire des prévisions sur la durée de vie du lien.

- **La stabilité :** La stabilité d'une route R entre un noeud source et un noeud destination à un instant donné est définit comme l'affinité la plus basse des liens entre le noeud de cette route.

- **La Recency:** La topologie des réseaux ad hoc change rapidement , les informations concernant la topologie deviennent rapidement incorrectes. Ceci signifie que n'importe quelle information qu'un noeud *A* reçoit concernant un autre noeud *B* est seulement partiellement correcte puisque il y a une différence de topologie entre l'instant ou les informations concernant le noeud *B* on été recueillis , et l'instant ou ces informations ont été livrées au noeud *A*. Si un noeud reçoit des agents des informations différentes

concernant un autre noeud dans le réseau, il doit garder celles qui soient plus correctes; qui sont les informations les plus récentes. Pour pouvoir déterminer degré d'exactitude d'une information (La recency), chaque noeud dans le réseau maintient un compteur de récency qui est initialisé à 0. Quand un agent arrive à un noeud il incrémente son compteur et enregistre sa nouvelle valeur avec les informations recueillis à partir du noeud. La valeur de récency d'un noeud représente le nombre de fois que le noeud a été visité par des agents. Si deux agents ont des information différentes au sujet du même noeud alors, l'agent portant la plus grande valeur de receny a des informations plus récentes.

- **Le Temps de Migration des agents :** On ne permet pas à un agent visitant un noeud d'émigrer immédiatement à un autre noeud. Un agent sera forcé de rester dans un noeud pendant une période préspécifiée appelée Time-to-Migrate (TtM), avant de pouvoir migrer. Le temps de migration TtM permet de contrôler la congestion du réseau due au trafic des agents. Par exemple, si TtM = 100 millisecondes, ca veut dire que la médium sans fil transmettra l'agent toutes les 100 millisecondes, Si on suppose qu'un agent prendrait approximativement 3 millisecondes. pour migrer physiquement d'un noeud à l'autre, alors le médium sans fil serait libre du trafic d'agent 97 pour cent du temps. La réduction du trafic d'agent (en augmentant TtM) réduit la fréquence à laquelle les agents peuvent visiter les noeuds du réseau. Ceci peut s'avérer peu convenable si le réseau est fortement mobile et la topologie change rapidement car les informations de topologie deviennent rapidement incorrectes. Le compromis est ainsi entre la congestion et la précision des informations.

Un agent comprend les trois composants suivants :

- L'identifiant de l'agent.
  Le programme de l'agent.
  La mémoire de l'agent.

La mémoire de l'agent contient un ensemble de variables d'état du réseau telle que l'affinité et la recency, elle contient aussi des informations concernant l'état des liens entre les noeuds.

La taille de la mémoire est un paramètre important qui affecte le performances du protocole, une taille de mémoire plus grande permet aux agents de fournir plus d'informations aux noeuds, mais en même temps elle augmente les frais de migration des agents. Le nombre d'agents utilisé est aussi un paramètre important , plus il y a d'agents pour le routage plus les frais augmentent. La taille de la mémoire et le nombre d'agents doivent être choisis afin d'avoir un équilibre entre les frais et l'efficacité du routage.

Quand un agent arrive à un noeud N il exécute les étapes suivantes :

3. Il enregistre dans le noeud les dernières informations recueillis qui se trouvent dans sa mémoire.
4. l'agent détermine tous les noeuds qui sont des voisins de N.
5. Il détermine en suite le noeud voisin qui a la plus petite valeur de récency, c'est le noeud qui a été le moins rendu visite comme perçu par le noeud N. La récency d'un noeud est le nombre de visites d'agents que le noeud a reçu.
6. Si ce voisin de N n'a pas été visité dans les 3 visites précédentes par d'autres agents à partir du noeud N, l'agent choisit ce voisin comme prochaine destination, Si non l'agent choisit le deuxième voisin le moins visité. Ceci assurera que les agents arrivant à un même noeud ne choisissent pas la même destination consécutivement.
6. Après le choix de la bonne destination l'agent Incréments la valeur du recency du noeud N et l'enregistre dans sa mémoire, il enregistre aussi les informations d'états des liens du noeud. L'agent migre ensuite vers sa prochaine destination.

L'information qu'un noeud a obtenu d'un agent devient de plus en plus imprécise avec le temps, ceci est due à la mobilité des noeuds, cependant le noeud obtient à chaque fois des agents une mise à jour de ces informations. Chaque noeud a donc une vue plus ou moins précise sur l'affinité des liens dans le réseau.

MARP réduit les valeurs d'affinité de chaque lien avec le temps. ceci suppose que l'affinité des liens est toujours en diminution. Après certain temps, si un noeud constate qu'une route utilisée a atteint une basse stabilité (indiquant qu'une erreur d'itinéraire est imminente), le noeud calcul une nouvelle route plus stable en utilisant les informations de topologie reçu des agents et l'utilise à la place de la première.

En réduisant l'affinité un noeud fait en quelque sorte des prévisions pessimistes sur l'affinité des liens, ce qui permet d'éviter les coupures de routes et d'avoir des communications continuent entres les noeuds. Il est probable qu'un lien du réseau existe en réalité, mais que le noeud l'a supprimé, mais il est peu probable, qu'un noeud considère qu'un lien existe alors qu'il n'existe pas en réalité.

Les agents mobiles dans MARP sont utilisés pour recueillir et diffuser les informations de routage dans le réseau, un agent mobile se déplace seulement à un noeud à la fois au lieu d'inonder le réseau avec les paquets de mise à jour. Le nombre d'agents peut aussi être utiliser

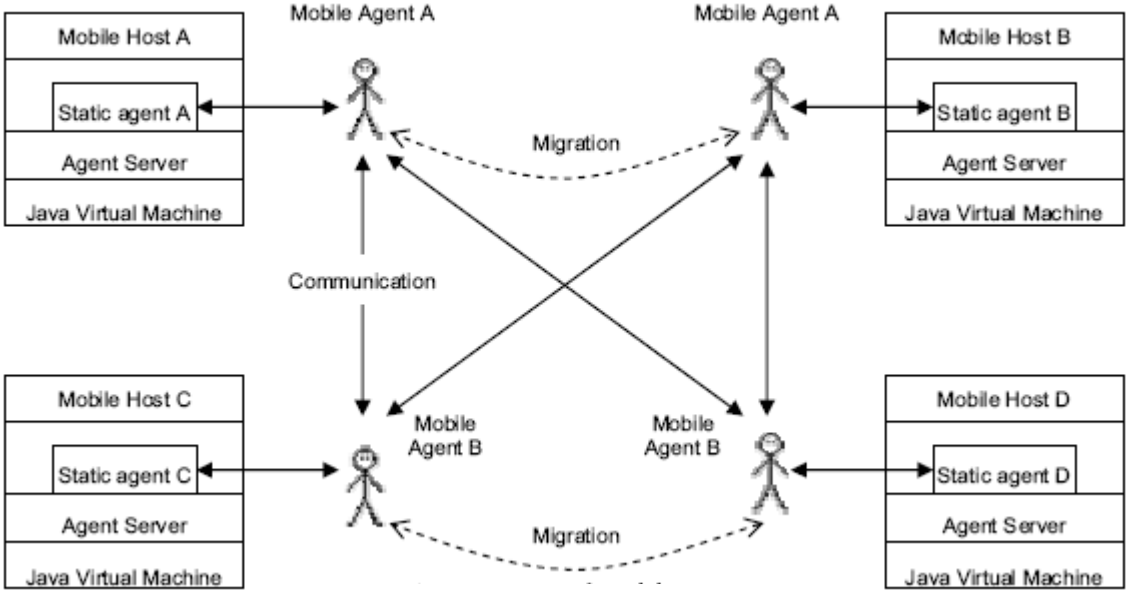
pour contrôler le trafic dans le réseau.

L'établissement des routes dans MARP est dépendant des agents qui visitent les noeuds et fournissent les informations, si un noeud veut envoyer des paquets de données à une destination pour la quelle il n'a pas de routes assez récentes, il doit enregistrer les paquets dans son buffer et attendre jusqu'à ce qu'un agent lui fournissent les routes nécessaires, ce qui augmente le délai initial, en plus dans certains cas un noeud transportant des agents se déconnecte du réseau ce qui diminue le nombre d'agents participant au routage ce qui a son tour diminue les performances.

3.4.2 Un modèle générique a base d'agents

Dans [2] un modèle générique a base d'agent pour effectuer le routage des données dans les réseaux ad hoc a été proposé. Dans ce modèle on utilise des agents mobiles et des agents statiques pour déterminer les routes entres les différents noeuds. Chaque noeud dans le réseau contient un agent statique qui est chargé de la gestion des ressources locales tel que l'énergie et capacité mémoire. Les agents mobiles explorent le réseau et fournissent les informations de routage au agents statiques, les agents statiques vont utiliser ces informations pour calculer la meilleure route pour transmettre les paquets.

Figure 3.1 : Le modèle d'agents pour le routage.



Chaque noeud dans le modèle exécute un agent serveur qui fournit les fonctionnalités nécessaires aux agents statiques et aux agents mobiles tel que la migration et la

communication.

Chaque noeud contient un agent statique. La tache de l'agent statique est de maintenir la table de routage, de décider quelles routes utilisées pour le routage et de gérer les ressources locales. De l'autre coté les agents mobiles sont responsables de la collecte des informations a partir des agents statiques, de mettre à jour les tables de routage et de découvrir de nouvelles routes. Ils informent aussi les agents statiques et les autres agents mobiles des changements dans le réseau.

3.4.3 Protocole MWAC

Dans [9] on propose un protocole de routage à base d'agent pour les réseaux mobiles ad hoc, appelé MWAC (*Multi-Wireless-Agent Communication*) dans ce protocole on associe à chaque noeud un agent. Les agents vont s'organiser d'une manière dynamique en groupes ayant une structure hiérarchique afin de localiser au mieux l'inondation et ainsi obtenir un gain en ressources. L'organisation des agents va émergé de leurs interactions et de l'évolution de leurs états.

Un groupe dans ce protocole contient :

- Un ***agent représentant*** qui va gérer les communications au sein de son groupe,
- Un ou plusieurs ***agents de liaison*** : ils appartiennent à plusieurs groupes et permettent aux différents représentants de communiquer entre eux,
- Aucun ou plusieurs ***simples membres*** qui n'ont aucun rôle particulier dans le groupe si ce n'est recevoir et traiter les paquets qui leurs sont destinés et transmettre leurs propres paquets.

La figure suivante est un exemple de l'organisation des agents en groupes, un agent est soit un représentant (R), soit un agent liaison (L), soit un simple membre (S).

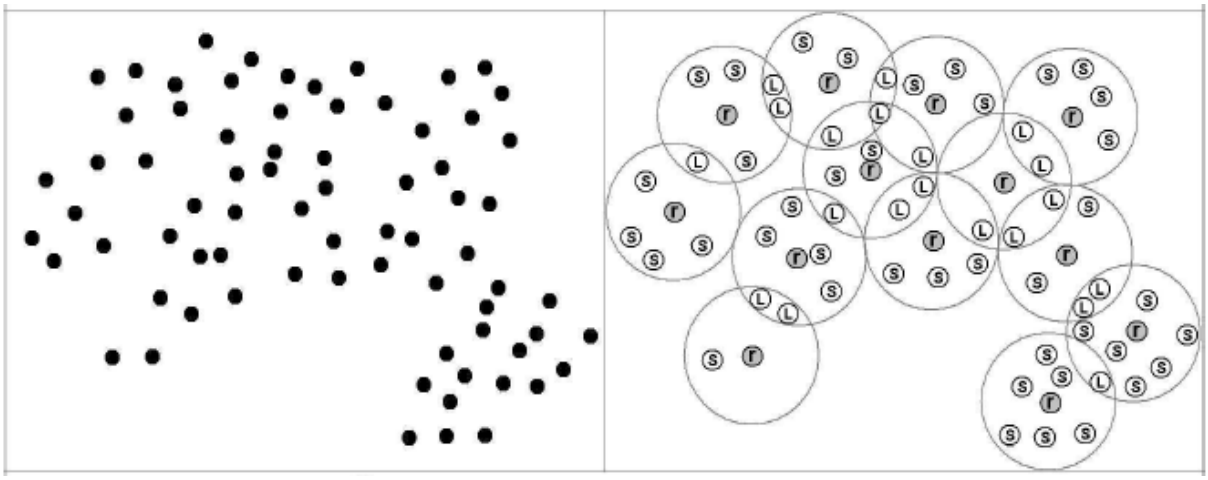


Figure 3.2. : Organisation des agents dans MWAC

Un agent représentant devra traiter toutes les requêtes d'envois de paquets des membres de son groupe, quand le destinataire d'un paquet n'est pas voisin de l'émetteur, c'est à lui que revient le rôle de trouver une route et de vérifier le bon acheminement du paquet. C'est aussi lui qui est inondé par les requêtes de recherche de route des autres représentants. Il est donc plus sollicité que les autres agents de son groupe. Un agent de liaison est chargé d'assurer la communication entre les groupes c'est-à-dire ses voisins qui sont représentants. Ce sont eux qui distribuent les requêtes de recherche de route aux représentants. Les agents simples membres n'ont aucun rôle particulier dans la gestion des paquets.

Les données que doivent contenir les agents sont variables. Elles dépendent du statut de l'agent dans le groupe :

- **Données sur soi :** Tout agent connaît son identifiant et son statut dans le groupe représentant (R), agent liaison (L) ou simple membre (S).
- **Données sur les autres :** Les données qu'un agent possède sur les autres dépendent du rôle de l'agent. Chaque agent possède une table de voisins. Cette table contient pour chaque voisin un triplet <identifiant, rôle, groupe> auquel il appartient. Si l'agent est un représentant il aura en plus une liste des groupes adjacents qui lui permet de connaître l'identifiant des groupes et donc de leur représentant voisin. Si l'agent est un simple membre (agent SM) alors il ne contient que la table des voisins définie précédemment. Si l'agent est un agent de liaison il ne possède, tout comme l'agent simple membre, qu'une seule table, celle des voisins. Il sait qui sont les groupes voisins grâce à l'identifiant des représentants qu'il perçoit.

Au début un agent va envoyer le message *QuiSontMesVoisins* pour connaître ses voisins et leur demander de se présenter. ses voisins répondent avec le message *JeSuisUnDeTesVoisins* précisant le triplet <id,rôle,groupe> qui leur est associé. L'agent détermine, en fonction de son voisinage, le rôle qu'il doit prendre. Il le signalera à ses voisins via le message *JeChangeDeRôle*. Le protocole d'interaction entre un agent et ses voisin pour déterminé son rôle est illustré dans la figure suivante.

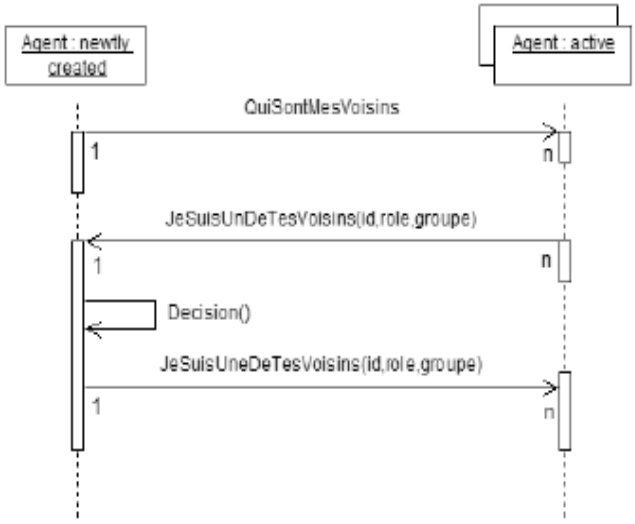


Figure 3.4 : Interaction entre un agent et ses voisin pour déterminer son rôle.

Chaque agent calcul une fonction score qui va permettre d'élire les représentants et de déterminer le rôle de chaque agent. La fonction sore est calculée en multipliant le nombre de voisins par le niveau d'énergie de l'agent. L'algorithme suivant est ensuite utilisé :

SI monNombreDeVoisin**=** 0 **ALORS**
// On a des voisins
SI nbRepresentantsVoisin = 0 **ALORS**
// Aucun de nos voisins n'est représentant : on décide de le devenir. Ce cas intervient quand on vient
// juste de créer l'agent ou quand il vient de se déplacer.
monRôle = _;
SINON SI nbRepresentantsVoisin = 1 **ET** monRôle=REPRESNTANT **ALORS**
// Un de nos voisins est représentant : on se soumet à son autorité et ce même s'il est moins efficace
// que nous : on privilégie (pour le moment) la stabilité dans l'organisation à sa performance.
MonRôle = _;

SINON
//Il existe, dans notre voisinage, plusieurs représentants.
SI monRôle = R **ALORS**
// Je suis moi-même représentant : j'entre en conflit avec les autres prétendants à ce rôle. Une
// élection va avoir lieu et l'agent au meilleur score restera en place
ProcedureElectionRepresentant()
SINON
// On n'est pas représentant : on devient agent de liaison pour ces représentants
monRôle =
FINSI
FINSI
SINON
// On n'a pas de voisin : on n'a plus aucun rôle
monRôle =
FINSI

Dans l'algorithme précédant une procédure d'élection du représentant est lancée quand au moins deux agents voisins sont représentants. Le protocole d'élection du représentant est illustré dans la figure suivante :

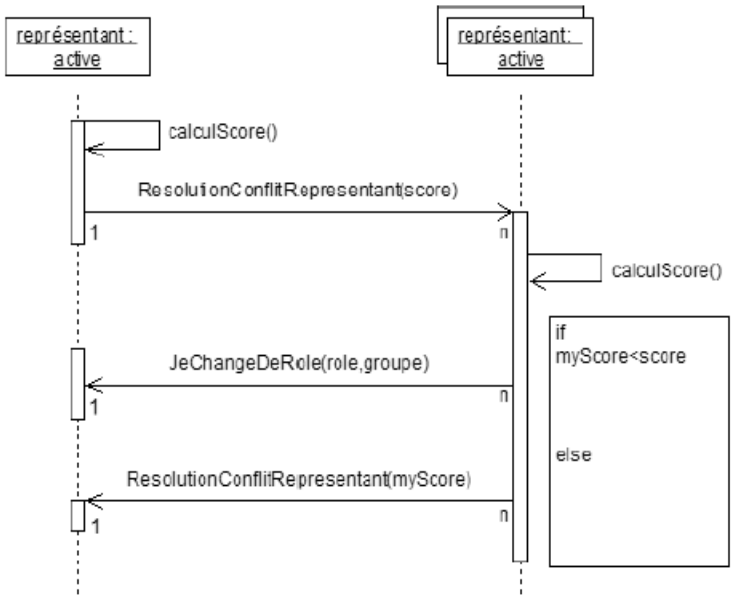


Figure 3.5 : Protocole d'élection du meilleur représentant.

Le message *ResolutionConflitRepresntant* est un message utilisé par un représentant en conflit avec un ou plusieurs autres représentants pour communiquer son score, Le message

JeChangeDeRole est utilisé par l'agent qui a changé de rôle pour en avertir ses voisins. Ce message contient comme information l'identifiant de l'agent, son rôle et son groupe.

Dans ce protocole on diffuse les requêtes de recherche de route aux agents représentants, ce qui permet de mieux localisé l'inondation et ainsi de réduire le trafic engendré par l'inondation. Mai la formation et la maintenance de la structure des groupes engendrent un trafic supplémentaire dans le réseau. Aussi les agents représentants sont responsables de la transmission des paquets de données il sont donc plus sollicité que les autres agents.

3.5. Conclusion

Dans ce chapitre on a présenté la technologie des agents et des systèmes multiagents. Nous avons vu que l'agent intègre des aspects qui sont nécessaires pour gérer les réseaux ad hoc comme la coopération l'autonomie et l'intelligence. Nous avons aussi présenté la technologie des agents mobiles, nous avons vu que les agents mobiles sont une alternative pour découvrir la topologie du réseau et éviter d'utiliser l'inondation. Nous avons aussi présenté quelques protocoles de routage a base d'agents. Dans le chapitre suivant nous allons proposé un protocole de routage pour les réseaux mobiles ad hoc basé sur les agents.

Annexe

1. Exemple de scénario de mobilité

```
#
# nodes: 50, pause: 0.00, max speed: 10.00, max x: 1500.00, max y: 300.00
#
$node_(0) set X_ 1483.525932550755
$node_(0) set Y_ 165.329876003930
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 1422.688740519634
$node_(1) set Y_ 221.985447679873
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 1322.987716803664
$node_(2) set Y_ 217.440243048493
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 177.449665593340
$node_(3) set Y_ 238.191082292177
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 977.609687098957
$node_(4) set Y_ 210.546073572148
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 1068.585465258401
.
.
.
```

```
$node_(44) set Z_ 0.000000000000
$node_(45) set X_ 75.524902602001
$node_(45) set Y_ 283.334307915889
$node_(45) set Z_ 0.000000000000
$node_(46) set X_ 1132.797219702467
$node_(46) set Y_ 4.447258875014
$node_(46) set Z_ 0.000000000000
$node_(47) set X_ 179.669812175287
$node_(47) set Y_ 289.500308250092
$node_(47) set Z_ 0.000000000000
$node_(48) set X_ 407.300907703719
$node_(48) set Y_ 25.942405289044
$node_(48) set Z_ 0.000000000000
$node_(49) set X_ 190.301653016997
$node_(49) set Y_ 218.688192471663
$node_(49) set Z_ 0.000000000000

$ns_ at 0.000000000000 "$node_(0) setdest 1291.632870715027 152.028714591165 9.945346531140"
$ns_ at 0.000000000000 "$node_(1) setdest 327.008964032369 181.488683156121 6.991966381746"
$ns_ at 0.000000000000 "$node_(2) setdest 1117.857717446612 91.989199336095 6.116459097765"
$ns_ at 0.000000000000 "$node_(3) setdest 458.483138322945 282.859947074873 2.576274728837"
$ns_ at 0.000000000000 "$node_(4) setdest 101.221589384156 289.810342290567 9.219421636695"
$ns_ at 0.000000000000 "$node_(5) setdest 1138.638258524416 278.299475155373 3.666068085596"
.
.
.
$ns_ at 0.000000000000 "$node_(44) setdest 357.762988287653 78.987653886301 7.054189671127"
$ns_ at 0.000000000000 "$node_(45) setdest 880.782435695349 194.676955484973 4.238342598913"
$ns_ at 0.000000000000 "$node_(46) setdest 547.632960348310 190.949332205919 4.271979001018"
$ns_ at 0.000000000000 "$node_(47) setdest 1182.485649468614 63.655125568683 8.641747531361"
$ns_ at 0.000000000000 "$node_(48) setdest 707.545574211860 64.683415194543 8.627501096232"
$ns_ at 0.000000000000 "$node_(49) setdest 1391.987174058643 155.328818808402 1.155231264487"
```

2. Exemple de scénario de trafic

```
# créer pour chaque noued : udp,null et cbr

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 64
$cbr_(0) set interval_ 0.05
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)

set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 64
$cbr_(1) set interval_ 0.05
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
```

```
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 64
$cbr_(2) set interval_ 0.05
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)

set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 64
$cbr_(3) set interval_ 0.05
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
.
.
.
set udp_(48) [new Agent/UDP]
$ns_ attach-agent $node_(48) $udp_(48)
set null_(48) [new Agent/Null]
$ns_ attach-agent $node_(48) $null_(48)
set cbr_(48) [new Application/Traffic/CBR]
$cbr_(48) set packetSize_ 64
$cbr_(48) set interval_ 0.05
$cbr_(48) set random_ 1
$cbr_(48) set maxpkts_ 10000
$cbr_(48) attach-agent $udp_(48)

set udp_(49) [new Agent/UDP]
$ns_ attach-agent $node_(49) $udp_(49)
set null_(49) [new Agent/Null]
$ns_ attach-agent $node_(49) $null_(49)
set cbr_(49) [new Application/Traffic/CBR]
$cbr_(49) set packetSize_ 64
$cbr_(49) set interval_ 0.05
$cbr_(49) set random_ 1
$cbr_(49) set maxpkts_ 10000
$cbr_(49) attach-agent $udp_(49)

$ns_ at 31.313289999999999 "$ns_ connect $udp_(0) $null_(14)"
$ns_ at 31.31429 "$cbr_(0) start"
$ns_ at 36.31429 "$cbr_(0) stop"

$ns_ at 92.307659999999998 "$ns_ connect $udp_(1) $null_(48)"
$ns_ at 92.308660000000003 "$cbr_(1) start"
$ns_ at 97.308660000000003 "$cbr_(1) stop"

$ns_ at 48.42109 "$ns_ connect $udp_(2) $null_(38)"
$ns_ at 48.422089999999997 "$cbr_(2) start"
$ns_ at 53.422089999999997 "$cbr_(2) stop"
.
.
.
$ns_ at 15.69769 "$ns_ connect $udp_(47) $null_(17)"
```

```
$ns_ at 15.698689999999999 "$cbr_(47) start"
$ns_ at 20.698689999999999 "$cbr_(47) stop"

$ns_ at 23.077349999999999 "$ns_ connect $udp_(48) $null_(0)"
$ns_ at 23.07835 "$cbr_(48) start"
$ns_ at 28.07835 "$cbr_(48) stop"

$ns_ at 81.239519999999999 "$ns_ connect $udp_(49) $null_(45)"
$ns_ at 81.240520000000004 "$cbr_(49) start"
$ns_ at 86.240520000000004 "$cbr_(49) stop"
```

3. Exemple de fichier trace

```
D 21.274218143 _30_ RTR  CBK 937 cbr 84 [13a 13 1e 800] ------- [3:0 0:1 27 19] [32] 3 6
r 21.277319625 _36_ RTR  --- 939 cbr 84 [13a 24 1e 800] ------- [18:0 40:1 29 36] [83] 2 4
f 21.277319625 _36_ RTR  --- 939 cbr 84 [13a 24 1e 800] ------- [18:0 40:1 28 25] [83] 2 4
r 21.283258802 _42_ AGT  --- 1051 cbr 84 [13a 2a 22 800] ------- [39:0 42:1 27 42] [53] 4 3
r 21.292558136 _34_ RTR  --- 1053 cbr 84 [13a 22 24 800] ------- [39:0 42:1 28 34] [54] 3 3
f 21.292558136 _34_ RTR  --- 1053 cbr 84 [13a 22 24 800] ------- [39:0 42:1 27 42] [54] 3 3
r 21.294768741 _34_ RTR  --- 1058 cbr 84 [13a 22 24 800] ------- [39:0 42:1 28 34] [55] 3 3
f 21.294768741 _34_ RTR  --- 1058 cbr 84 [13a 22 24 800] ------- [39:0 42:1 27 42] [55] 3 3
r 21.296919347 _34_ RTR  --- 1060 cbr 84 [13a 22 24 800] ------- [39:0 42:1 28 34] [56] 3 3
f 21.296919347 _34_ RTR  --- 1060 cbr 84 [13a 22 24 800] ------- [39:0 42:1 27 42] [56] 3 3
s 21.298220616 _4_ AGT  --- 1069 cbr 64 [0 0 0 0] ------- [4:0 35:1 32 0] [32] 0 4
r 21.298220616 _4_ RTR  --- 1069 cbr 64 [0 0 0 0] ------- [4:0 35:1 32 0] [32] 0 4
s 21.298220616 _4_ RTR  --- 1069 cbr 84 [0 0 0 0] ------- [4:0 35:1 30 15] [32] 0 4
r 21.299529519 _42_ AGT  --- 1053 cbr 84 [13a 2a 22 800] ------- [39:0 42:1 27 42] [54] 4 3
r 21.301840414 _42_ AGT  --- 1058 cbr 84 [13a 2a 22 800] ------- [39:0 42:1 27 42] [55] 4 3
r 21.304031334 _15_ RTR  --- 1069 cbr 84 [13a f 4 800] ------- [4:0 35:1 30 15] [32] 1 4
f 21.304031334 _15_ RTR  --- 1069 cbr 84 [13a f 4 800] ------- [4:0 35:1 29 47] [32] 1 4

r 21.306521439 _47_ RTR  --- 1069 cbr 84 [13a 2f f 800] ------- [4:0 35:1 29 47] [32] 2 4
f 21.306521439 _47_ RTR  --- 1069 cbr 84 [13a 2f f 800] ------- [4:0 35:1 28 3] [32] 2 4
s 21.308421396 _39_ AGT  --- 1070 cbr 64 [0 0 0 0] ------- [39:0 42:1 32 0] [59] 0 3
r 21.308421396 _39_ RTR  --- 1070 cbr 64 [0 0 0 0] ------- [39:0 42:1 32 0] [59] 0 3
.

.

.
```

4. Pragramme d'analyse du ratio

```
#!/usr/bin/perl -w
use strict;

 my $paquets_envoyer = 0;
 my $paquets_recu = 0;
 my $paquets_ratio = 0;
while (<STDIN>) {

   if (/^#/) {
      next;
   }

   my @line = split();
```

```
    if (!defined($line[6])) {
        next;
    }

    $line[2] =~ /_(\d+)_/;

    my $lognode = $1;
     if ($line[0] eq "s" && $line[6] eq "cbr" && $line[3] eq "AGT") {

        $line[13] =~ /\[(\d+):/;
        my $ipsrc = $1;
        $line[14] =~ /(\d+):\d+/;
        my $ipdst = $1;

        print "Node $lognode: CBR packet generated at time = $line[1] src=$ipsrc dst=$ipdst\n";
        $paquets_envoyer= $paquets_envoyer+1;
    }

    if ($line[0] eq "r" && $line[6] eq "cbr" && $line[3] eq "AGT") {$paquets_recu = $paquets_recu + 1;}
}
```

4. Exemple de fichier de configuration de la simulation

```
set opt(chan)   Channel/WirelessChannel

set opt(prop)   Propagation/TwoRayGround
set opt(netif)  Phy/WirelessPhy
set opt(mac)    Mac/802_11
set opt(ifq)    Queue/DropTail/PriQueue
set opt(ll)           LL
set opt(ant)       Antenna/OmniAntenna
set opt(x)      1500   ;
set opt(y)            300   ;
set opt(ifqlen) 50           ;
set opt(seed)   0.0
set opt(tr)           Agents.tr    ;
set opt(nam)          Agents.nam   ;
set opt(adhocRouting)  AODV
set opt(nn)           50           ;
set opt(cp)           "../mobility/scene/senario"
set opt(sc)           "../mobility/scene/mobility"
set opt(stop)         100.0   ;
LL set mindelay_              50us
LL set delay_                 25us
LL set bandwidth_             0

Agent/Null set sport_          0
Agent/Null set dport_          0
Agent/CBR set sport_           0
Agent/CBR set dport_           0
Agent/TCPSink set sport_       0
Agent/TCPSink set dport_       0
```

```
Agent/TCP set sport_            0
Agent/TCP set dport_            0
Agent/TCP set packetSize_       1460

Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0

set ns_         [new Simulator]

#set wchan      [new $opt(chan)]
#set wprop      [new $opt(prop)]
set wtopo       [new Topography]

set tracefd     [open $opt(tr) w]
set namtrace    [open $opt(nam) w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace 1500 300

$wtopo load_flatgrid $opt(x) $opt(y)

set god_ [create-god $opt(nn)]

$ns_ node-config -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \
                 -channelType $opt(chan) \
    -topoInstance $wtopo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace OFF

for {set i 0} {$i < $opt(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0              ;
#       $node_($i) topography $wtopo
}
```

```
puts "Loading connection pattern..."
source $opt(cp)

puts "Loading scenario file..."
source $opt(sc)

for {set i 0} {$i < $opt(nn)} {incr i} {

   $ns_ initial_node_pos $node_($i) 20
}

proc record {} {
      global sink0 sink1 sink2 f0 f1 f2
      #Get an instance of the simulator
      set ns [Simulator instance]
      #Set the time after which the procedure should be called again
      set time 3
      set now [$ns now]
```

[31]: « Unicast et multicast dans les réseaux ad hoc sans fil » Anis Laouiti thèse de doctorat a l'université de versailles saint-quentin-en-yvelines Juillet 2002.

[32]: « Routage unicast et multicast dans les réseaux mobiles ad hoc » Hassnaa MOUSTAFA Thèse de doctorat a l'Ecole Nationale Supérieure des Télécommunications 2004.