Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider Biskra
Faculté des Sciences et des Sciences de l'ingénieur
Département d'Informatique
Ecole doctorale

N°d'ordre :……
Série :……

Mémoire

Présenté en vue de l'obtention du diplôme de Magister en Informatique

Option**: Génie Logiciel**

Approche agent mobile pour l'adaptation des réseaux mobiles ad hoc

Par :

Mr BOUZAHER Abdelaziz

Soutenu le : / /

Devant le jury :

- …….	Président
Mr. Kazar Okba, M.C. Université de Biskra	Rapporteur
-……..	Examinateur
-……..	Examinateur

Résumé :

Dans les réseaux ad hoc, un protocole de routage est fortement lié à la dynamique de l'environnement due à la mobilité des nœuds. Le comportement d'un certain protocole nécessite alors une adaptation pour répondre aux changements des caractéristiques de l'environnement principalement l'énergie, le débit des liens, la qualité de service et la topologie.

L'effet de la mobilité des nœuds sur les liens et la topologie du réseau est très lisible. Les protocoles de routage employé dans ce type des réseaux doivent prendre en considération ce facteur de mobilité pour garantir une meilleure mise en œuvre et maintenance de la topologie afin d'assurer la continuité de communication et la reprise lorsqu'une déconnexion.

D'une autre part, les agents mobiles représentent un outil adéquat pour l'adaptation des systèmes aux environnements décentralisés et évolutifs. Ils dotent des caractéristiques qui augmentent la tolérance aux pannes comme la capacité d'autonomie (indépendance lors de l'exécution), de proactivité (capacité de prendre des décisions de manière autonome) et de communication.

L'objectif de ce travail consiste à proposer une approche qui utilise le paradigme « agent mobile » pour traiter le problème d'adaptation des réseaux mobiles ad hoc et de traiter la mobilité fréquente.

Mots clés : réseaux ad hoc, routage, agent mobile, mobilité, métriques, adaptation du réseau.

ملخص

إن برتوكول التوجيه في الشبكات اللاسلكية ad hoc متعلق بشدة بديناميكية البيئة المحيطة وذلك بسبب الحركة الدائمة لعناصر الشبكة, وبالتالي فإن سلوك بروتوكول التوجيه يتطلب تعديلا معيننا للاستجابة للتغيرات في خصائص هذه البيئية المتمثلة أساسا في الطاقة، التدفق في خطوط الاتصال، نوعية الخدمة وبنية الشبكة.
إن أثر حركة عناصر الشبكة على خطوط الاتصال وبنية الشبكة واضح جدا. فبروتوكولات التوجيه المستعملة في هذا النوع من الشبكات لا بد أن تأخذ بعين الاعتبار هذا العامل وذلك لضمان التنفيذ والصيانة الأمثل لبنية الشبكة وبالتالي ضمان الاتصال المستمر أو استئنافه في حالة الانقطاع.
ومن ناحية أخرى فان العون المتنقل يمثل أداة ملائمة لتكييف النظم التي تعمل في بيئة لامركزية وقابلة للتطور، فإن له من الخصائص ما يمكنه من رفع القدرة على مواجهة العطب مثل الذاتية (الاستقلال في الأداء)، روح المبادرة (القدرة على اتخاذ قرارات مستقلة) والاتصالات.
الهدف من هذا العمل هو اقتراح هندسة مبنية على أساس العون المتنقل من اجل التعامل مع مشكلة تكيف شبكات ad hoc مع الحركة المتكررة لعناصر الشبكة.

الكلمات الأساسية: شبكة ad hoc، برتوكول التوجيه، العون المتنقل، الحركية، وحدات القياس، تكييف الشبكة.

Table des matières

Liste des figures

Liste des tables

Introduction générale

A. Contexte

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calcul portables poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but des réseaux : « *L'accès à l'information n'importe où et n'importe quand* ». Le concept des réseaux mobiles ad hoc essaie d'étendre les notions de la mobilité à tous les composants de l'environnement. Le nouvel environnement résultant appelé l'environnement mobile, permet aux unités de calcul, une libre mobilité et il ne pose aucune restriction sur la localisation des usagers.

Les environnements mobiles offrent une grande flexibilité d'emploi. En particulier, ils permettent la mise en réseau des sites dont le câblage serait trop onéreux à réaliser dans leur totalité, voire même impossible (par exemple en présence d'une composante mobile).

Le développement de la technologie sans fil se fait dans deux directions : les réseaux cellulaires avec infrastructure de communication et les réseaux sans infrastructure de communication.

Les réseaux mobiles sans infrastructure également appelés réseaux ad hoc ou IBSS (Independent Basic Service Set) ne comportent pas des entités fixes, tous les noeuds du réseau sont mobiles. Les noeuds communiquent, selon la distance qui les sépare, par deux modes de communication : soit les noeuds mobiles peuvent directement communiquer (en transmission ad

hoc) car ils sont dans la portée de transmission, soit ils doivent utiliser d'autres noeuds mobiles comme des relais pour acheminer les paquets à destination (la transmission est multisauts).

Cette mobilité engendre des nouvelles problématiques liées aux caractéristiques propres à l'environnement mobile tel que : une fréquente déconnexion, un débit de communication et des ressources modestes, et des sources d'énergie limitées.

Les protocoles de routages utilisés dans ce genre de réseaux doivent fournir des mécanismes d'adaptation au changement de l'environnement pour résoudre ou minimiser ces problèmes.

B. Motivation

Les systèmes multi-agents (SMA) avec des agents autonomes fournissent une nouvelle méthode pour analyser, faire de la conception et implémenter des applications complexes et distribuées, car ils font partie du domaine IAD (Intelligence Artificielle Distribuée) en bénéficiant aussi d'autres disciplines comme les sciences cognitives, sociologie, et psychologie sociale.

Aujourd'hui, la plupart des applications nécessitent de distribuer des tâches entre des entités autonomes (ou semi-autonome) afin d'atteindre leurs objectifs d'une manière optimale.

Les agents seront principalement utilisés dans les réseaux sans fil pour améliorer les méthodes de localisation, pour contrôler la signalisation sur le réseau, la sécurité, réduire les accès et adapter le réseau aux besoins de l'utilisateur.

Il est à noter que l'utilisation des systèmes multi-agents pour la conception des applications pour les réseaux mobiles est grandement affectée par les contraintes des ces environnements tels que la bande passante limitée, la grande latence et la limite d'énergie.

D'autres domaines d'application des agents dans la télécommunication sont, également, importants comme la proposition ou la composition dynamique des services personnalisés aux usagers.

Un agent mobile est une sorte d'agent qui se déplace d'un site à un autre en cours d'exécution pour accéder à des données ou à des ressources distantes. Il se déplace avec son code son état d'exécution et ses propres données. La décision de migration peut se faire à l'initiative de l'agent lui-même de manière autonome.

D’une autre part, le modèle d’agent mobile n’exige pas des connexions permanentes ce qui très recommandé pour un réseau ad hoc caractérisé par une déconnexion forte.

C. Contributions

Notre travail s’intéresse aux protocoles de routage, nous étudions l'utilisation de la technologie d'agents mobiles pour l’adaptation des réseaux ad hoc à la mobilité.

Généralement, la mobilité est mesurée par chaque protocole de routage à l’aide des métriques. Pour réaliser notre objectif, nous avons proposé un ensemble des métriques liées à la population d’agents, la motivation de ce choix est que la mobilité d’un nœud est réellement calculée par rapport à son environnement local (lui-même), le voisinage et le niveau global réseau. C’est pourquoi nous avons proposé un ensemble des agents qui vont crées des clusters (grappes) afin de contrôler le mouvement des nœuds.

D. Organisation de document

Les environnements mobiles sont caractérisés par de fréquentes déconnexions et des restrictions sur les ressources utilisées, surtout si tous les usagers du système sont mobiles ce qui est le cas pour les réseaux ad hoc. Ces limitations transforment certains problèmes, ayant des solutions évidentes dans l'environnement classique, en des problèmes complexes et difficiles à résoudre. Parmi ces problèmes, figure le problème de routage et de mobilité.

Le reste de ce document est divisé en 5 chapitres. Dans le premier chapitre, nous avons introduit les réseaux ad hoc, le routage et les différentes classifications. On parle aussi de l’adaptation de réseaux face au changement de l’environnement et on discute quelque modèle pour la simulation des mouvements des nœuds dans les réseaux ad hoc. Dans le deuxième chapitre, nous avons donné une vue générale sur les agents mobiles. Pour le troisième chapitre, nous discutons quelques approches qui utilisent les agents mobiles pour résoudre le problème de routage et la mobilité fréquente des nœuds dans réseau ad hoc. Dans le quatrième chapitre, nous

allons décrire notre contribution qui est basée sur les agents mobiles, cette approche est implémentée dans le cinquième chapitre et on termine par une conclusion générale.

Chapitre I

Les réseaux ad hoc

1.1 Introduction

Un réseau ad hoc (MANET pour mobile ad hoc network) représente un système distribué complexe constitue d'un grand nombre d'entités mobiles appelant nœuds. La communication entre ces noeuds se fait à travers des interfaces sans fil qui utilisent les ondes radio comme un support de transmission de l'information.

Les réseaux sans fil traditionnels reposent sur une infrastructure partielle représentée par des stations de base fixe reliées par des liaisons filaires. Par contre, un réseau ad hoc ne se base sur aucune infrastructure définie au préalable, ceci revient aux exigences de la nature des applications supporte le modèle ad hoc (fort dynamisme, surpassez le problème de structure, coût et délai d'installation, …).

Généralement, les réseaux ad hoc sont conçus pour les environnements dont les échanges produisent fréquemment à cause de la mobilité des nœuds (exp : un nouveau venu qui va s'intègre dans l'ensemble des nœuds). Cette mobilité provoque des connexions et/ou des déconnexions apparues pendant l'exécution en temps réel des applications, cette situation nécessite des efforts de maintenance afin de répondre aux besoins des utilisateurs en termes de communication et disponibilité des services. Comme il est impossible de réaliser la phase de maintenance manuellement par un superviseur, le système doit être doté de la capacité de s'adapter au changement de l'environnement.

Dans ce chapitre, nous visons à donner un aperçu sur les réseaux ad hoc et quelques concepts liés à cette notion tels que le routage, la mobilité et leur effet, adaptation…etc.

1.2 Historique

À l'origine les réseaux ad hoc sont utilisés pour les applications militaires (réseau tactique) pour améliorer et garantir la communication dans les champs de bataille, l'absence d'une infrastructure est recommandée dans ce genre d'environnement.

Au début des années 70, la première utilisation d'un réseau avec un support radio au sein de projet Packet Radio Network[1] (PRNet) en 1973 de DARPA (The Defense Advanced Research Projects Agency), il dispose d'une architecture distribuée qui partage le canal de diffusion (broadcast) en utilisant une combinaison des méthodes Aloha et CSMA pour l'accès au canal avec une technique de routage store-and-forward multi-hop qui élargir la zone de couverture par répétition des paquets.

Par la suite, en 1983, *le Survivable Radio Networks* (SURAN) a été développé aussi par DARPA. L'objectif était d'étendre le réseau afin de dépasser les limitations (en particulier permettre le passage à des réseaux comportant énormément des noeuds, gérant la sécurité, l'énergie,…). En 1987, l'introduction des technologies LPR(Low-cost Packet Radio) et SCN (Survivable Communication Network), il y a plusieurs d'autres projets qui portent sur ce domaine tel que : GloMo (Global Mobile) un système d'information de DRPA fournis des services multimédias sur une connexion sans fil, WINGs (Wireless Internet Gateways) une architecture réseau peer-to-peer , MMWN (Multimedia Mobile Wireless Network de GTE Internetworking) une architecture réseau à base des clusters, TI (Tactical Internet) une implémentation d'un réseau mobile multi saut en 1997, ELB ACTD(Extending the Littoral Battle-space Advanced Concept Technology Demonstration) un autre réseau ad hoc financé par l'armée américaine en 1999.

Les recherches sont apparues dans le monde commercial au années 90 avec l'apparition de protocole 802.11 de l'IEEE (Institute of Electrical and Electronics Engineers).

Le groupe de travail MANET2 (Mobil Ad hoc Network) de l'IETF (Internet Engineering Task Force) est l'un des groupes actifs qui s'intéressent aux réseaux ad hoc.

1.3 Définition

Un réseau mobile ad hoc, appelé généralement MANET (Mobile Ad hoc NETwork), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans

fil, sans l'aide d'une infrastructure préexistante ou administration centralisée. Un réseau ad hoc peut être modélisé par un graphe $G_t = (V_t, E_t)$ où V_t représente l'ensemble des noeuds (c'est-à-dire les unités ou les hôtes mobiles) du réseau et E_t modélise l'ensemble les connections qui existent entre ces noeuds. Si $e = (u, v) \in E_t$, cela veut dire que les noeuds u et v sont en mesure de communiquer directement à l'instant t.

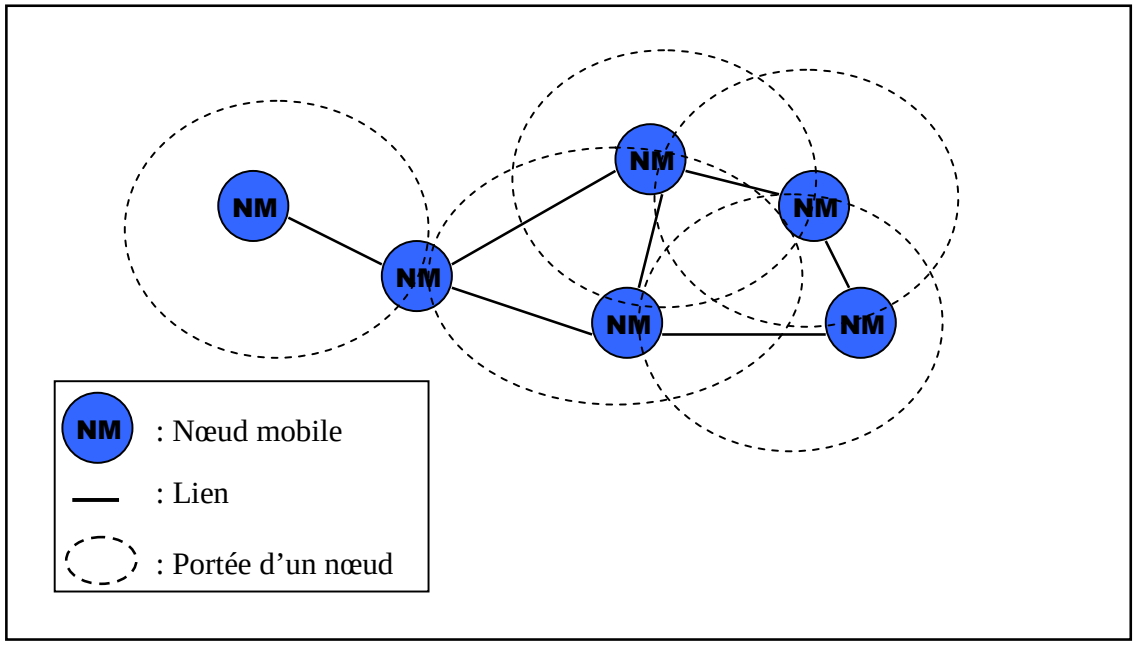

Figure 1.1 : Modélisation d'un réseau ad hoc.

1.4 Les caractéristiques des réseaux ad hoc

Les réseaux mobiles ad hoc sont caractérisés par ce qui suit :

- Une topologie dynamique : Les unités mobiles du réseau se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer, à des instants imprévisibles, d'une manière rapide et aléatoire. Les liens de la topologie peuvent être unis ou bidirectionnels.

- Une bande passante limitée : Une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte soit modeste.

- Des contraintes d'énergie : Les hôtes mobiles sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables. Le paramètre d'énergie doit être pris en considération dans tout contrôle fait par le système.

- Une sécurité physique limitée : Les réseaux mobiles ad hoc sont plus touchés par le paramètre de sécurité, que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.

- L'absence d'infrastructure : Les réseaux ad hoc se distinguent des autres réseaux mobiles par la propriété d'absence d'infrastructures préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau d'une manière continue.

- L'hétérogénéité des noeuds : Un noeud mobile peut être équipé d'une ou plusieurs interfaces radio ayant des capacités de transmission variées et opérantes dans des plages différentes de fréquence. Cette hétérogénéité de capacité peut engendrer des liens asymétriques dans le réseau. De plus, les noeuds peuvent avoir des différences en terme de capacité de traitement (CPU, mémoire) de logiciel et de mobilité (lent, rapide).

- Multihops ou multi saut : Un réseau ad hoc est qualifié par « multihops », car plusieurs noeuds mobiles peuvent participer au routage et servent comme routeurs intermédiaires.

1.5 Les domaines d'applications des réseaux ad hoc

En plus de leurs utilisations dans les applications tactiques militaires, les réseaux ad hoc sont utilisés également dans les domaines civils. On peut citer :

- **Les services d'urgence** : opération de recherche et de secours des personnes, tremblement de terre, feux, dans le but de remplacer l'infrastructure filaire.

- **Le travail collaboratif et les communications dans des entreprises ou bâtiments** : dans le cadre d'une réunion ou d'une conférence par exemple.

- **Applications commerciales** : pour un paiement électronique distant (taxi) ou pour l'accès mobile à l'Internet, ou service de guide en fonction de la position de l'utilisateur.

- **Réseaux de senseurs** : Les capteurs, chargés de mesurer les propriétés physiques des environnements (comme la température, la pression...), sont dispersés (le plus souvent lâchés d'un avion ou d'un hélicoptère) par centaines, voire par milliers sur le site, effectuent leurs mesures et envoient les résultats à une station par l'intermédiaire d'un routage ad hoc à travers le réseau.

- **Le cadre informatique :** Dans le cadre de l'informatique, les réseaux ad hoc peuvent servir à établir des liens entre ses différents composants. Dans ce cas, on parle non plus de LAN (Local Area Network) mais de PAN (Personnal Area Network).

1.6 Les protocoles de routage

Dans tous les domaines, le recours aux standards est primordial. La normalisation des protocoles de routage est l'objectif de groupe de travail MANET de l'IETF.

1.6.1 Définition

Le routage est une méthode d'acheminement des informations vers la bonne destination à travers un réseau de connexion donnée, il consiste à assurer une stratégie qui garantit, à n'importe quel moment, un établissement de routes qui soient correctes et efficaces entre n'importe quelle paire de noeuds appartenant au réseau, ce qui assure l'échange des messages d'une manière continue. Vu les limitations des réseaux ad hoc, la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante.

1.6.2 Classification

La classification des protocoles de routage se fait en fonction de la méthode de création et de maintenance de routes lors de l'acheminement des données. Selon les informations de routage échangées et les méthodes de calcul des routes utilisées, on distingue deux grandes familles de routage, les routages à vecteurs de distance et les routages à état de liens et un autre hybride entre les deux.

1.6.2.1 Routage à vecteurs de distance

Dans un routage à vecteurs de distance, chaque noeud diffuse périodiquement sa table de routage à ses voisins, la table contient les adresses des noeuds destination du réseau et la distance en nombre de sauts pour atteindre chacun d'eux. Un noeud met sa table de routage à jour s'il trouve une route plus courte que celle qu'il a dans sa table, ou si le noeud par lequel il passe pour atteindre une destination donnée change la distance vers cette destination, ou encore s'il trouve un noeud inconnu (c'est-à-dire, qui n'existe pas dans sa table).

1.6.2.2 Routage à état de liens

Dans un routage à état de liens, chaque noeud vérifie l'état des liaisons avec ces voisins (peut aussi calculer le coût de ces liens), et diffuse un paquet contenant ces informations à tout le réseau. Ces diffusions permettent à chaque noeud d'avoir une connaissance complète de la topologie du réseau. Pour calculer les routes, l'algorithme du plus court chemin de Dijkstra est utilisé.

1.6.2.3 Routage hybride

Comme leur nom indique, ce type de routage combine des aspects du routage à état de liens et du routage à vecteur de distance. Cette solution mettant en commun les avantages des deux approches précédentes en utilisant une notion de découpage du réseau.

1.6.3 Protocole de routage dans le réseau ad hoc

Selon la classification précédente, on distingue trois classes de protocoles : les proactifs, les réactifs et les hybrides.

1.6.3.1 Les protocoles de routage proactif

L'essentiel de ce type de routage est l'établissement des routes immédiates entres les sources et les destinations, pour cela, chaque nœud maintient une table de routage contient des informations concernant les autres nœuds de réseaux. Comme l'environnement est mobile, la mise à jour des tables se fait dans un intervalle de temps régulier ou lorsqu'il y a un changement dans l'une des tables de routage. Cette tâche est accomplie par l'échange des messages entre les nœuds. Ces messages permettent de calculer les chemins suivant des critères comme le nombre de sauts séparant les noeuds, le délai et la bande passante disponible sur le chemin.

L'avantage de ces protocoles réside dans l'acheminement des informations qui connaîtront la route à prendre et la transmission sera considérée comme immédiate. L'inconvénient majeur est le coût de maintenance des connaissances sur la topologie et de routage dû à l'envoi périodique des messages. Ceci génère une consommation continue de la bande passante.

De nombreux protocoles proactifs existent. Nous allons présenter les suivants :

1.6.3.1.1 Le protocole OLSR (Optimized Link State Routing)

OLSR [2] est un protocole proactif à état de lien optimisé. Il propose des routes optimales en termes du nombre de sauts dans le réseau en utilisant les relais multipoints. Cette technique consiste à sélectionner un sous-ensemble des voisins appelés les relais multipoints ou MPRs (Multi-Protocol Router) pour chaque nœud. Les nœuds forment le MPR sont ceux qui possèdent une distance de deux sauts. Les relais multipoints sont utilisés dans le but de minimiser le trafic dû à la diffusion des messages de contrôle dans les réseaux. La sélection des relais multipoint s'effectue par la diffusion de deux types de messages :

- Les nœuds échangent des messages HELLO contenant la liste de ses voisins pour déduire la nature des liens symétrique ou asymétrique. Cela permet de choisir un sous-ensemble de voisins dont les liens sont symétriques avec une distance de deux sauts.
- Un message TC (Topology Control) envoyé par les MPR périodiquement dans le réseau, permet la mise à jour des tables de routage.

Une table de routage est construite sur chaque noeud et le routage de donnée s'effectue saut par saut sans aucune intervention de l'OLSR dont son rôle s'arrête à la mise à jour des tables de routage.

1.6.3.1.2 Le protocole TBRPF (Topology Dissemination Based On Reverse Path Forwarding)

C'est un protocole à état de lien. TBRPF [3] échange régulièrement des données sur la topologie du réseau afin de mettre à jour les tables de routage. Deux types de message Hello sont échangés, le premier Hello contient la liste des voisins et le deuxième (Hello différentiel) contient les changements par rapport au dernier Hello envoyé.

Chaque noeud transmet périodiquement des informations sur les sous arbre de la topologie (possible tout l'arbre) qu'il a construite. Il repose sur l'algorithme de Dijkstra pour déterminer le chemin le plus court.

1.6.3.1.3 Le protocole de routage DSDV

DSDV [4] est un protocole à vecteurs de distance basé sur l'algorithme distribué de Bellman-Ford avec quelques améliorations.

Chaque noeud maintient une table de routage qui contient une entrée pour chacun des autres noeuds du réseau, cette entrée contient les informations suivantes :

- identifiant du noeud ;
- le nombre de sauts pour atteindre ce noeud ;
- un numéro de séquence attribué par le noeud destination, ce numéro permet de reconnaître la dernière mise à jour de route et préserve ainsi le réseau du bouclage.

Les mises à jour dans DSDV sont transmises périodiquement à travers deux types de paquets :

- Mise à jour complète : elle correspond à un envoi de toutes les informations de la table de routage et nécessite plusieurs paquets pour l'envoi.

- Mise à jour incrémentale : un paquet contenant les changements depuis la dernière mise à jour complète est envoyé, cette opération ne nécessite qu'un seul paquet. Une table additionnelle est maintenue par chaque noeud pour la sauvegarde des mises à jour incrémentales.

1.6.3.2 Les protocoles de routage réactifs

L'approche réactive se base sur un fonctionnement « à la demande ». Lorsqu'un noeud a besoin d'une route, une procédure de découverte globale est lancée. Dés que la découverte de la route est terminée, une procédure prend en charge la maintenance jusqu'à ce que la destination soit inaccessible à partir du noeud source ou que le noeud source n'aura plus besoin de cette route.

La majorité des approches utilisées lors de la découverte des routes sont basées sur le mécanisme d'apprentissage en arrière (backward learning). Au niveau de chaque nœud et à la réception d'une requête, il sauvegarde la route depuis la source dans une table et renvoyer la requête. Une fois la destination atteinte, elle peut envoyer une réponse en utilisant le chemin inverse, la mis à jour de la route se fait, tant qu'il est en cours d'utilisation. Il existe d'autres techniques tels que « routage source » qui sont utilisées pour tracer les chemins demandés dés le départ par la diffusion des informations.

L'avantage de ce type de protocole est la réduction de la surcharge des réseaux par la mise à jour régulière des tables de routage. Cependant, la recherche des chemins peut dégrader les performances des applications interactives par exemple en terme de délai d'établissement de la route.

De nombreux protocoles réactifs existent. Nous allons présenter les suivants :

1.6.3.2.1 Le protocole AODV (Ad hoc On Demand Distance Vector)

Le protocole AODV [5], créé par le concepteur de DSDV, est présenté comme une amélioration de ce dernier. Il réduit le nombre de diffusions de messages, et cela, en créant les routes lors du besoin, contrairement au DSDV, qui maintient la totalité des routes.

À la demande d'une route, la source diffuse une requête de route RREQ (Route REQest). Lorsque un nœud reçoit ce message, il établit une entrée dans sa table de routage (mémorise le

noeud précédent). Dès que la destination est trouvée, elle émet à son tour une requête de réponse de type RREP (Route REPly) à travers le chemin inverse vers la source qui peut commencer l'envoi des données.

La topologie de réseaux peut parfois être modifiée, pour pallier à cette problématique, AODV utilise également la numérotation de séquence qui permet aux noeuds d'utiliser les routes les plus récentes (fresh routes). Une entrée dans la table de routage contient : l'adresse de destination, l'adresse du noeud suivant, la distance en nombre de sauts, la numérotation de séquence, le temps d'expiration de chaque entrée dans la table.

Afin de maintenir des routes consistantes, une transmission périodique du message "HELLO" est effectuée. Si trois messages "HELLO" ne sont pas reçus consécutivement à partir d'un noeud voisin, le lien en question est considéré défaillant.

Le protocole AODV ne présente pas de boucle de routage, en outre il évite le problème de comptage à l'infini de l'algorithme de Bellman-Ford, ce qui offre une convergence rapide quand la topologie du réseau ad hoc change.

1.6.3.2.2 Le protocole DSR (Dynamic Source Routing)

Le protocole DSR [6] est un protocole réactif basé sur le routage par la source, c'est-à-dire que la source des données détermine le chemin complet par lequel les données vont transiter et ce dernier sera transmis avec les données. Dans chaque paquet transmis, il y a un champ qui contient la séquence de noeuds à suivre pour atteindre la destination.

Le protocole DSR repose sur deux mécanismes :

- Un mécanisme de découverte de route dont la source inonde le réseau par une requête RREQ. Au passage de paquet, chaque nœud insert un identificateur qui lui désigne. Si l'opération de découverte est réussite, l'initiateur (la source) reçoit un paquet réponse de route RREP qui liste la séquence de noeuds à travers lesquels la destination peut être atteinte. Le paquet requête de route contient donc un champ enregistrement de route, dans lequel sera accumulée la séquence des noeuds visités durant la propagation de la requête dans le réseau. Afin d'éviter la duplication de route, un identifiant unique par requête est mis en place. Il est constitué du couple : « adresse de l'initiateur, identificateur de requête ». Ainsi, si un noeud reçoit deux fois le même couple, il sera ignoré. La destination reçoit plusieurs requêtes RREQ mais effectue le choix de route le plus approprié.

- Un mécanisme de maintenance de route nécessaire afin d'assurer la validité des liens. Si la défaillance d'un lien est indiquée par un nœud, un message erreur de route (route error) est envoyé à la source du paquet. Alors, le noeud est supprimé de l'enregistrement de route et l'ensemble des routes possédant ce noeud sont tronqués en ce point. Puis réinitialisation de la procédure de recherche de destination.

DSR offre donc un routage qui se base sur la mémorisation des routes dans les paquets et qui assure l'absence de boucle de routage.

1.6.3.3 Les protocoles hybrides

Les protocoles de routage hybrides se présentent comme une alternative entre le routage proactif et le routage réactif. Il fait appel au protocole proactif pour apprendre le proche voisinage (deux ou trois sauts) et au protocole réactif pour la recherche de route.

1.6.3.3.1 Le protocole ZRP (Zone Routing Protocol)

ZRP [7] est un protocole de routage hybride (proactif/réactif). Le routage réactif se limite à la zone déterminée par le nombre de sauts ρ, ainsi si $\rho = 2$, pour chaque noeud, le routage proactif sera limité aux voisins qui se trouvent jusqu'à deux sauts. Dans ses spécifications, ZRP utilise DSDV avec des petites modifications comme protocole de routage proactif, mais n'importe quel autre protocole proactif pourrait être utilisé.

Pour le routage interzone, ZRP emploie un protocole réactif. Si le noeud source ne trouve pas la destination dans sa zone, il envoie une requête de route aux noeuds de bord de sa zone. Par exemple, si $\rho = 2$, les noeuds de bord de la zone sont ceux qui se trouvent à deux sauts à partir de la source, les noeuds qui se trouvent à un saut, sont dans la zone, mais ne sont pas des noeuds de bord. Chaque noeud qui reçoit la requête, vérifie si la destination se trouve dans sa zone. Si c'est le cas, il envoie une réponse de route au noeud source. Sinon, il envoie la requête aux noeuds de bord de sa zone qui vont exécuter chacun la même procédure.

IARP (ou IntrAzone Routing Protocol) repose sur un protocole à état de lien permet la construction, au niveau de chaque noeud interne à la zone, une table des routes optimales vers les voisins proches.

IERP (ou IntErzone Routing Protocol) se charge de rechercher les routes, à la demande, située dans la zone externe. Comme dans tout protocole réactif cette recherche se fait par inondation, mais dans un souci de contrôle de flux, un couple « adresse, identifiant de requête » est constitué sur chaque source. Si une requête a déjà été traitée par un noeud, il n'en tiendra plus compte par la suite. Cependant un troisième protocole, nommée BRP intervient pour affiner le contrôle de flux.

BRP (ou Bordercast Routing Protocol) permet un contrôle d'inondation du réseau plus précis. On se place sur un noeud source situant dans la zone IARP et connaît tous ses voisins, il est donc inutile de tous les interroger. Cela implique donc de se contenter d'interroger les noeuds de notre zone de routage. Cela limite dans un premier temps le broadcast et l'on parle plutôt de bordercast. Ensuite, il s'agit de déterminer pour chaque noeud si la destination appartient à sa zone de routage sinon il reproduit le traitement pour ces noeuds.

Afin de ne pas aboutir à une tempête de broadcast, chaque message BRP est identifié de manière unique (numéro de séquence et adresse origine). Ainsi, chaque noeud traité ignorera un autre message BRP (on parle de Loopback Termination).

Étant basé sur l'approche réactive, les déterminations de routes se font par des requête « RREQ », les réponses par « RREQ » et la détection d'erreurs est transmise par « RRER » (noeud inaccessible).

1.6.3.3.2 Le protocole ZHLS (Zone Based Hierarchical)

Le protocole ZHLS [8] est un protocole hybride hiérarchique basé sur la décomposition d'un réseau en zone. Contrairement à la plupart des protocoles dits hiérarchiques, il n'y a pas ici de représentant pour chaque zone. La topologie d'un réseau est ainsi partagée en deux niveaux :

- Un niveau noeud indique la façon dont les noeuds d'une zone sont connectés entre eux physiquement. Un lien virtuel peut exister entre deux zones s'il existe au moins un noeud d'une autre zone.
- Un niveau zone qui renseigne sur le schéma de connexion des différentes zones.

Ces différents niveaux entraînent donc deux différents types de liens : les liens inter-noeuds et les liens inter-zones.

L’adressage mis en place consiste en un identifiant de zone, un identifiant de nœuds et l'utilisation de LSP (Link State Packet) pour l’état des liens. Deux classes de LSP sont possibles : celles orientées noeuds pour lesquels un noeud donne des informations sur son voisin et celles orientées zone.

1.6.3.4 Avantages et inconvénients des protocoles

Les protocoles proactifs permettent le maintien d'une table de routage à jour par l'échange périodique des messages. Ces tables étant à jour, l'envoi de ces messages se fait rapidement. Cependant, on ne peut nier que l'émission régulière de ces paquets occupe une partie de la bande passante, qui risque d'augmenter en fonction du nombre de noeuds présents sur le réseau. Les protocoles réactifs sont, comme expliqués précédemment, basés sur une construction du réseau à la demande. Leur avantage se trouve donc dans le fait qu'il n'y a pas de surconsommation de bande passante comme pour les protocoles proactifs. En revanche, on peut s'inquiéter du délai nécessaire, avant l'envoi d'un message, pour trouver la route.

Les protocoles hybrides ont les avantages des deux approches précédentes en utilisant une notion de découpage du réseau. Cependant, il rassemble toujours quelques inconvénients des deux approches proactives et réactives.

1.7 Adaptation dans les réseaux mobiles ad hoc

On entend par adaptation la capacité de réagir face aux variations des contraintes de l’environnement. Dans le cas des réseaux ad hoc, les caractéristiques principales de l'environnement qui conduisent à proposer des adaptations protocolaires sont : l'énergie, le débit des liens, la qualité de service et la topologie [9].

1.7.1 Objectif d’adaptation

La mobilité des nœuds dans les réseaux ad hoc a un grand effet sur leurs performances. Les protocoles de routage employés dans ce type des réseaux doivent prendre en considération ce facteur de mobilité afin de garantir la continuité de communication et d’assurer la reprise

lorsqu'une déconnexion tout en conservant le plus possible l'énergie, la qualité de service et la bande passante.

L'objectif *de l'adaptation est de lier le comportement d'un protocole à l'environnement dynamique du réseau afin d'améliorer les performances.*

1.7.2 Effets attendus d'une adaptation

Une meilleure adaptation doit conserver le plus possible d'énergie. En général, une entité mobile possède une batterie autonome avec un degré connu et consommable d'énergie, les protocoles de routage doivent fournir des mécanismes pour réduire la consommation d'énergie tel que l'ajustement dynamique de la puissance de transmission en fonction de la distance ou diminuer le nombre d'émissions…etc.

La qualité de service est une autre caractéristique qui nécessite une adaptation, les protocoles de routage doivent prendre en considération la charge de trafic sur le réseau pour fournir la meilleure qualité de service.

La mobilité des noeuds génère des changements de connectivité. Un nœud d'un réseau ad hoc se déplace, il peut joindre ou quitter un réseau ad hoc à tout moment, c'est-à-dire que la topologie de réseau est ambiante, une bonne adaptation doit permettre au réseau de s'organise de manière dynamique et efficace pour assurer la stabilité des communications et pour améliorer les performances.

Dans un réseau ad hoc, la liaison est caractérisée par un état évolutif de fonctionnement, le débit d'émission de chaque lien radio change selon le temps et l'emplacement du noeud, voir peut devenir nul. Une adaptation doit lier le choix d'une route à l'existence d'une liaison, et peut également prendre en compte le débit réel de celle-ci.

1.7.3 Différents types d'adaptation

Le calcule des métriques permet aux protocoles de s'adapter à leurs environnement, soit par le changement de la manière de fonctionnement ou mode (exp : de proactive vers réactive, changer l'algorithme de sélection de chemin selon les exigences de la qualité de service…etc.) soit par changement des paramètres (augmenté la fenêtre d'anticipation si le réseau est stable…etc.). Selon la couche d'où provenir les paramètres pris en compte dans le calcule des métriques en distingue deux types d'adaptation : adaptation inter-couches et Auto-adaption[9].

1.7.3.1 Adaptation inter-couches

Dans ce type, les paramètres de calcul des métriques sont extraits de plusieurs niveaux de la pile des protocoles ou tout simplement de plusieurs couches différentes. Les architectures existantes pour l'échange d'informations entre les couches sont :

- une communication directe entre les couches même celles non adjacentes. Cette architecture nécessite l'ajout des nouvelles interfaces. Proposer par [10].
- Une interaction vers une entité intermédiaire : Cette architecture utilise une entité intermédiaire pour gérer les interactions inter-couches afin de garder et d'assurer le fonctionnement normal de la pile des protocoles classique. Proposer par [11].
- Une architecture à base de services pour minimiser les problèmes d'interactions et de communications entre ses différents composants. Cette architecture peut être qualifiée comme architecture sans couche. Proposer par [10].

Le problème de l'adaptation inter-couche est que ces systèmes sont plus difficiles à mettre en place et aussi la cohérence lorsqu'il y a un changement dans l'une des couches.

1.7.3.2 Auto-adaption

Pratiquement, les plus par des protocoles de routage fonctionnent dans une seule couche du modèle OSI, les paramètres pris en compte dans le calcul des métriques sont obtenus à partir de cette couche protocolaire. Cette séparation permet de garantir que l'évolution d'une couche n'affecte pas la cohérence de toute l'architecture parce que les couches sont indépendantes. L'échange d'information entre les couches se fait à travers des points d'accès bien définis et même standardisés.

1.7.4 Travaux d'adaptation sur les protocoles de routage

Selon les caractéristiques d'énergie, débit des liens, qualité de service et de topologie, plusieurs travaux qui basent sur la détection et le calcule des métriques sont réalisés pour l'adaptation du réseau ad hoc. La table 1.1 illustrée quelque protocole de ce type. D'autres protocoles utilisent la prédiction pour prendre une décision d'adaptation.

Adaptation à	Métrique	Nom de protocole
Energie disponible	Energie	Minimum Battery Cost Routing (MBCR). Min-Max Battery Cost Routing (MMBCR). MRPC (Maximizing network lifetime for reliable routing in wireless environments).
Etat de liaison	- Force du signal	Signal Stability-based Adaptive Routing (SSA). Associativity-Based Routing (ABR). Advanced signal strength based link stability estimation modèle (ASBM). Link quality of route.
	- Taux de perte	Auto-Rate Fallback (ARF). Loss-Di_erentiating ARF (LD-ARF). Collision-Aware Rate Adaptation (CARA). Receiver-Based Auto Rate (RBAR). Differential Rate Adaptation (DRA).
Charge de trafic	QOS	Dynamic Load-Aware Routing (DLAR). Load-Balanced ad hoc Routing (LBAR). Free-Degree Adaptive Routing (FDAR).
Dynamique de la topologie	Densité et mobilité	Adaptive Zone Routing Protocol (AZRP). Fast Optimized Link State Routing (F-OLSR). Adaptive Routing Protocol (ARPM). Cluster Source Routing (CSR).

Table 1.1 Synthèse des travaux sur l'adaptation en réseaux ad hoc (Inspiré de [9]).

1.7.5 Travaux basés sur la prédiction de la mobilité

La prédiction de la mobilité est la capacité d'évaluer la position future à partir de position passée. Cette technique de la prédiction de la mobilité peut être utilisée pour améliorer la gestion des réseaux mobiles.

Parmi les modèles de prédiction utilisés on distingue :

- Les modèles déterministes de premier ordre. Ce sont des modèles qui ne prennent en considération que la position et une vitesse fixée.

- Les modèles stochastiques visent à obtenir une prédiction correcte avec une haute probabilité et non exactement correcte.
- Les modèles à base d'historique : utiliser habituellement pour la prédiction de la macro-mobilité d'un terminal. En effet, l'historique des mouvements indiquer les trajectoires préférées par les utilisateurs.
- Les modèles hiérarchiques : ce sont des modèles très précisent utiliser généralement pour la micro-prédiction.

1.7.6 Quelques protocoles utilisés la prédiction de mobilité

- Vecteur de distance avec prédiction de mobilité (DV-MP) [12] représente le coût de lien comme la durée du lien prédit dans les approches à vecteur de distance et améliorer leurs performances.
- Kinetic Minimum Spanning Trees (KMST) [13]: KMST utilise le modèle stochastique de prédiction pour la construction de l'arbre de mesure qui utilise une stratégie de maintenance non périodique.
- Dead-Reckoning (DRM) [14]: améliorer les performances de DSR par l'utilisation de la prédiction de durée des liens au lieu de nombre de saut.
- Reliable On-Demand Routing Protocol (RORP) [15]: Cette approche ouvre des routes avec poids représentés la durée minimum de chaque lien comprise dans la route. Alors, la source choisit la route basée sur la durée maximum de lien.
- AODVMOvement Prediction Routing (AODV-MOPR) [16]: les noeuds sélectionnés pour établir la route entre la source et la destination sont sélectionnés dépendent de leur direction et vitesse. Ce qui améliorer la stabilité des routes dans AODV.

1.7.7 La mise en œuvre d'une adaptation

Un processus global de l'adaptation [9] prend en considération trois ensembles d'éléments :

- l'environnement réseau, qui est perceptible au travers de métriques,
- le comportement à adapter, c'est-à-dire les algorithmes à appliquer selon les valeurs des métriques, et
- les performances que l'on cherche à optimiser.

1.7.7.1 Les Métriques

La métrique est une mesure qui indique l'état d'un noeud, de son voisinage ou bien de l'ensemble du réseau, par exemple la puissance d'énergie, la puissance de transmission, la force du signal, le trafic et la mobilité.

La mesure peut consister en une combinaison de plusieurs paramètres comme le nombre des noeuds, le nombre des liens, l'état d'énergie, etc. ; enfin, la métrique donne des valeurs qui peuvent être utilisées pour adapter un comportement de protocole.

Plusieurs métriques sont utilisables par l'algorithme d'adaptation, par exemple un protocole de routage adapte son fonctionnement à la densité du réseau et à la mobilité.

1.7.7.2 Type des métriques

Le calcule des métriques ce fait selon trois vues :

- Une vue locale telle que le nombre de changements de connexions.
- Une vue de voisinage telle que la métrique de vitesse relative entre deux nœuds [17]

$$RS(i,j,t) = \left| \overrightarrow{V}_i(t) - \overrightarrow{V}_j(t) \right|$$

- Une vue réseau telle que l'éloignement entre 2 nœuds. Le concept est spécifié par une fonction de la distance entre 2 noeuds i et j, F (dij(t)).

1.7.8 Les modèles de mobilité

Les modèles de mobilité sont des modèles synthétiques utilisés pour la simulation de comportement des nœuds. Ces modèles sont répartis en deux classes, selon le mode de déplacement des nœuds. Dans une première classe, le mouvement d'un nœud se fait indépendamment des autres, en appels modèles de mobilité par entité, tandis que dans la deuxième classe le mouvement des nœuds se fait en groupe et en appels les modèles de mobilité par groupe.

1.7.8.1 Les modèles par entité

Dans ces modèles, chaque noeud se déplace indépendamment des autres. Le mouvement est soit aléatoire sans prend en considération l'état précédent (sans mémoire), soit par la prise en

compte des états précédents. Dans ce deuxième cas, l'état à chaque instant (vitesse, direction et position) est en fonction de l'état précédent (avec mémoire).

1.7.8.1.1 Les modèles sans mémoire

Dans ces modèles, un noeud choisit une position et une vitesse d'une façon absolument aléatoire et sans aucune mémoire du passé. Dans ces modèles, on peut avoir fréquemment des comportements extrêmes des noeuds comme un arrêt soudain, une accélération soudaine et des tours brutaux. Les modèles existants dans cette catégorie sont :

1.7.8.1.1.1 Modèle Random Walk (RW)

Développé par Zonoozi et Dassanayake [18] pour la simulation de mouvement aléatoire. Dans ce modèle, chaque noeud choisit aléatoirement un angle de direction dans $[0,2\pi]$ et une vitesse dans $[V\min, V\max]$. Le déplacement du noeud se fait pendant un temps *t* ou d'une distance *d*. Un noeud mobile qui atteint la limite de la simulation, recommencer le processus et choisit une nouvelle direction et une nouvelle vitesse indépendamment du choix précédent et se déplace dans sa nouvelle trajectoire.

1.7.8.1.1.2 Modèle Random Waypoint (RWP)

Défini par Johnson et Maltz [19]. Les nœuds sont distribués d'une manière uniforme ou équilibrée sur la surface de simulation. Le Random Waypoint est pour modéliser tous les scénarios dans lesquelles, les noeuds se déplacent vers une destination, prennent un repos en arrivant, avant de se déplacer vers une autre destination et ainsi de suite. Dans ce modèle chaque noeud choisit aléatoirement, comme destination un point de coordonnées (x, y) dans la surface de simulation, et une vitesse entre *0* et *Vmax.*

Random Waypoint est le modèle le plus utilisé dans les simulations due à la facilité de son implémentation, mais il n'est pas adapté au comportement complexe des nœuds.

Certaines études signalons le problème de convergence du temps de simulation, les auteurs proposent une solution simple qui est de choisir une valeur minimale pour la vitesse.

1.7.8.1.1.3 Modèle Random Direction

Développer par Royer el al [20]. C'est une modification de modèle précédent RWP. Dans RWP la probabilité qu'un noeud mobile choisit une nouvelle destination localisée au centre de la surface de la simulation ou de passer à travers le centre est haute. Le Random Direction a été créé pour éviter l'effet de concentration des noeuds au centre produit par le Random Waypoint. Dans ce modèle, chaque noeud choisit aléatoirement un angle de direction dans $[0,2\pi]$ et une vitesse dans $[V\min, V\max]$.

La différence entre ce modèle et le Random Walk est qu'ici le noeud ne voyage pas pendant un certain temps ou d'une certaine distance, mais se déplace suivant la direction choisie jusqu'à atteindre le bord de la surface de simulation où il prend un temps de repos. Une fois le temps de pause terminé, le noeud répète le même processus.

1.7.8.1.1.4 Modèle Restricted Random Waypoint

Ce modèle a été, pour la première fois, décrit dans [21]. L'idée de ce modèle de mobilité est extraite du fait que la plupart des gens se déplacent pour un certain temps dans une même localité avant d'aller vers une autre localité. Donc dans ce modèle, la surface de simulation contient des rectangles qui représentent des villes liées par des autoroutes. Chaque nœud utilise le Random Waypoint pour se déplacer dans l'une des villes un certain nombre de fois spécifiées par un paramètre, avant de voyager vers une autre ville où il va se déplacer pour un certain moment et ainsi de suite .

1.7.8.1.1.5 Modèle Brownian Motion

Brownian Motion [22] est totalement un modèle aléatoire du mouvement. La direction de mouvement est une variable aléatoire continue entre 0 et 2 et la vitesse est aussi aléatoire à tout temps donné. Chaque noeud mobile après une période aléatoire déplace dans un certain bord de la surface de simulation où le mouvement est complètement isolé.

1.7.8.1.1.6 Modèle Manhattan Grid :

Le modèle Manhattan Grid [23] est proposé pour modéliser les villes avec les routes et leur intersection. Au début, chaque nœud choisit un point aléatoire dans une route quelconque

pour le démarrage. Puis il choisit une destination aléatoire et prend une direction vers cette destination, sans vitesse définie au préalable.

Lorsqu'un nœud mobile atteint sa destination, après une période aléatoire, il reprend le processus.

1.7.8.1.2 Les modèles avec mémoire

Dans ces modèles, appelés aussi corrélés, la vitesse et la direction à chaque instant, dépendent de l'instant précédant.

1.7.8.1.2.1 Modèle Boundless

Dans ce modèle, la position et la vitesse d'un noeud à tout instant (*t+At),* dépendent de la position et de la vitesse à l'instant t. La position (x, y) du mobile et sa vitesse í sont mises à jour chaque At unité de temps comme suit [24] :

v(t+At)= min [max(v(t) + Av, 0), Vmax]; (1)

0(t+At)= 0(t) + A0 ; (2)

$V\max$: La vitesse maximale.

Av : Le changement de vitesse distribuée uniformément entre $[-A\max * At, A\max * At]$.

$A\max$: Accélération maximale qu'un noeud peut avoir.

AO : La variation de la distribution, distribuée uniformément entre $[-a * At, a * At]$

a : Valeur maximale du changement d'angle q'un noeud peut avoir.

Le Boundless a un effet de bord différent des modèles déjà cités. Un noeud qui atteint le bord, continue pour sortir et rentrer de l'autre côté de la surface de simulation. Cette action modélise un nœud qui sort définitivement de la simulation avec un nouveau qui arrive en même temps pour le remplacer.

1.7.8.1.2.2 Gauss Markov

Le modèle de mobilité Gauss Markov a été proposé dans [25]. Gauss Markov est un modèle de mobilité semblable à Boundless, dans le sens que la position et la vitesse à tout instant,

dépendent de la position et de la vitesse au moment précédent. La vitesse, la direction et la position d'un noeud varient selon les formules suivantes :

$$\begin{cases} v_t^x = \alpha v_{t-1}^x + (1-\alpha)\upsilon^x + \sigma^x \sqrt{1-\alpha^2}\, w_{t-1}^x \\ v_t^y = \alpha v_{t-1}^y + (1-\alpha)\upsilon^y + \sigma^y \sqrt{1-\alpha^2}\, w_{t-1}^y \end{cases}$$

Avec

$\overline{V}_t = \left[v_t^x , v_t^y \right]^T$ et $\overline{V}_{t-1} = \left[v_{t-1}^x , v_{t-1}^y \right]^T$ sont les vitesses dans t et t-1 respectivement.

$\overline{W}_{t-1} = \left[W_{t-1}^x , W_{t-1}^y \right]^T$ processus Gaussian aléatoire.

$\overline{\alpha} = \left[\alpha^x , \alpha^y \right]^T$, $\overline{\upsilon} = \left[\upsilon^x , \upsilon^y \right]^T$ et $\overline{\sigma} = \left[\sigma^x , \sigma^y \right]^T$ vecteurs de niveau de mémoire.

Si α = 0 un effet absolument aléatoire est obtenu et le modèle est semblable à Random Walk.
Si α = 1 un effet linéaire est obtenu la vitesse du nœud à l'instant t égale à la vitesse à t-1.
Pour d'autres valeurs de α, la vitesse et la direction à chaque instant, dépend de vitesse V_{t-1} et de variable aléatoire de Gausse W_{t-1} de l'instant t-1.
Pour assurer qu'un noeud ne reste pas près d'un bord de la simulation, les noeuds sont poussés loin du bord quand ils sont à moins d'une certaine distance du bord. Cet effet est réalisé en modifiant la valeur de la direction moyenne *d* au cours de la simulation. Par exemple, lorsqu'un noeud est proche du bord droite, la valeur de *d* change à 180^o, alors la nouvelle direction du noeud l'éloigne du bord de la simulation.
On peut bien remarquer pour les deux modèles déjà cités (Boundless et Gauss Markov), que les valeurs de la vitesse, de la direction et de la position à chaque instant, dépendent des valeurs à l'instant précédent ce qui crée un mouvement plus souple des noeuds.

1.7.8.1.2.3 Markov Random Path

Le Markov Random Path [26], appelé aussi *A Probabilistic Version of Random Walk* et qui était, pour la première fois, proposé par Chiang , utilise une chaîne de Markov pour modéliser le mouvement d'un noeud. La chaîne de Markov est une suite de variables aléatoires (Xn) telle que, pour chaque n, Xn+1 soit indépendante de Xk, pour k= n-1, et dépend uniquement de Xn.

Dans ce modèle, les mouvements des noeuds sont séparés en directions horizontales et verticales. Chaque direction représente une variable aléatoire. La Figure 1.2 montre le schéma de la chaîne de Markov utilisée pour faire varier les coordonnés d'un noeud dans chaque mouvement :

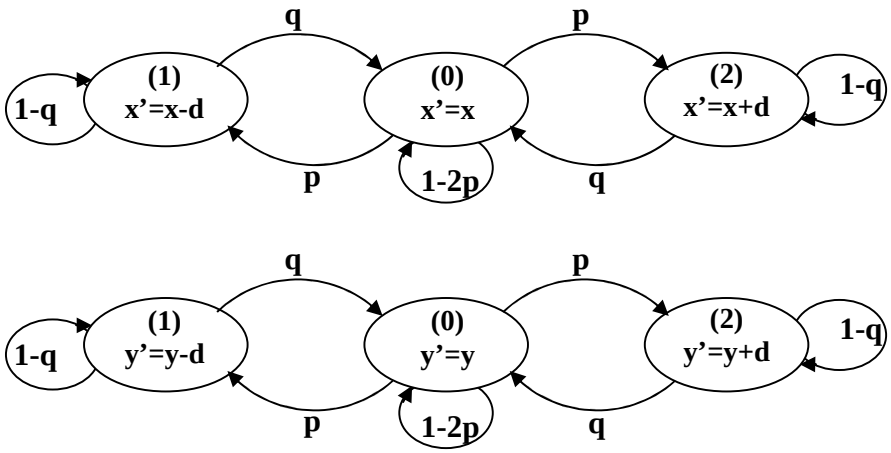

Figure 1.2 : Schéma de passage pour le Markov Random Path.

- L'état (0) : Garder les mêmes coordonnés (x, y).
- L'état (1) : La position est décrémentée.
- L'état (2) : La position est incrémentée.
- p et q sont les probabilités de passage d'un état à l'autre.

Le déplacement est d'une distance *d* fixe. On a donc deux chaînes de Markov, une pour le déplacement suivant l'axe des *x* et l'autre pour le déplacement suivant l'axe des *y.* Selon la valeur des probabilités *p* et *q,* les coordonnés (x, y) d'un noeud vont augmenter, diminuer ou rester stables. Par exemple, pour une valeur élevée de *p,* un noeud a plus de chance de se déplacer en avant plutôt qu'en arrière

1.7.8.1.2.4 City Section (CS)

Le City Section modélise le déplacement des noeuds (Voitures, camions, gens, . .) dans une ville. Dans ce modèle [27], la surface de simulation représentée par une grille, symbolise des rues horizontales et verticales dans une ville. Au lieu de spécifier une vitesse maximale aux noeuds, on spécifie une vitesse limite pour les routes. Chaque noeud commence la simulation sur

un point prédéfini, qui est l'intersection de deux routes. Le noeud choisit aléatoirement une destination, qui est aussi l'intersection de deux routes, et commence son voyage vers cette destination en choisissant le chemin qui nécessite le moins de temps pour arriver. À son arrivée, le noeud choisit une nouvelle destination et répète le même processus, sans prendre un temps de repos. Dans ce modèle, on peut utiliser les cartes géographiques réelles avec la possibilité d'avoir des règles de sécurité de conduite comme la distance entre deux noeuds consécutifs ou une limitation de la vitesse d'un nœud.

1.7.8.1.2.5 Le modèle de mobilité avec obstacles

Le modèle de mobilité avec obstacles [28] a été conçu pour modéliser le mouvement des nœuds mobiles dans les terrains qui ressemblent à des topographies réelles. Des objets modélisent les bâtiments et d'autres structures qui empêchent les mouvements des nœuds, ainsi que leur transmission sans fil.

Les déplacements des nœuds sont représentés par le diagramme de Voronoï. C’est un graphe planaire qui modélise les mouvements. Les nœuds choisissent les routes les plus courtes dans le diagramme de Voronoï. Les nœuds peuvent se déplacer à l'intérieur des bâtiments, car le plus court chemin entre deux endroits peut exiger le passage par l'intérieur d'un bâtiment.
Le placement des objets et des chemins qui relient sont calculés au début de la simulation et ne changent pas pendant toute la simulation. Les nœuds sont distribués au hasard le long des chemins, ils choisissent une destination, puis ils se déplacent vers cette destination en suivant le chemin le plus court à partir de sa position courante.

Après être arrivé à sa destination, un nœud prend une période de temps. Il choisit alors une nouvelle destination, calcule le chemin le plus court pour l'atteindre, et reprend le mouvement.

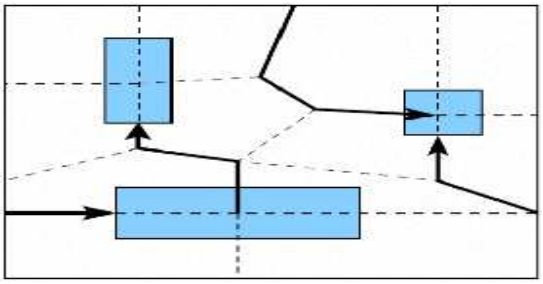

Figure 1.3 : Mouvements avec obstacles utilisant le diagramme de Voronoï.

1.7.8.2 Les modèles de groupe

Les modèles de groupe sont introduits pour couvrir les insuffisances des modèles par entité dans la modélisation de l'aspect collectif de quelques applications

1.7.8.2.1 Le modèle exponentiel aléatoire corrélé

Dans ce modèle, le mouvement de chaque groupe est contrôlé indépendamment des autres groupes. À chaque étape de temps, un groupe se déplace d'une distance aléatoire dans une direction aléatoire. Chaque noeud change ses coordonnées polaires, qui sont une distance et un angle.

1.7.8.2.2 Modèle de mobilité de colonne

Dans ce modèle (Column Mobility Model) [29], chaque groupe des noeuds peut avoir une ou plusieurs références. Une référence est un nœud du groupe qui a pour rôle de guider les autres nœuds pendant leur déplacement. Au début de la simulation, les références de chaque groupe sont placées d'une façon formant une colonne et chaque noeud est placé en relation avec sa référence, autour de laquelle, il a le droit de se déplacer en utilisant l'un des modèles de mobilité par entité. Une référence peut avoir un seul noeud autour d'elle. La position de l'axe des références change de la manière suivante (Figure 1.4) :

Nouvelle_position (références) = ancienne_position (références) + vecteur anticipé.

Le vecteur anticipé est calculé suivant un angle aléatoire entre 0 et *it* radian (puisque le déplacement est seulement en avant) et une distance aléatoire.

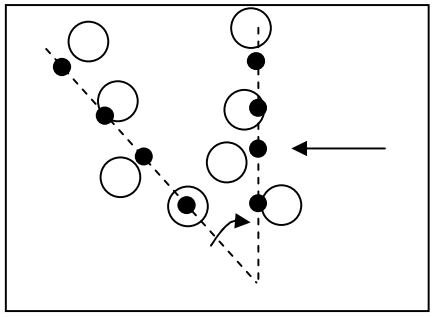

Figure 1.4 : Mouvement des noeuds utilisant le modèle Colonne.

1.7.8.2.3 Modèle de mobilité de communauté nomade (NCMM)

Dans ce modèle, chaque groupe des noeuds possède un seul point référence en commun [29]. Les noeuds de chaque groupe se déplacent autour de leur point référence en utilisant un modèle de mobilité par entité (Random Walk) et ne peuvent pas dépasser une certaine distance, précisée dans les paramètres, qui est la distance maximale entre un noeud et sa référence. Le déplacement d'une référence se fait aussi suivant un modèle singulier. La Figure 1.5 montre le déplacement d'un groupe de trois noeuds, présentés en couleur, autour de leur référence qui est en noir. Les noeuds ainsi que la référence, utilisent le Random Walk.

Quand le point de référence change, tous les nœuds mobiles dans le groupe se déplacent vers le nouveau secteur défini par le point de référence et commencent à errer autour du nouveau point de référence (Figure1.6).

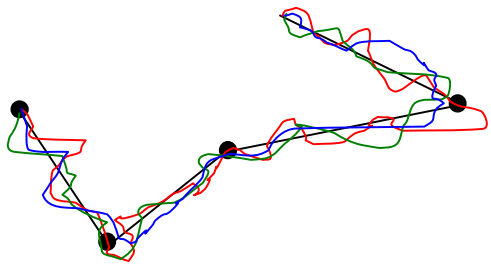

Figure 1.5 : Déplacement d'un groupe des nœuds.

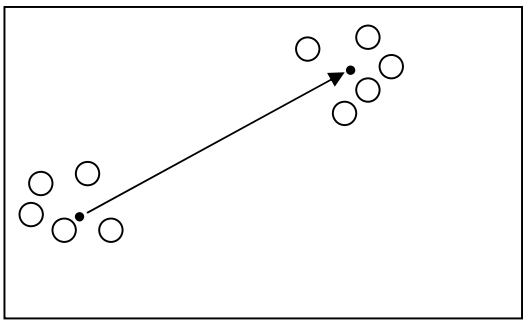

Figure 1.6 : Modèle de mobilité de communauté nomade.

1.7.8.2.4 Modèle de mobilité de poursuite

Défini par Sanchez . Le modèle (pursue models) [26] de mobilité de poursuite contient, comme *Nomadic Community,* un seul point de référence. La différence est que la référence joue ici le rôle d'une cible qui est poursuivie par les autres noeuds, comme le mouvement d'un groupe de policiers essayant d'attraper un voleur. La position de chaque noeud dans le groupe varie de la manière suivante:

Nouvelle_position = position_ancienne + accélération (cible) + vecteur_aléatoire.

Accélération (cible) est une information sur le déplacement de la cible et le vecteur aléatoire est le mouvement d'un noeud selon un modèle de mobilité singulier (Random Walk par exemple). Ce mouvement est limité puisqu'il s'agit de poursuivre la cible sans la dépasser. La Figure 1.7 illustre le déplacement d'un groupe utilisant le modèle Purse. Le noeud en blanc représente la référence.

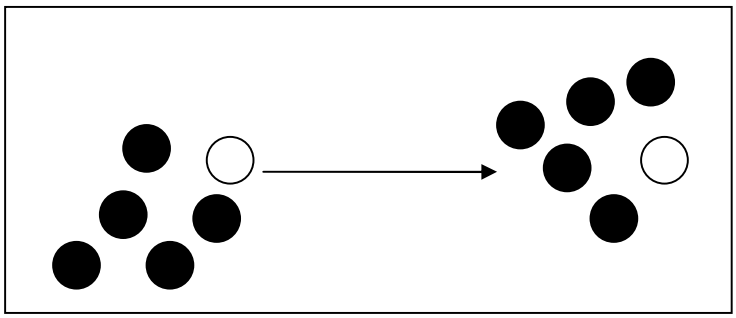

Figure 1.7 : Déplacement selon le modèle Purse.

1.7.8.2.5 Modèle de mobilité d'un groupe avec point de référence (RPGM)

Défini par Hong, Gerla et al [30]. Le modèle de mobilité d'un groupe avec point de référence noté RPGM (*Referenced Point Group Mobility Model)* représente le mouvement aléatoire d'un groupe des nœuds aussi bien que le mouvement aléatoire de chaque nœud individuellement dans le groupe. Les mouvements du groupe sont basés sur le chemin parcouru par un centre du groupe qui est utilisé pour calculer le mouvement du groupe par l'intermédiaire d'un vecteur de mouvement $\overrightarrow{GM}$ qui peut être choisi aléatoirement ou être prédéfini.

Le mouvement du centre du groupe caractérise complètement le mouvement de son groupe (la direction et la vitesse). Les nœuds individuels se déplacent aléatoirement par rapport à

leurs propres points de référence prédéfinis, dont les mouvements dépendent du mouvement du groupe.

La Figure 1.8 illustre le mouvement de 3 noeuds utilisant le modèle RPGM. Le modèle RPGM a été conçu pour faire face à des scénarios tels qu'une avalanche après laquelle une équipe de secours se composant d'humains et des chiens travaille en coopération. Les guides humains (centre des groupes) tendent à définir un chemin général puisqu'ils connaissent habituellement l'endroit approximatif des victimes. Chacun des chiens crée son propre chemin aléatoire autour du secteur général choisi par leurs guides humains.

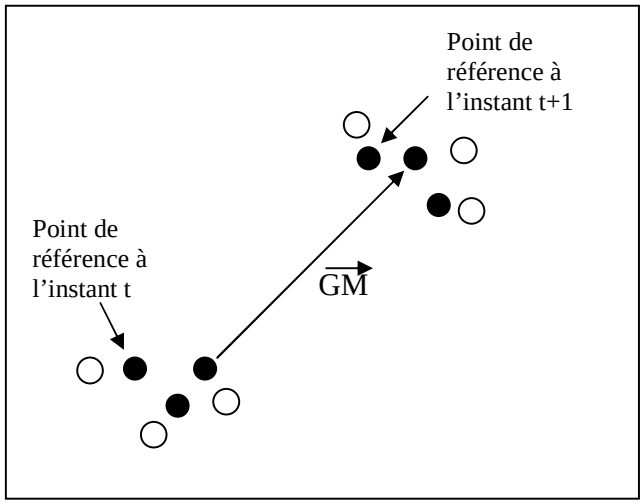

Figure 1.8 **:** Mouvements des nœuds utilisant le modèle RPGM.

1.8 Conclusion

Dans ce chapitre nous avons présenté quelques notions sur les réseaux ad hoc et des protocoles de routage du groupe MANET qui ont été proposés pour assurer le service de routage dans les réseaux mobiles ad hoc.

Les protocoles proposés sont généralement classés en deux catégories : les protocoles proactifs et les protocoles réactifs. Les protocoles des deux catégories essayent de s'adapter aux contraintes imposées par les réseaux ad hoc. Les réseaux ad hoc doivent s'organiser automatiquement, rapidement, et pouvoir s'adapter aux conditions de propagation, au trafic et aux

différents mouvements dans le but d'assurer la connectivité du réseau malgré l'absence d'infrastructure et la mobilité des nœuds.

Nous avons introduit aussi l'adaptation et quelques notions en relation (métrique…..). La prédiction de la mobilité est très importante, c'est un challenge dans les réseaux ad hoc. Plusieurs modèles sont proposés par les chercheurs et appliqués pour améliorer le fonctionnement des réseaux mobiles ad hoc.

Dans ce chapitre nous avons vu aussi quelques modèles de mobilité utilisés pour la simulation des mouvements des nœuds, ces modèles sont répartis en deux classes selon le mouvement des nœuds, soit individuel (par entité) soit collectif (par groupe).

Dans le chapitre suivant nous allons voir les concepts associes aux agents mobiles.

Chapitre II

Les agents mobiles

2.1 Introduction

Les systèmes multi-agents ou SMA forment une branche de l'Intelligence artificielle dans laquelle des métaphores sociologiques ou biologiques sont employées pour la conception et la mise en oeuvre des systèmes artificiels intelligents. Un agent logiciel est une entité autonome capable de communiquer, disposant de connaissances et d'un comportement privé ainsi que d'une capacité d'exécution propre [31].

L'informatique mobile peut indifféremment désigner la mobilité matérielle ou la mobilité logicielle. La mobilité matérielle est le déplacement d'un terminal physique, tel qu'un ordinateur portable ou un téléphone. Alors que la mobilité logicielle est le déplacement d'un programme logiciel entre deux terminaux physiques. L'entité logicielle mobile est alors appelée composant, agent, application mobile [37].

De nos jours, les agents mobiles acquièrent de plus en plus d'importance qui revient à leur capacité de migration d'une manière autonome à travers un réseau dans le but d'avoir accès aux ressources et services distants.

Les agents mobiles représentent un outil adéquat pour l'adaptation des systèmes aux environnements décentralisés et évolutifs. Ils sont dotés des caractéristiques qui augmentent la tolérance aux pannes telle que la capacité d'autonomie (indépendance lors de l'exécution), de proactivité (capacité à prendre des décisions de manière autonome) et de communication.

Dans cette section nous avons introduit quelques concepts liés à l'agent mobile et leur application dans les environnements mobiles.

2.2 Historique

L'agent mobile est un nouveau paradigme pour la conception des systèmes distribués. Il est basé sur le concept du code mobile, l'introduction de la mobilité est pour l'amélioration des approches traditionnelles de client-sreveur.

2.2.1 Code mobile

La mobilité du code se traduit par le transfert de code d'une application d'un site source vers un site destination. Puis le lancement de l'exécution de ce code sur le site destination. L'idée d'envoyer un code aux différentes machines via un réseau est motionnée la première fois par Rulifson et son groupe en 1969 [32]. Ils introduisent le langage DEL (Decode-Encode-Language). L'idée est de télécharger d'un programme écrit en DEL au début de l'ouverture d'une session entre deux hôtes distants, ce programme permet de contrôler la communication en utilisant une petite bande passante.

2.2.2 Évaluation à distance

Dans l'évaluation à distance, c'est le site du client qui dispose du savoir-faire propre au service à réaliser. Les ressources et l'unité d'exécution étant au démarrage sur le serveur, la mise en route du service est réalisée après que le serveur ait reçu le code à exécuter. Les résultats sont renvoyés une fois le service achevé, le code et l'unité d'exécution sont alors supprimés.

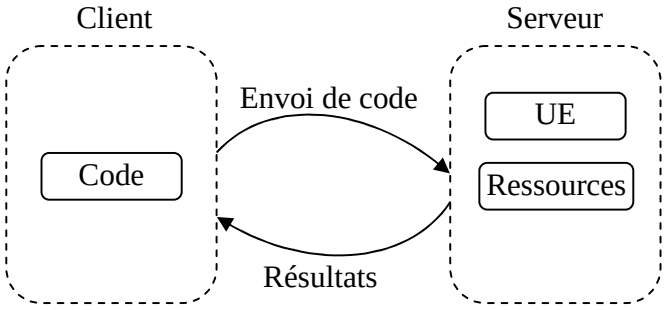

Figure 2.1 : Envoi de savoir-faire.

2.2.3 Code à la demande

Dans ce cas, le client dispose de l'unité d'exécution et des ressources, mais pas du savoir-faire qui va être récupéré auprès du serveur. Il s'agit donc de l'inverse du cas précédent. Ainsi, un client adresse une requête uniquement pour récupérer un code précis afin de l'exécuter localement avec les ressources présentes.

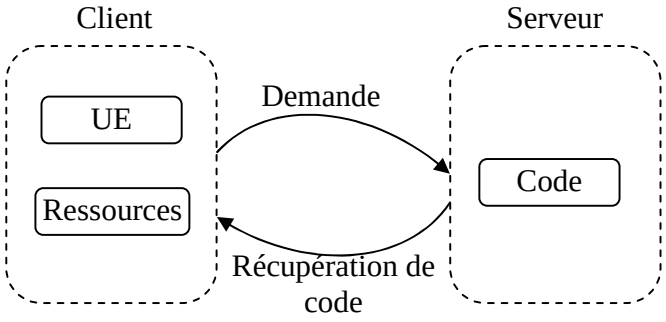

Figure 2.2 : Récupération de savoir-faire.

2.2.4 Objet mobile

L'idée est d'ajouter un peut d'autonomie à un message. Cette technique permet de créer un message qui peut migrer ver des hôtes distants, ce message contenir des données et de code qui doit exécuter sur chaque serveur. L'avantage par rapport aux techniques précédentes est que le message peut transférer pas à un seul hôte distant, mais elle capable de visiter tous les serveurs de réseau.

2.2.5 Processus mobile

Dans ce schéma, un code en cour d'exécution sur un hôte est capable de capturer l'état d'exécution et migrer ver un autre. Durant l'exécution du service, tous les éléments se retrouvent sur le site serveur. L'intérêt de cette méthode est qu'elle permet d'appliquer le même code à des ressources différentes réparties sur le réseau sans avoir télécharger ce code auprès du client à chaque fois.

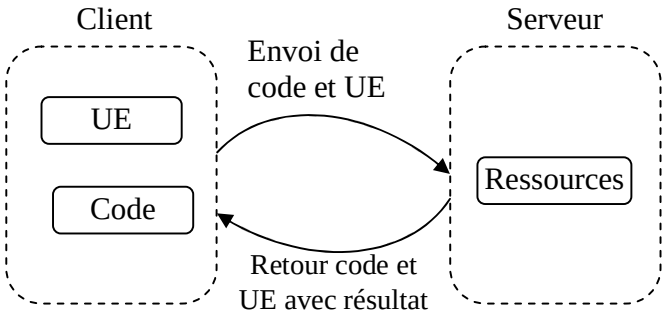

Figure 2.3 : Migration de code.

2.2.6 Agents mobiles

Le schéma de mobilité des agents mobiles dérive de deux domaines différents, c'est-à-dire les agents venant de l'intelligence artificielle avec les systèmes multi-agents [31] et des systèmes distribués avec la migration de processus [33]. Nous pouvons dire que le schéma à agents mobiles est une généralisation de la migration de processus où le déplacement est à l'initiative même du code.

Les agents mobiles ont été introduits initialement en 1994 avec l'environnement Telescript [34] qui permettait à des processus de choisir eux-mêmes de se déplacer sur les sites d'un réseau afin de travailler localement sur les ressources.

2.3 Définition

Un agent mobile [35, 36] est un agent logiciel qui peut se déplacer d'un site à un autre en cours d'exécution pour se rapprocher de données ou de ressources. Il se déplace avec son code et ses données propres, mais aussi avec son état d'exécution. L'agent décide lui même et de manière autonome ses mouvements. En pratique, la mobilité ne se substitue pas aux capacités de communication des agents, mais elle complète (la communication distante, moins coûteuse dans certains cas, reste possible).

Un agent mobile consiste en un code et un état, contient son propre contrôle. il est capable de déplacer son code et son état entre les machines, ce déplacement est appelé migration. Une

migration devrait inclure toutes les parties « essentielles » d'un agent, de cela il peut continuer à fonctionner d'une manière autonome.

2.4 Caractéristiques des agents mobiles

Les agents mobiles peuvent être considéré comme un nouveau paradigme de conception des systèmes distribués, ils possèdent plusieurs caractéristiques telles que :

1- Les agents mobiles peuvent être utilisés dans des réseaux hétérogènes où il est impossible d'identifier la fiabilité des connexions ou la sécurité de réseau.
2- Un agent mobile qui se déplace vers une machine distante transporte son code, ses données et son état d'exécution.
3- La décision de migration vers un autre emplacement revient seulement aux agents.
4- L'agent mobile capable de migrer plusieurs fois vers différentes destinations afin d'accomplir leur tâche, par contre au code mobile qui est transformé une seule fois vers la destination.
5- Un agent mobile capable de suspendre son exécution à un point donné, de migrer ver un autre noeud, puis de relancer son exécution depuis le même point.

2.5 Différents types de mobilité d'agent

Comme il est signalé précédemment, un agent mobile consiste en un code, données et un contrôle. La mobilité d'agent nécessite la précision des ressources à transporter avec l'agent.

Pour les données, chaque système d'agent mobile doit fournir un mécanisme pour transférer les données, tel que les variables globales ou le tas de données. Un agent mobile a deux états : un état interne et un état externe.

L'état interne d'un agent comprend tous ces états qui existent, indépendamment de toute entité à l'extérieur de l'agent. Ils peuvent être changés sans nécessairement exiger un changement correspondant dans tous les objets à l'extérieur. L'état interne inclut, par exemple, une variable de cycle d'une boucle ou un compteur du programme de l'agent courant.

Contrairement à l'état interne, l'état externe comprend les parties de l'état qui correspondent à quelques entités extérieures. L'état externe comprend, par exemple, des fichiers ouverts ou des rapports de la communication (local ou éloigné) aux autres agents ou autres

processus. Il est alors certain qu'idéalement l'état interne complet de l'agent devrait être transféré au cours de la migration.

De même façon l'agent mobile doit fournir un mécanisme pour transférer le code d'un agent, bien que les approches varient largement. Dans la première approche, l'hôte envoyant peut transférer tout le code nécessaire à l'hôte destinataire avec les données de l'agent. Cette approche garantie que l'agent a tout ce qu'il a besoin à son arrivée (importante dans les systèmes espérant supporter le traitement sans connexion). Dans la deuxième, il peut aussi envoyer les données simplement et attend que la destination demande les modules de code spécifiques auxquels l'agent a besoin.

Il existe deux types ou degrés de mobilité des applications [38] : faible et forte.

2.5.1 Mobilité faible

La mobilité faible intervient au cours de l'exécution d'une application [37]. Elle consiste à transférer le code et les données seulement. La migration d'une machine source vers une machine destination passe par :

- Tout d'abord, l'interruption de l'exécution de l'application sur le site source,
- puis le transfert du code et de l'état courant des données utilisées par l'application du site source vers le site destination,
- et, enfin, la reprise de l'exécution de l'application sur le site destination. Arrivée sur le site destination, l'application mobile reprend son exécution depuis le début, tout en possédant les valeurs mises à jour de ses données.

De ce fait, la plupart des plates-formes à agents mobiles qui se basent sur des systèmes orientés objet fournissent une mobilité faible à leurs agents tels que Aglets, Ajanta, Concordia et Mole.

2.5.2 Mobilité forte

En plus des informations prises en compte par la mobilité faible (code et état courant des données), la *mobilité forte* prend en compte l'état courant de l'exécution de l'application [37]. Ainsi, une application fortement mobile, qui se déplace au cours de son exécution d'un site source à un site destination, commence son exécution sur le site destination au point où elle a été interrompue sur le site de départ. La mobilité forte d'une application se traduit alors par:

- L'interruption de l'exécution de l'application sur le site source,
- puis le transfert du code, de l'état courant des données et de l'état courant de l'exécution de l'application du site source vers le site destination,
- Et enfin, la reprise de l'exécution de l'application sur le site destination. Arrivée sur le site destination, l'application mobile ne reprend pas son exécution depuis le début, mais elle poursuit au point même où elle a été interrompue sur le site source.

La plupart des mises en œuvre en mobilité forte se présentent sous forme de migration de processus (flot d'exécution de l'application) tels que Nuttall et Milojicic, et d'autres sous forme de migration de thread tels que Ara, D'Agent, Agent Tcl et Telescript.

2.6 Avantages des agents mobiles

Parmi les avantages des agents mobiles, on peut citer :

1. **Flexibilité** : un agent mobile peut déplacer entre machines de différent système tout en reste opérationnel, en plus la taille de population d'agents peut être s'adapter facilement à la taille du réseau.
2. **Performance et Efficacité :** l'utilisation des agents mobile augmente les performances des applications distribuées. Elle fournit une meilleure utilisation des ressources, permet une haute qualité de services, réduit latence et consommation de la bande passante. Les agents affectent moins les performances de chaque machine puisqu'ils peuvent travailler sur les ressources ayant uniquement un rapport avec leur champ de vision. Le gain au niveau de l'échange d'informations est notable (la

migration de l'agent vers l'information au lieu de transférer un grand volume d'information).

3. **Fiabilité** : un système multi agent est tolérant aux fautes. Si un agent est hors-service, il reste d'autres agents qui peuvent accomplir leur fonction ou même régénérer l'agent si le système donne la possibilité.
4. **Portabilité** : les agents supportent plus facilement les systèmes distribués et hétérogènes, plusieurs environnements de développement des agents mobiles support plusieurs langages de programmation en même temps ce qui permet une compatibilité avec plusieurs systèmes.
5. **Tolérance aux pannes** : les agents mobiles sont basés sur une exécution asynchrone et une certaine autonomie. La déconnexion fréquemment apparaître dans les environnements dynamiques entre les différents sites communicants n a pas de grand effet sur le fonctionnement de système.

2.7 Inconvénients des agents mobiles

Si les systèmes d'agents mobiles ont des avantages indéniables, ils ont également des inconvénients notables.

1. **Manque d'infrastructure et de standards**. Les agents ont besoin du support fixe d'une « plate-forme », or aucun standard n'est réellement appliqué et leur conception varie encore beaucoup d'un système à l'autre. De même, les difficultés non résolues empêchent la formation d'une infrastructure suffisamment fournie pour permettre le développement d'applications économiquement intéressantes.
2. **Continuité d'exécution sur différentes machines :** L'un des problèmes des agents mobiles est celui de la continuité de l'exécution d'un agent sur plusieurs machines de différentes technologies qui support différentes plateformes ou langages de programmation de développement des agents.
3. **Sécurité** : la sécurité consiste à empêcher des accès, et/ou des modifications, non autorisés aux éléments d'un système informatique. l'utilisation des agents mobile générer plusieurs problèmes de sécurité liés a la nature des systèmes d'agents mobiles, comme système, fonctionnant en environnement ouvert (c.-à-d. Internet) et

non fiable (réseaux mobiles). Il existe plusieurs formes d'intrusion telle que les agents malicieux qui peuvent consommer des ressources (mémoire, bande passante, …), accès aux informations confidentielles et même attaques les autres agents. Les agents mobiles ont la propriété de transférer le code entre les domaines de la protection et de relancer l'exécution de code. le système doit surveiller l'exécution du code pour s'assurer que les droits d'accès sont toujours mis en vigueur

2.8 La migration des agents

La migration est le processus de déplacement d'un agent d'un site à un autre, tout en transportant les éléments nécessaires constitue l'agent afin d'accomplir leur tâche. En distingue deux types de migration [39] :

2.8.1 La migration ciblée

La migration ciblée c'est le cas d'une rencontre volontaire où les agents souhaitent coopérer avec un autre agent parfaitement identifié, mais dont la localisation est incertaine. Sous ces hypothèses, les agents s'informent de l'état de l'environnement et choisissent la prochaine destination correspondant au site permettant de se rapprocher le plus possible d'agent cible.

2.8.2 La migration libre

La migration libre c'est le cas des rencontres involontaires où les agents se retrouvent sur un site quelconque et souhaitent coopérer avec les agents présents. Dans ce cas, les agents se déplacent au hasard vis-à-vis des autres agents, une fois ils arrivés sur un site ils informent les agents et engagent les coopérations possibles.

Il est à noter que différentes études ont été menées dans le cadre de la migration au hasard afin de l'appliquer dans la gestion de réseau, mais la migration ciblée n'a pas encore été examinée en profondeur dans les environnements dynamiques [39].

2.9 Les Interactions entre agents

Comme les systèmes multi-agents sont constitués d'une société d'agent, l'interaction entre ces agents est essentielle pour la mise en œuvre de l'organisation. '' Une interaction est la mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques… [31].

Une coordination peut être nécessaire pour améliorer le fonctionnement global du système. La notion de coordination entre agents peut être définir comme la description des différentes étapes respectées par les intervenants lors d'une coopération afin d'en garantir le déroulement intégral et ininterrompu [39].

La sollicité de réaliser des tâches de coordination et remarquable lorsque plusieurs agents travaillent sur le même lieu, utilisent les mêmes ressources, ou résolvent des sous problèmes qui ne sont pas complètement indépendants (conception d'un objet complexe par exemple).

La communication entre les agents se faire de deux manières, soit par l'intermédiaire des actions qu'ils font sur l'environnement et qui sont visibles pour les autres agents, soit par l'intermédiaire de messages qu'ils s'envoient les uns aux autres.

Un agent ne peut pas agir directement sur les représentations internes d'un autre agent : quand il envoie un message, l'agent qui le reçoit va l'interpréter et y répondre selon son propre protocole.

2.9.1 La coopération entre agents

Les buts nécessitant la coopération sont des buts sociaux. La coopération est nécessaire quand un agent ne peut pas atteindre ses buts sans l'aide des autres agents .Un agent peut avoir besoin d'un autre agent parce que cet agent a des compétences qu'il n'a pas, ou parce qu'il faut être plusieurs pour réaliser la tâche (une grande tâche qui nécessite un petit temps de réponse).

Il existe différentes méthodes de coopération. Les concepteurs des applications sur des environnements dynamiques doivent être choisir quels sont les types de coopération les mieux adaptés.

2.9.1.1 Coopération directe

C’est une interaction qui autorise uniquement les communications locales et directes entre deux agents, c'est-à-dire deux entités d’un même système souhaitant communiquer.

La coopération directe permet d’améliorer le comportement général des applications, plus que les agents sont nombreux, plus les coopérations seront fréquentes et plus l’amélioration de l’ensemble de l’application est importante. Lorsque les agents utilisent cette coopération directe, ils tirent bénéfice du travail déjà réaliser par leurs homologues et peuvent alors affiner leurs tâches futures,

En fait, cette coopération directe se pose comme l’élément de base de toute phase de dialogue entre agents et servira à proposer tout un ensemble de coopérations plus complexes, mais qui respecte les critères de localisation et d’asynchronisme imposés par la coopération directe.

2.9.1.2 Coopération indirecte

La coopération indirecte est l’utilisation des agents intermédiaires afin de réaliser une communication indirecte à distance entre deux agents. En fait, il s’agit d’une suite de coopérations directes successives avec les agents intermédiaires afin de mettre en relation les participants d’une coopération indirecte.

L’une des applications les plus représentatives de l’utilité de cette coopération indirecte est la mise en place d’algorithmes basés sur la notion de rumeurs ou encore appelés algorithmes des fourmis. Dans ces algorithmes, un ensemble d’agents se déplace en collectant des informations, chacun d’eux traite les données récupérées et laisse sur les sites visités ce qu’il estime important pour les autres agents. Lorsque ceux-ci récupèrent les éléments déposés, ils les traitent à leur tour en fonction de ce qu’ils auront appris durant leurs propres migrations.
Si l’information se confirme, ils accroissent sa véracité pour le reste des agents ou la diminuent dans le cas contraire.

2.9.2 La composition

Une caractéristique notable des agents mobiles est leur capacité d’adaptation au contexte changeant. Lors de ses déplacements, un agent collecte des informations qui lui permettent de s’adapter aux contraintes liées à l’accomplissement de sa tâche.

Pour adapter son comportement, un agent peut récupérer un élément particulier qui lui permettra d'accomplir sa tâche. Cet élément peut être un simple objet ou même un agent complet. C'est ce dernier mécanisme est appelé composition.

La composition peut se réaliser lorsqu'un agent absorbe complètement un autre agent ou en utilisant des mécanismes de hiérarchisation (une arborescence) qui permettent de raisonner sur un groupe d'agents. Pour la hiérarchisation, on détermine l'agent père (origine) de la hiérarchie et on obtient la composition rattachant les agents au père.

Lorsque l'agent père se déplace, c'est toute la hiérarchie qui migre avec lui et on obtient ainsi le comportement classique d'un agent composé. L'intérêt de cette méthode repose sur la facilité d'échange d'agents entre différentes hiérarchies facilitant d'autant les actions de composition/décomposition. De plus, le découplage entre l'agent père et ceux composés garantit une meilleure tolérance aux fautes, car la disparition du père n'entraîne pas automatiquement celle du reste de la hiérarchie.

2.9.3 La délégation

La délégation c'est un mécanisme qui donne la possibilité à un agent de créer des sous agents [39]. Un agent peut identifier lors de son exécution plusieurs sous tâches indépendantes qu'il pourrait traiter en parallèle. Dans ce cas, l'agent crée des sous agents auxquels il attribue les sous-tâches en question puis, optionnellement à la fin de leur exécution, il récupère les résultats obtenus.

Un exemple d'application de la délégation se porte sur les calculs diffusants souvent utilisés pour implanter des applications Internet telles que la recherche d'informations ou le commerce électronique. Un calcul diffusant correspond à toutes les applications composées d'un processus initiateur effectuant un calcul local puis lançant l'exécution d'un ensemble d'autres processus possédant le même comportement. Ce type de comportement peut facilement s'exprimer grâce aux agents mobiles à travers une succession de duplications et de migrations.

Un premier agent mobile commence l'enquête en se clonant et migrant vers des sites de proximité, chaque clone poursuit à son tour le calcul en adoptant le même comportement.

2.10 Les différents types d'agents

Les agents mobiles sont classifiés en deux grands types d'agents [39] : les légers et les lourds. Les légers correspondent à des agents dont la tâche principale ne nécessite pas d'engager de longues phases de traitements locaux. Au contraire, ils stationnent très peu de temps sur le même site en privilégiant les déplacements fréquents et rapides. En opposition, les agents lourds vont nécessiter des temps de visite sur les sites beaucoup plus conséquents à cause de longues phases de traitements locaux, ils vont dès lors se déplacer peu souvent et relativement lentement.

2.10.1 Les agents légers

Les agents légers exécutent une tâche qui effectue de courtes phases de calcul local et ils vont donc migrer fréquemment et rapidement. Ils sont nommés « légers », car il s'agit d'agents de petite taille dans le sens où leur code exécutable est réduit le plus possible et les informations qu'ils véhiculent sont peu volumineuses. Cette petite taille va leur donner la faculté d'un déplacement très bref dû à un temps de transmission très court grâce à leur faible coût en bande passante.

Le domaine d'application des agents légers s'inscrit principalement dans la réalisation des services de base du système qui demande une forte indépendance par rapport aux déplacements des sites et une grande réactivité aux changements de l'environnement. Pour ce faire, ces agents seront nombreux et vont généralement utiliser la migration libre en multipliant de brèves phases de coopération, directe ou indirecte, avec des partenaires prise au hasard.

2.10.2 Les agents lourds

À l'opposé des légers, les agents lourds réalisent une tâche imposant de longues phases de traitements locaux et en conséquence ils effectuent de rares déplacements qui sont relativement lents. Ces agents sont dits « lourds », car la taille du code exécutable ainsi que celle des données transportées seront beaucoup plus volumineuses que celles des agents légers. Ces agents lourds sont généralement utilisés lors de la conception d'applications dites « métier » qui nécessitent un savoir-faire particulier bien plus complexe que celui des agents légers. Dans ce type d'applications, le développeur connaît généralement les autres services métiers qu'il souhaite utiliser et va donc choisir de mettre en place la migration ciblée pour établir les rencontres

nécessaires à la mise en oeuvre de la coopération. C'est là où ce situe le principal problème d'un agent lourd, car les longues phases de traitements vont le rendre beaucoup moins mobile que les sites supports et vont nécessiter des adaptations à chaque fin de calcul pour appréhender les modifications de l'environnement et pouvoir effectuer la prochaine rencontre. De plus, vu le coût important d'une migration pour un agent lourd, il faudra minimiser le nombre des déplacements qui lui permet d'atteindre son partenaire et par conséquent il ne pourra pas naviguer au hasard de site en site, mais il devra orienter ses mouvements afin d'optimiser le coût général de cette phase de recherche.

2.11 Modèles de communication pour les agents mobiles

En général, les techniques de communication des agents mobiles sont classées en deux types de modèle de communication, d'après [40] :

2.11.1 Passage des messages

Ce premier type de communication permet aux agents d'envoyer des messages entre eux. C'est l'une des formes de communication directe ou l'agent qui envoie un message doit connu le nom et l'emplacement de l'agent qui doit recevoir ce message, cela nécessite que soit l'agent connu à l'avance les autre agents, soit par la mise en œuvre d'un service qui permet aux agents d'obtenir des informations sur les autres agents. D'une autre part l'agent récepteur doit comprendre le message reçu et décider quelle action à faire.

La forme la plus simple de passage de message est la communication point à point dont un seul agent émetteur envoie un message à un seul récepteur.
La deuxième forme de passage de message est la communication multi point (multi-cast ou broadcast), ce type est utilisé lorsque plusieurs agents travail ensemble pour résoudre un problème.

2.11.2 Espace d'information

Ce modèle de communication fournit à tous les agents un seul espace pour l'échange d'information. C'est une forme indirecte de communication. Tout simplement un agent peut

écrire une donnée dans espace commun et les autres agents peut le lire. C'est à la responsabilité des agents d'allons voir s'il y a une modification ou une nouvelle donnée dans l'espace d'information. Toutes les approches d'espace d'information découplé la communication entre agents lorsqu'ils ne sont pas besoin de synchroniser leur communication.
Cependant, dans les systèmes qui utilisent le *blackboard* pour la communication, chaque information doit possède un identificateur spécifier par l'agent rédacteur et doit être connu par les agents lecteurs. D'autres systèmes utilisent une autre forme de communication « tuple » c'est une extension de blackboard on ajoutant des mécanismes associatifs à l'espace d'information partagé.

2.12 Domaines d'application

Dans cette partie, nous présentons quelques domaines d'applications envisageables pour les agents mobiles.

- **La mise à jour des ressources réparties**

Les agents mobiles peuvent être utilisés pour l'exploration et la mise à jour des ressources réparties au sein d'un réseau. Ces ressources peuvent être applicatives (services) et/ou matérielles (routeur-serveur). Ces mises à jour consistent à remplacer, ou à ajouter, des fonctionnalités à un ensemble d'éléments. Par exemple, actualiser un routeur avec la nouvelle version d'un protocole.

L'utilisation des agents mobiles permet de décrire simplement l'exploration d'un réseau et permet d'envisager une mise à jour totalement découplée d'un service central en déléguant une partie de l'administration aux agents mobiles. Dans ce cas, un agent possède la mise à jour à appliquer et va se déplacer de site en site, que se soit un ordinateur hôte ou un routeur, et va actualiser tous les éléments trouvés. En se déplaçant, les agents peuvent plus facilement accéder aux éléments appartenant aux infrastructures décentralisées et surtout peuvent, grâce à leur traitement local, appliquer la mise à jour sans craindre d'être interrompus.

.

- **Découverte de contexte**

La découverte de contexte désigne la récupération des informations permettant de caractériser la situation des éléments présents dans l'environnement. La découverte de contexte

se définit alors comme l'utilisation du contexte afin de donner des informations significatives aux applications et/ou des services aux utilisateurs. Dans la découverte de contexte, l'exploration de l'environnement est facilitée par la mobilité des agents qui peuvent se charger de trouver toutes les informations utiles présentes sur les sites voisins. De plus, leur capacité d'adaptation leur permet d'interpréter le contexte en fonction de besoins spécifiques et/ou de réagir en fonction d'événements perçus dans le contexte : par exemple, récupérer certains services bien précis ou capter le déplacement d'une unité mobile.

- **Grille de calcul**

Le « grid computing» consiste à regrouper dynamiquement au sein d'un même réseau virtuel, tout un ensemble de machines, de ressources et d'utilisateurs. Ceux-ci peuvent être hétérogènes et dispersés à l'échelle planétaire. L'objectif des grilles de calcul est de permettre à des organisations dispersées (entreprises, universités, etc.) de partager des applications (services), des données et des ressources (processus, disque). Le but est de mettre en commun les capacités de chaque entité.

Avec les grilles de calcul, les agents mobiles trouvent un champ d'application naturel. En effet, les caractéristiques de déplacement, d'adaptation, de coopération vont permettre de tirer parti de ces environnements. On peut voir une grille comme un marché ouvert où des marchands (organisations) proposent un ensemble de produits (services) aux clients (utilisateurs) présents dans l'environnement. Les clients explorent le marché pour trouver les meilleurs produits en fonction de leurs besoins. Les agents mobiles vont pouvoir adopter ce comportement naturel de négociation. Si un client cherche à utiliser un service en fonction d'un certain nombre de critères, il crée un agent qui va aller de serveur en serveur afin de trouver le service correspondant le mieux aux critères définis par l'utilisateur. Une fois qu'il l'a trouvé, il réalise le service visé et ramène les résultats à l'utilisateur.

Le deuxième intérêt des agents mobiles est leur capacité à représenter un utilisateur déconnecté. Avec un utilisateur nomade, se connectant de manière intermittente à la grille, l'utilisation des agents mobiles permet de déléguer la réalisation de la tâche que l'utilisateur souhaite exécuter. L'autonomie de l'agent permet à l'utilisateur de sortir de la grille durant le temps de la réalisation de la tâche. Lorsque l'utilisateur se connectera et que l'agent aura achevé

sa tâche, il pourra récupérer les résultats obtenus par l'agent. L'utilisateur peut, de plus, avoir changé complètement de lieu entre-temps.

- **Application orientée client nomade**

Avec l'augmentation de l'utilisation des unités mobiles de petite taille, les clients nomades sont capables de se déplacer facilement tout en exécutant des applications complexes et en ayant la capacité de rester connectés à un réseau sans fil.
Avec ces unités mobiles, les utilisateurs nomades vont vouloir accéder à leurs applications et données favorites quels que soient les environnements les entourant. De plus, lors de ces déplacements, un utilisateur va se retrouver au sein d'environnements aux caractéristiques diverses et variées, que ce soit au niveau de la performance du réseau ou au niveau de la multitude de services. La capacité d'adaptation des agents mobiles va leur permettre de s'accommoder de cette diversité en fonction des besoins des utilisateurs. Par exemple, lorsqu'un utilisateur nomade souhaite afficher ses photos, l'agent pourra pré-traiter une image haute définition pour l'adapter à la résolution de l'écran de son unité mobile.

- **Reconfiguration dynamique et administration**

La reconfiguration dynamique des applications à distance peut se présenter sous différents aspects : la modification de l'architecture de l'application par l'ajout ou la suppression de certains de ses composants logiciels, la modification de la mise en œuvre des composants ou le changement de la distance géographique de l'application. Cette modification consiste à déplacer des composants logiciels de l'application vers de nouveaux sites, sous forme d'agents mobiles.

L'administration à distance du système nécessite parfois la réinitialisation des machines sans interrompre certains services (ou des applications) en cours d'exécution sur ces machines. De tels services peuvent alors être transférés vers d'autres machines. Par exemple, *Jumping Beans*^{MD} de *Ad Astra Engineering* permet à un administrateur e système de gérer son réseau à distance. Ce produit permet de construire des agents mobiles pour l'installation et la maintenance des logiciels à distance et à la demande de l'administrateur.

- **Communications hétérogènes et asynchrones**

Bien que, les réseaux de communication offrent beaucoup d'avantages potentiels, cependant, ils peuvent soulever autant de problèmes quand des systèmes incompatibles (format de données, système d'exploitation, architecture interne) doivent être reliés. Les agents mobiles peuvent résoudre ce genre de problèmes.

Certaines applications exigent plusieurs interactions client/serveur qui nécessitent une connexion réseau assurée pendant une longue période, ou dans plusieurs transmissions séparées. Si un agent mobile est utilisé, le client n'a pas besoin de maintenir une connexion quand ses agents accèdent et traitent des informations sur le serveur. Ce qui permet d'augmenter les communications asynchrones entre le client et le serveur.

- **La conception et le développement des applications d'Internet**

La conception et le développement des applications d'Internet tel que la recherche d'information exigent un accès dynamique et peut être mobile aux ressources de ces applications. Par conséquent, c'est une approche basée sur les agents mobiles autonomes. Elle introduit de nouveaux problèmes et de nouvelles exigences sur les approches traditionnelles de développement des programmes distribués. Cette approche permet alors de concevoir et de développer des applications en terme d'entités autonomes, c.-à-d. des agents s'exécutant de façon proactive, réagir à des changements de son environnement.

2.13 Plateforme de développement

On distingue trois approches pour concevoir et implémenter une plateforme d'agents mobiles. La première consiste à recourir à un langage de programmation qui comprend des instructions pour les agents mobiles. Compaq^{MD} a essayé sans succès cette approche avec le projet *Oblic*.

La deuxième approche consiste à implanter le système d'agents mobiles comme des extensions du système d'exploitation. Enfin, la dernière approche construit la plate-forme comme une application spécialisée qui tourne au-dessus d'un système d'exploitation.

Plusieurs systèmes, commerciaux et autres, supportant la mobilité sont actuellement disponibles. Ils sont basés généralement sur la dernière approche et implémenté comme une collection des librairies java.

Dans cette section, nous en présentons brièvement les plates formes les plus répandues et les plus importantes.

- **Agent TC1**

Un des premiers systèmes supportant les agents mobiles a été développé par Robert Gray, David Kotz et d'autre [41]. À l'origine, le système n'avait pas de nom et utilisait TCP/IP, le courrier électronique et la commande UNIX « rsh » comme mécanismes de transfert. Plus tard, la même équipe a introduit « Agent Tcl », un système utilisant Tcl comme langage principal de programmation des agents.

Toutefois, à la base, le système supporte plusieurs langages, à savoir Tcl, Java et Scheme, et récemment le système a été rebaptisé « D'Agents ». Notons qu'Agent TCL est un des seuls systèmes actuels n'étant pas basés directement sur Java.

- **Aglets**

Le « Aglet Workbench » [42] d'IBM Japon est sans doute un des systèmes les plus connus. Il possède une architecture similaire à celle des applets Java et un protocole de transfert entre hôtes (*Aglet Transfer Protocol*). Le système est facile à charger du Web et à installer sur tout système supportant Java.

- **Concordia**

Concordia [43] est également un système basé sur Java, mais développé par le Horizon Systems Laboratory de Mitsubishi. C'est un système récent, orienté vers des applications au niveau des entreprises. Par conséquent, la sécurité du système est exceptionnellement soignée.

- **Mole**

Ce système a été développé dans l'Institute of Parallel and Distributed High-Performance Systems à l'Université de Stuttgart en Allemagne [44]. Elle implémente une mobilité faible, c'est-à-dire seuls les données et l'état de l'agent sont transférés.

- **Telescript**

Telescript [34], développé par General Magic, peut être considéré comme la référence dans le domaine. Il est parlant que son inventeur, Jim White, tient un brevet sur la notion même d'agents mobiles. Telescript était un des seuls systèmes supportant le transfert de l'état d'exécution des agents. Ce système est conçu pour une application industrielle et exigeait des ressources importantes.

- **Odyssey**

Le successeur de Telescript, développé également par General Magic, s'appelle Odyssey. Différent du premier, ce dernier est basé sur Java. Il est largement inspiré par Telescript, mais n'à pas exactement la même fonctionnalité. Par exemple, Odyssey ne supporte pas le transfert de l'état d'exécution. Le système n'a toujours pas dépassé la version bêta, et ne joue visiblement pas le rôle important que jouait Telescript dans la stratégie de General Magic.

- **Sumatra**

Sumatra [45] a été développée pour mesurer la performance des agents dans la gestion des réseaux. Il implante une application *Komodo* qui surveille l'état (les délais) du réseau. L'application test est *adaptalk,* une application de « chat » internet. Les textes entrés par les usagers sont acheminés aux destinataires par des agents mobiles déterminent dynamiquement leur trajectoire en tenant compte des délais de réseau.

- **Voyager**

Voyager [46] est un *Object Request Broker* (ORB) proposé par ObjectSpace. Il offre des services pour agents, il s'agit d'un système « branché » incorporant plusieurs normes industrielles

populaires à l'heure actuelle, notamment ORB, CORBA, DCOM, RMI et support les JavaBeans. Voyager permet d'envoyer des messages asynchrones (sans réponse) et synchrones (appels de fonctions à distance) et permet aussi l'envoi des messages sélectifs (multicast).

2.14 Conclusion

Dans ce chapitre nous avons introduit les agents mobiles. La nature distribuée des systèmes à base d'agents mobiles lui permet de s'adapter aux environnements où le système n'offre qu'une gestion locale du contexte. Par leur autonomie, leur adaptation et leur mobilité, les agents mobiles sont compatibles avec la mobilité des sites.

Dans le plus part des applications, les systèmes à agents mobiles semblent être plus efficaces que les systèmes classiques à agents stationnaires, car dans un système informatique, les interactions intra-site sont moins coûteuses que les interactions inter-sites.

Plusieurs plateformes de développement des agents mobiles sont utilisées dans le monde industriel. La plus part parmi ces plateformes sont basées sur le java pour améliorer et garantir la portabilité des applications.

Dans le chapitre suivant nous allons voir quelques approches a base d'agents mobiles pour le routage dans les réseaux ad hoc.

Chapitre III

Applications des agents mobiles pour le routage dans un réseau ad hoc

3.1 Introduction

Les agents mobiles peuvent être utilisés pour l'exploration et la mise à jour des ressources réparties au sein d'un réseau .la nature distribuée du problème de routage conduit plusieurs chercheurs a proposés des solutions à base d'agent pour ce problème.

Dans ce contexte, de nombreux travaux ont été élaborés afin d'introduire la technologie multi-agent et en particulier la technologie d'agents mobiles et les concepts liés à cette dernière.

Ce chapitre est dans l'objectif de discuter quelques applications des agents mobiles dans le routage pour un réseau mobile ad hoc.

3.2 MAGNET

Dans leur article «MAGNET : ad hoc network system based on mobile agents» le trinôme N. Kawaguchi, K. Toyama, Y. Inagaki [47] propose de faire implémenter tout le réseau on se basant sur les agents mobiles, l'idée principale est d'utiliser l'agent mobile comme unité principale de tout le système. Chaque paquet circule dans le réseau est un ensemble des agents mobiles de plus chaque routeur est aussi un agent mobile, le lien entre deux hôtes est dirigé et

maintenu par l'agent *link manager*. Le routage se fait par l'agent mobile lui-même sous la communication avec un agent stationnaire de routage.
Un seul agent peut jouer le rôle d'un paquet en cour de transmission ou d'une application utilisateur et parfois les deux en même temps (par exemple un agent contenir un algorithme de routage et d'autre part fourni une interface utilisateur).
La structure du système a les caractéristiques suivantes : la reproduction d'agent et la hiérarchie d'agent.

3.2.1 La reproduction d'agent

Afin d'éliminer l'effort de développement de protocole de communication entre les agents (inter-agent) et entre les hôtes (inter-host) le groupe développant MAGNET autorise seulement les communications entre les agents qui ont le même identificateur ID, ces agents sont appelés *replica agents*.

Au moment de création des agents, chaque agent possède un identificateur unique, la reproduction donne la possibilité de trouver des agents qui en le même ID, les mêmes fonctionnalités et les mêmes autorités sur des différents hôtes, cette opération peut être faite avant la migration de l'agent vers un autre hôte.

Lorsqu'un agent remarquer l'arriver d'un replica-agent (avec un ID identique) sur le même hôte, la synchronisation des données peut prendre lieu.

3.2.2 Hiérarchie d'agent

Dans MAGNET chaque agent peut contenir des agents à l'intérieur de lui-même ce qui rende la représentation hiérarchique la plus adéquate. Un agent fils peut enregistrer, replacer ou disposer par son parent, un agent peut replacer ou disposer par ces agents ascendants, quand un agent se déplace d'un hôte à un autre ses agents descendants se déplacent vers le même hôte avec la même hiérarchie.

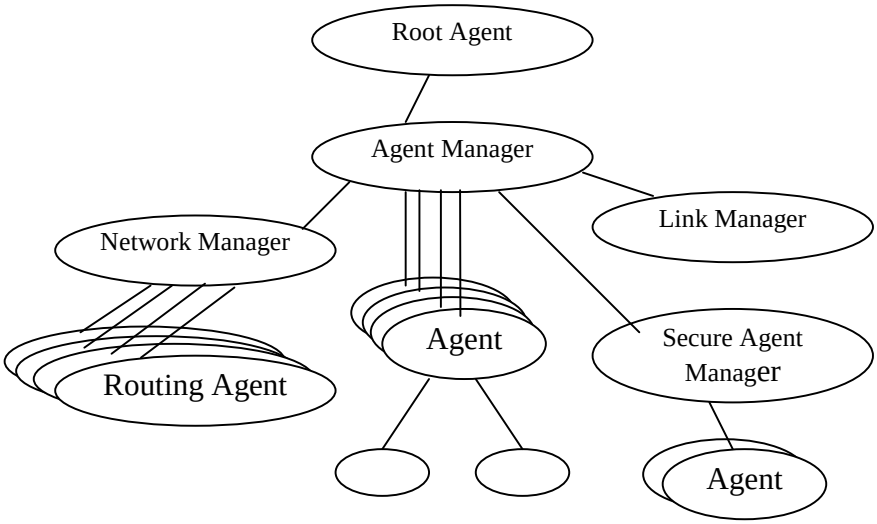

Figure 3.1 : Hiérarchie d'agents

La Figure 3.1 montre la configuration canonique de hiérarchie agent ou Root Agent est le seul agent qui ne se déplace pas, Agent manager est le responsable de registration d'agent et la communication entre agents et avec d'autres agents manager, il contient d'autre sous agents tels que Network manager, Link manager et même d'autre agent manager.

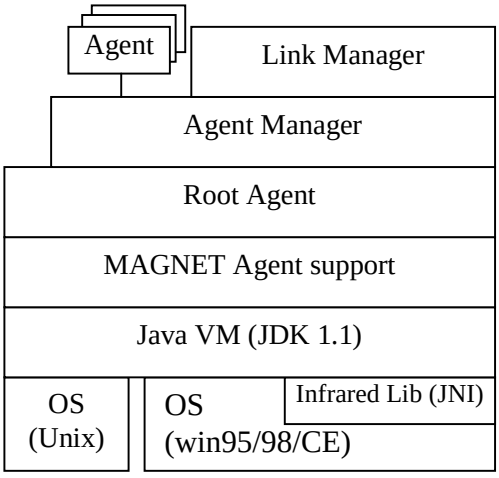

Figure 3.2 : Configuration de MAGNET

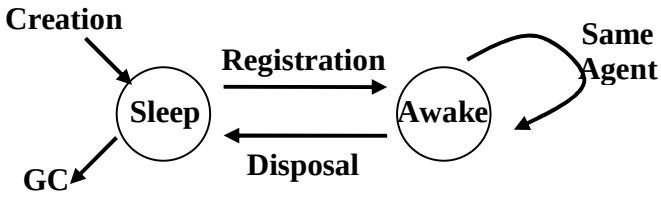

Figure 3.3 : Cycle de vie d'un agent

3.2.3 Adaptation de réseaux

Afin de faire face au changement dynamique de la topologie du réseau, l'utilisation des agents mobile et la hiérarchie d'agent permettent une extension dynamique de protocole et de service fourni par MAGNET.

L'agent Link Manager jouer le rôle principal dans la construction de réseau, il prend en charge la découvert des hôtes adjacents et de l'établissement de communication avec eux au niveau de la couche liaison. Il est aussi le responsable d'un ensemble des agents nommés Link Monitor,ces agents sont notifiés par l'agent Link Manager lorsque la détection d'un événement apparut sur le réseau(changement de topologie à l'arriver d'un hôte ou disparition d'un autre). L'agent Link Monitor est autonome à faire le reste de traitement de l'événement.

La gestion de la couche réseaux est à la charge de l'agent *Network Manager* et *Routing agent*, lorsque *l'agent Manager* fait appel l'agent *Network Manager* pour envoyer un agent à une destination, ce dernier demander à l'agent *routing agent* de délivrer l'agent à sa destination. Chaque agent *routing agent* possède des stratégies de sélection le chemin de routage.

3.2.4 Avantages et Inconvénients

- **Avantages**
  - Il ne se base sur aucun autre réseau, les agents mobiles forme eux-mêmes le réseau et fourni le service de routage.
  - La reproduction des agents peut être vue comme une méthode de communication sécurisée parce qu'elle peut éviter le truquage d'un ID par le partage des clés par exemple.
  - Un hôte mobile non préparer vient d'intégrer dans le réseau n'a pas besoin des protocoles déjà installés, seulement la migration d'agent de protocole permet de mise en oeuvre.
  - Possibilité d'appliquer cette méthode avec divers protocole.

- **Inconvénients**

Malgré la bonne structuration du système et la simplicité de construction de réseaux, la surcharge du au nombre d'agents augmenter croisement avec le nombre de nœuds reste tout jour un problème.

3.3 Approche à base d'agent pour le routage à vecteur de distance

3.3.1 Description

C'est une approche à base d'agent pour le routage à vecteur de distance. Les protocoles de routage à vecteur de distance sont des algorithmes basés sur les tables de routage, l'échange d'information (la métrique de distance) entre les nœuds pour la mise à jours des tables est nécessaire pour découvrir les routes [48].

Pendant leur déplacement d'un nœud à un autre, l'agent découvre les routes et transporte l'information de mise à jour. La description formelle de l'agent A est la suivante :

A (i, x, y, Rx, γ) ou

i : identificateur d'agent,

x et y représente la migration de l'agent de n_x vers n_y deux nœuds adjacents,

Rx : sous-ensemble de r_x table de routage de n_x,

γ : stratégie de migration.

Chaque entrée dans la table de routage r_x, e_{xi} (métrique de meilleur chemin entre n_x est n_i) correspond à un vecteur V_{xi} des booléens de taille égale nombre des nœuds voisins à n_x représente s'il y a une modification de la valeur de e_{xi} (figure 3.4).

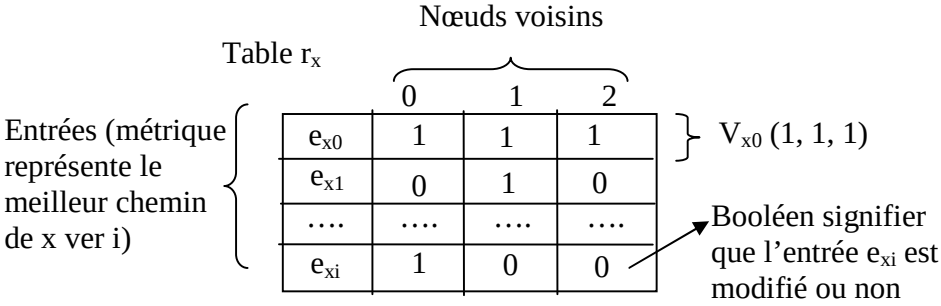

Figure 3.4 : Table de routage de nœud x

Lorsqu'une migration d'un nœud x vers un autre y, l'agent construit un sous-ensemble Rx de $r_{x,}$ il sélectionne les entrées qui ont $V_{xi}[y] = 1$. Par exemple, si l'agent veut migrer de n_x vers le nœud 0, il sélectionne les entrées comme illustrées la figure suivante :

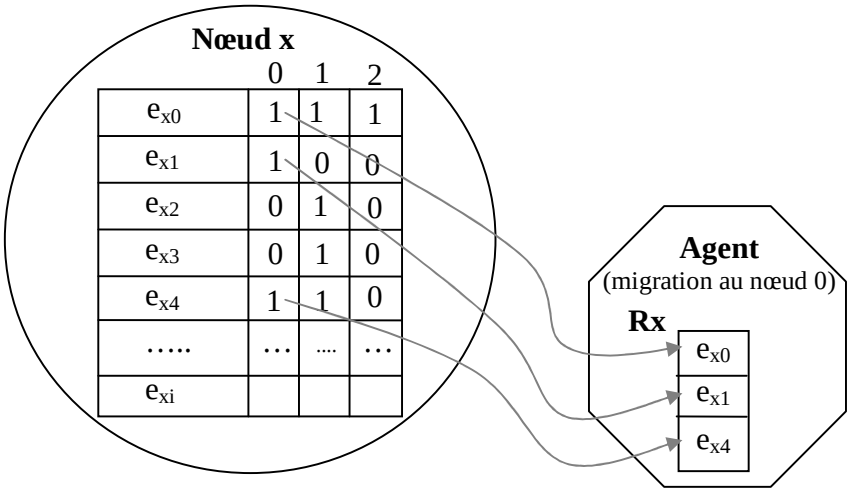

Figure 3.5 : Sélection des entrées par un agent.

Dans ADVR les agents commencent d'une manière arbitraire, au démarrage toutes les valeurs Vxi sont mises à 1 $\forall$ e_{xi} . Après la sélection d'une entrée e_{xi} a transporté vers un nœud voisin, l'agent copie e_{xi} dans son segment de donnée et met le booléen correspond dans la table de routage à 0 c'est-à-dire Vxi[y] :=0. Si un agent modifie une entrée, il met toutes les Vxi correspondantes à 1.

L'affectation d'un temps ΔT permet d'assurer la robustesse et la tolérance aux pannes, lorsque ΔT est expiré les valeurs de Vxi sont mises à 1 afin de provoquer la mise à jour.
Lorsqu'un agent A (i, x, y, Rx, γ) arrive à n_y il mettre à jour la table de routage selon l'équation :

D(y, j)=min(D(y, j),[d(y,x)+D(x,j)]) $\forall$ ***n_j in Rx*** ou

D(y, j) est une entrée dans la table de n_y représente le meilleur chemin n_y vers n_j

D(x, j) est une entrée dans la table Rx représente le meilleur chemin n_x vers n_j

d(y, x) le cout de passage de lien entre n_y vers n_x

Après la mise à jour, l'agent sélectionner la table Ry est migre vers un nœud adjacent en utilisant la stratégie de migration γ. Alors, à chaque nœud l'agent prend une décision concernant les données de routage à transmettre au noeud suivant.

La stratégie de migration dans ADVR est une combinaison de deux stratégies, la première est *la Stigmergie* inspirer biologiquement de la colonie des insectes. C'est un mécanisme utiliser par des individus naïfs (exp : les fourmis) pour communiquer avec les autres par un changement local dans l'environnement. La deuxième est *la recherche en profondeur d'abord* basée sur l'échange d'historique de migration entre les agents.

De mêmes façons qu'une fourmi dépose de phéromone (produit chimique) sur son chemin pour attirer l'attention des autres fourmis, un agent migrer d'un nœud n_x vers un nœud n_y dépose de la phéromone (valeur) sur le lien xy, un autre agent migrer depuis n_x choisi le lien possède la plus faible valeur de phéromone, c'est-à-dire l'agent va migrer vers la région la moins visité dans le réseau. D'autre part, l'agent ne transporte pas l'historique comme une partie de son segment de donnée, ils déposent seulement de la phéromone pour indiquer leurs présences sur le noeud.

3.3.2 Avantages et Inconvénients

- **Avantage**

Le déplacement d'agent permet en même temps de découvrir une route est mettre à jour les table au lieu seulement d'une propagation d'un message de mise à jour dans les approches traditionnelles.

La sélection d'un sous-ensemble de table de routage par l'agent réduit la surcharge de réseau se qui implique une utilisation moins de ressources et permet aussi de ne pas retarder les informations de routage importantes par des informations inutiles.

- **Inconvénient**

Le choix de la taille de population des agents est un problème dans ADVR, si la population est très petite ça risque de dégrader les performances de contrôler le comportement dynamique de réseau et si elle est très grande ça risque de conduise ver un manque de ressource. Une population optimale dépend de disponibilité des ressources qui change d'une manière

dynamique, alors il est nécessaire de faire plus de contrôle décentralisé entre les agents ce qui engendre un coût supplémentaire.

La migration des agents vers les régions les moins visitées ne dépend pas de dynamisme de réseaux, cette distribution des agents peut n'avoir un grand effet lorsque la région ciblée est moins dynamique ou possède un petit nombre des nœuds par rapport aux autres régions.

3.4 Approche à base d'agent pour le routage par multidiffusion

3.4.1 Description

La communication par multidiffusion (multicast) est un routage orienté groupe utiliser pour les application telle que le vidéo conférence, travail collaboratif… etc. cette technologie permet de conserver la bande passante du réseau. Elle réduit le volume de trafic en permettant à un hôte d'envoyer un seul paquet à un groupe d'hôtes désigné. Chaque groupe de multidiffusion est représenté par une seule adresse de destination multidiffusion.

Dans leurs article "Multicast routing in mobile ad hoc networks by using a multiagent system" S.S. Manvi , M.S. Kakkasageri [49] proposent ABMRS (Agent Based Multicast Routing Scheme). Cette approche utilise une collection d'agents statiques et mobiles: *Route manager* (agent staique), *Network initiation* (agent mobile), *Network management* (agent staique), *Multicast initiation* (agent mobile) et *Multicast management*(agent staique).

Le système fonctionne selon les étapes suivantes:

1. les deux agents *route manager* (statique) et *Network initiation* (mobile) coopérer afin d'identifier les noeuds fiables ; un nœud fiable est un nœud où le facteur de fiabilité a une valeur significative. Ce facteur est calculé à partir d'autre facteur (énergie, bande passante, mémoire disponible, nombre de mouvements)
2. les agents mobiles *Network initiation* connecter avec les noeuds fiables à travers les noeuds intermédiaires. Les nœuds intermédiaires sont utilisés pour connecter les nœuds fiables lorsqu'ils chacun d'eux n'est pas dans la portée des autres.
3. On utilisant les nœuds fiables et intermédiaires les agents *Route manage*r (statique) et *Network initiation* (mobile) construire ce qui similaire à une colonne de vertébrale pour la multidiffusion.

4. Les agents mobiles *Multicast initiation* sont utilisés pour joindre les membres de groupe de multidiffusion à la colonne de vertébrale de multidiffusion.
5. La gestion des membres de groupe de multidiffusion est assumée par les agents *Multicast management*(statiques) et la gestion de la colonne de vertébrale est à la charge des agents *Network management*(statique).

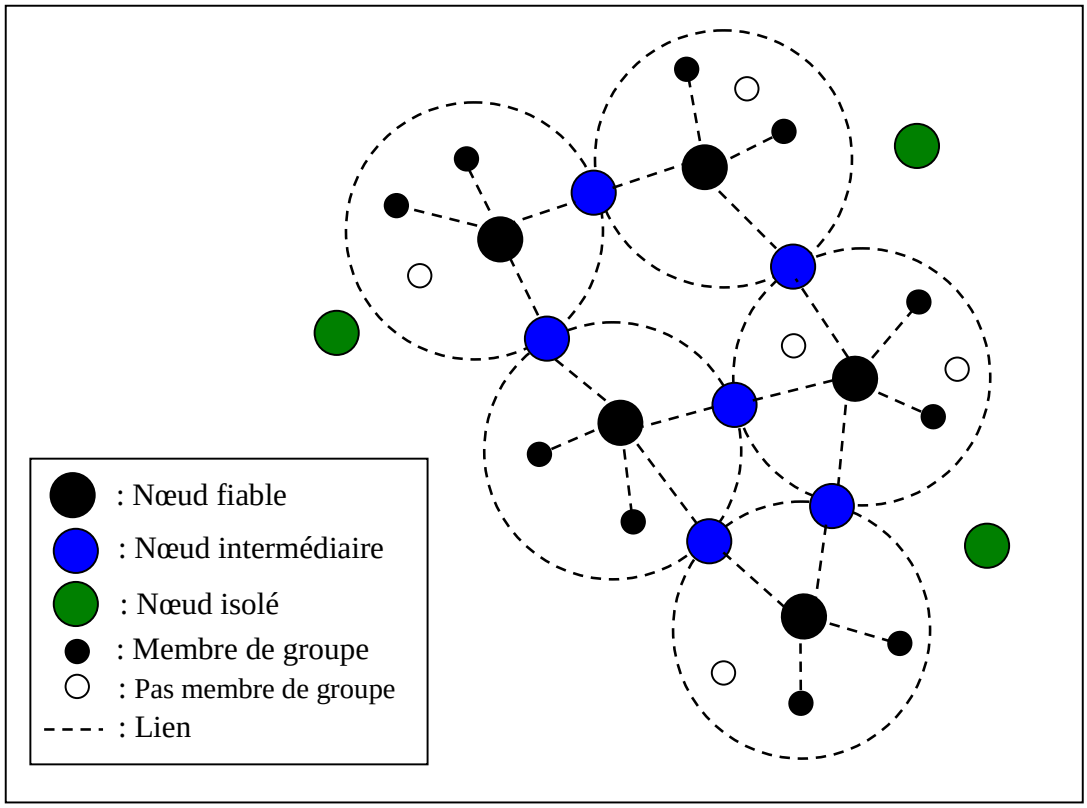

Figure 3.6 : Environnement de réseau.

3.4.2 Agence de routage multidiffusion

Le rôle de cette agence et de prendre des décisions de routage, la collection des informations et prévoir le comportement de réseau…etc. les interactions dans cette agence sont illustrées dans la figure suivante :

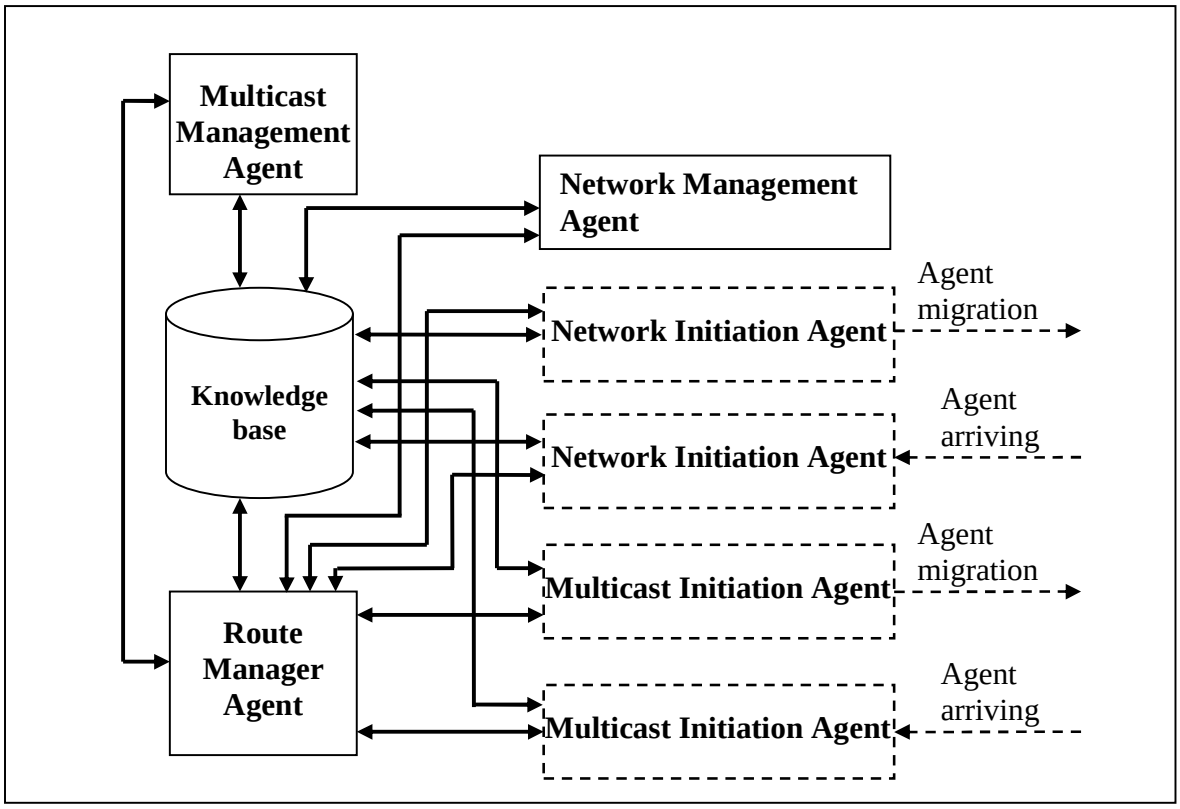

Figure 3.7 : Agence de routage de ABMRS.

L'agence consiste de l'ensemble des agents qui gèrent le réseau et une base de connaissance (KB) utiliser pour les communications entre les agents et pour la mise à jour fonctionne sous le principe de boite noire.

3.4.2.1 La base de connaissance (KB)

Elle contient des informations sur :

- L'état des nœuds (intermédiaire, fiable,fils….), facteur de fiabilité RF, la disponibilité (en terme d'énergie, bonde passante et mémoire utilisés pour le multicast ou non), les mouvements, identificateur de groupe ID et les membres.
- Le voisinage : identificateurs des voisins est leur état et l'état des connectivités (up or down).

- Les tables de routage : identificateur de multicast et le nœud de prochain saut (intermédiaire ou fiable).
- La liste des responsabilités : dans les nœuds fiables compris les identificateurs des noeuds membres de groupe multicast (nœud fils).

3.4.2.2 Les agents

- **Route Manager Agent (RMA)**: c'est un agent statique qui contrôler l'agence et collecter des informations depuis les autres agents, il est responsable de créer les agents et la base de connaissance et coordonner les différentes activités. Il fait appel à l'autre agent (*network initiation* et *multicast initiation*) pour la construction de colonne de vertébrale, table de routage et de mettre à jours périodiquement (les tables et la base de connaissance). Il mesure aussi les paramètres de calcul de facteur de fiabilité des nœuds RF.

- **Network Initiation Agent (NIA)**: c'est un agent mobile générer périodiquement (dépend de la mobilité des nœuds) par RMA pour collecter des informations telles que les RF depuis les nœuds voisins et de les mettre fiable lorsque la valeur de RF est grande. Il déplace aussi entre les nœuds fiables pour découvrir la connectivité entre eux.

- **Network Management Agent (NMA)**: c'est un agent statique réside dans les nœud fiable pour la maintenance des chemin, s'il y a une déconnexion lorsque le mouvement d'un nœud, NMA envoie une requête à RMA pour initialiser NIA pour découvrir le nouveau chemin.

- **Multicast Initiation Agent (MIA)**: c'est un agent mobile responsable de groupe de multicast. Il collecte les informations sur les membres de groupe, fourni un ID, déplacer entre les nœuds faibles pour inviter au groupe de multicast et construire un arbre de multicast qui contient différents types des noeuds.

- **Multicast Management Agent (MMA)**: c'est un agent statique, leur fonction principale est la maintenance de l'arbre de multicast. Il envoyer périodiquement un jeton dans le

groupe, si un nœud fils reçoit périodiquement ce jeton il connu qu'il est connecté aux réseaux sinon MMA lancer une procédure de recouvrement.

3.4.3 Avantages et Inconvénients

- **Avantage**

Permettre de faire un contrôle de mobilité des nœuds sur deux axes le premier au niveau des nœuds fiables forment la colonne de vertébrale et le deuxième au niveau de groupe.

- **Inconvénient**

Approche coûteuse en terme de mémoire (sur chaque nœud une base de connaissance contient un grand volume d'information).

3.5 Agent mobile pour un rouage adaptatif (Anet)

3.5.1 Description

Anet [50] est un algorithme adaptatif distribué à base d'agents mobiles proposés par Dorigo et Di Caro. Elle composer de deux ensemble d'agent F-Ant (Forward Ant) et B-Ant (Backward Ant). Les deux types d'agents ont la même structure, la différence est dans le positionnement dans l'environnement,

La description de l'algorithme est la suivante :

1- dans intervalle régulier, un agent F-Anet est lancé depuis chaque nœud vers une destination aléatoire. À chaque nœud traverser, l'agent mémoriser l'identificateur de nœud et le temps écouler depuis le lancement de l'agent jusqu'à l'arriver au nœud dans une mémoire de stockage S et insérer aussi dans un dictionnaire de structure D transporter avec l'agent.

2- l'agent utilise les informations stockées dans la table de routage pour sélectionner le saut suivant. Si un agent arrive à un nœud déjà visité, il détecte qu'un cycle est produit, ce problème est traité tout en supprimer le nœud de la pile et de libéré le mémoire associé.

3- Lorsque l'agent F-Anet atteint sa destination, il génère un agent B-Anet et transférer à lui toutes ses informations.

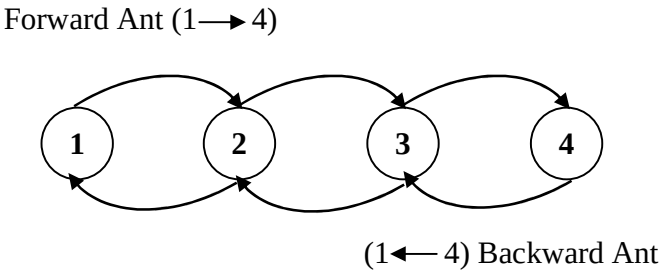

Figure 3.8 : Comportement de F-Ant et B-Ant

4- B-Anet traverse le chemin inverse en utilisant les informations stockées au niveau de la pile. À l'arrivée à un nœud, B-Anet met à jour les structures de données suivantes :

a- La table de routage, pour chaque paire (i, n), la probabilité *Pin* qu'un nœud n soit le nœud suivant si le nœud destination est i.

avec $\sum_{n \in Nk} Pin = 1, i \in [1, N], Nk = \{voi \sin s(k)\}$

la mise à jour se fait par l'incrémentation de la probabilité P_{df} associée avec f si la destination est d et la désincrémentation des probabilités P_{dn} de tous les nœuds n voisins pour la même destination.

b- la liste Tripk (μ_i, σ_i^2) estimer la valeur moyenne arithmétique μ_i et la variance associée σ_i^2. Cette liste représente l'état de réseau de point de vue de nœud k. la mise à jour de la liste se fait avec les valeurs stockées dans le mémoire de stockage S.

3.5.2 Avantages et Inconvénients

- **Avantage**

Anet est un algorithme très simple à implémenter. Le temps calculé pour atteindre chaque nœud par l'agent F-Anet peut donner une bonne indication sur le chemin à suivi.

- **Inconvénient**

Rien ne peut assurer que B-Anet trouve le chemin (suivi par la pile) est toujours maintient sur tout s'il est trop long ; de plus, le temps optimal pour atteindre un nœud est lié à d'autres facteurs tels que le trafic de réseau .

3.6 Conclusion

Dans ce travail nous avons traité le problème de routage dans les réseaux ad hoc en utilisant une approche multi agent. La fonction d'un protocole de routage est de garantir que les données transmises à partir d'un noeud source du réseau arrivent à un noeud destination.

Ce chapitre donne une petite synthèse de quelque approche utilisant les agents mobiles ou une collection des agents mobiles et d'autres statiques pour le routage.

L'objectif de notre travail est de développer un protocole de routage efficace pour les réseaux ad hoc basé sur une approche agent mobile. L'idée est qu'il considère un ensemble d'agents mobiles qui circule au tour de réseaux, collecte d'information et met à jour les tables de routage. Lorsqu'un agent doit acheminer des paquets de données vers un noeud destination, il négocie avec les noeuds voisins, sélectionne le meilleur voisin pour lui transmettre les données. Le critère de sélection des agents voisins est basé sur plusieurs paramètres tels que des connaissances sur d'autres agents, la distance entre les noeuds voisins, etc.

Afin de faire face à la mobilité des nœuds nous proposons quelque point à traités :

1- Pourquoi les agents dans les travaux cités précédemment circulent d'une manière aléatoire, il est préféré de distribués ces agents d'une manière à faire un équilibrage en fonction de nombre est de mouvement des nœuds dans chaque direction ou région de réseau. En d'autres termes, la région possède une forte dynamique doit contenir un nombre plus grand des agents.

2- Il faut penser à une manière fiable pour stocker les informations concernant les nœuds de réseau et les différents agents de façon à éviter la surcharge de l'agent par le transfert des informations volumineuses et en même temps évité la centralisation.

3- Pourquoi les chercheurs prennent comme objectif de développer leurs propres protocoles pendant qu'il est moins coûteux d'améliorer l'un des protocoles existants.

Dans le chapitre suivant nous allons développer notre approche basée agent mobile pour l'adaptation de réseau ad hoc en répondant aux réflexions précédentes.

Chapitre *IV*

Modélisation d'une approche Basée agent mobile pour l'adaptation d'un réseau mobile ad hoc

4.1 Introduction

Dans ce chapitre nous essayons d'introduire notre approche pour l'adaptation de réseau mobile ad hoc. Notre objectif est de réaliser un système multi agent basé sur des agents mobiles qui fournit des mécanismes pour faire face au problème dû à la mobilité des nœuds.
L'objectif consiste à :

1. La minimisation de l'utilisation de la bande passante par la réduction de la surcharge du réseau avec les informations de routage inutile si le réseau est stable, c'est-à-dire si l'intervalle des mises à jours est calculé en fonction de la mobilité, l'échange des informations de routage entre les nœuds se fait lorsqu'ils ont un vrai effet sur le routage.
2. Possibilité de création des clusters virtuels selon un algorithme qui prend en charge la mobilité des nœuds et la négociation entre les agents pour qualifier un nœud d'appartenir à un tel cluster.
3. Choisir des métriques qui indiquent de mobilité au niveau local au sein du cluster et au niveau global (tout le réseau).
4. Le déplacement des agents mobile se fait selon la dynamique du réseau, c'est-à-dire aux niveaux des nœuds qui disposent la plus grande probabilité de changer des liens d'une façon significative.
5. Pendant leur migration d'un nœud à un autre, l'agent doit recalculer les métriques et selon le résultat il faut modifier l'intervalle de mise à jour.

4.2 Principe

Le principe de notre travail consiste à utiliser deux types d'agents, des agents managers du cluster et d'autres adaptateurs de réseau.

La fonction principale des agents managers est la création et la maintenance des clusters, pour diviser le réseau en clusters les agents managers utilisent un algorithme simple de clusterage à 1 saut. Au niveau de chaque cluster un agent manager est responsable de gestion du cluster, la position de l'agent manager est en fonction de la mobilité des nœuds c'est-à-dire l'agent manager préfère toujours le nœud qui possède la plus faible mobilité.

Les agents adaptateurs de chaque cluster sont créés par l'agent manger, leur fonction est de visiter les nœuds d'un cluster, calculer des métriques et agir sur le protocole de routage afin d'adapter son fonctionnement à la mobilité. Le nombre des agents adaptateurs est en fonction du nombre des nœuds au sein d'un cluster.

Pour réaliser leur objectif, les deux types d'agents travaillent en collaboration, après un tour sur les nœuds, les agents adaptateurs rencontre leur agent manger du cluster pour échanger de l'information et attribuer les nouveaux rôles (les nouveaux nœuds ont visité)

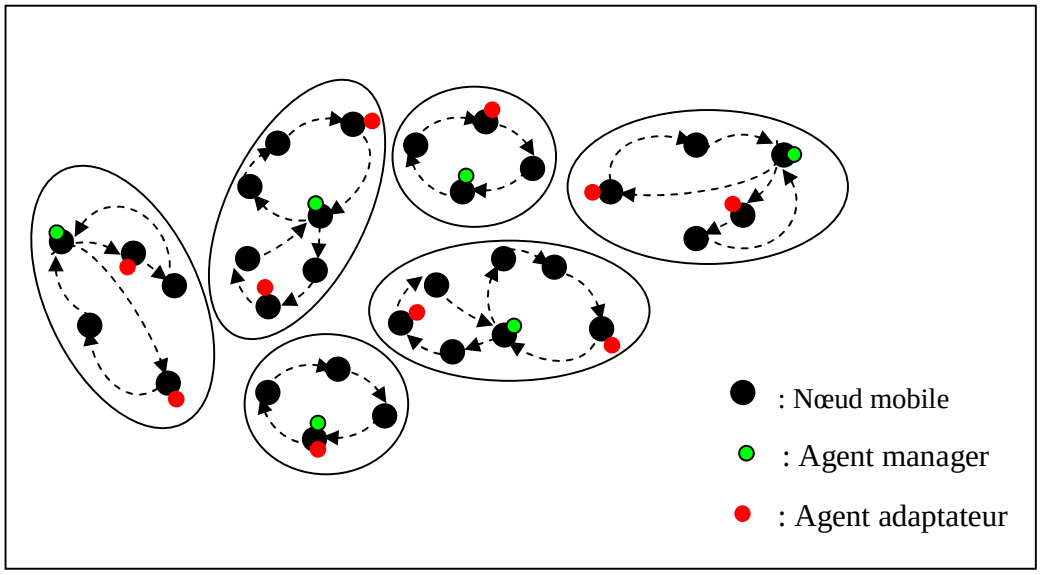

Figure 4.1 : Distribution des agents au niveau de réseau.

4.3 Description

Notre approche consiste à utiliser un ensemble des agents mobiles pour la perception de l'environnement et la détection du niveau de mobilité des nœuds afin de régler les paramètres du protocole de routage proactif.

Dans un protocole proactif à vecteur de distance, chaque nœud possède une table de routage et une table de voisinage, la mise à jour des informations de routage est recommandée lorsqu'il y a des changements de voisinage ou de réseau pour assurer la convergence du réseau. On entend par convergence la correspondance entre les informations de routage avec l'état actuel ou réel du réseau.

La détection des voisins ce fait à l'aide des Messages Hello délivrer par un nœud pour vérifier la présence d'un voisin dans un intervalle donné. Nous remarquons que l'intervalle de temps entre deux Messages Hello à un grand effet sur la convergence, en distingue deux cas de problème lier à ce paramètre :

- Le premier est lorsque la mobilité est forte et l'intervalle de temps et grand le changement des voisinages est très fréquent, ça risque de conserver des informations erronées pour une longue période ce qui engendre des échecs lorsque le nœud veut envoyer des données en plus la consommation de la bonde passante par les messages d'erreurs.

- Le deuxième cas c'est lorsque la mobilité est faible est l'intervalle de temps est petit, les échanges des Messages Hello sont très fréquents, mais non utile se qui consomme la bande passante d'une manière significative.

Dans notre approche nous proposons d'ajuster l'intervalle du temps en terme de mobilité de nœud au niveau local et de réseaux au niveau global.

4.3.1 Métriques proposées

Dans notre approche nous essayons de proposer des métriques de mobilité par rapport à la distribution des nœuds dans le réseau :

- **Densité du Cluster (DC) :** c'est le nombre des nœuds appartenant à un cluster. Cette métrique est essentielle pour déterminer le nombre des agents au sein d'un même cluster.

- **La Stabilité d'un Nœud dans un Cluster (SNC) :** stabilité qu'un nœud reste dans un cluster. Un entier incrémenté par 1 à chaque visite d'un agent à ce nœud s'il est toujours dans le même cluster, mise à zéro lorsqu'il change le cluster.

- **la Fréquence de Changement de Voisinage (FCV) :** c'est le nombre des voisins qui changent par rapport au voisinage de la dernière visite.

- **La Fréquence qu'un Nœud visite un Cluster (FNVC) :** Un entier incrémenté par 1 à chaque fois qu'un nœud visite un cluster. Il est utilisé par l'agent lorsque de migration pour déterminer la direction suivante (priorité d'un nœud par rapport aux autres).

4.3.2 Utilisation des métriques

Les métriques proposées sont utilisées pour étudier la mobilité d'un nœud par rapport au voisinage (est-ce que les nœuds voisins sont fréquemment changés ?) et leur déplacement entre les clusters (est-ce qu'il est stable dans un cluster ou non ?). Selon ces deux métriques on définit trois niveaux de l'intervalle des mises à jour et deux opérations sur ces niveaux.

Les niveaux sont : haut, moyen et faible.
Haut : un grand intervalle de temps.
Moyen : un moyen intervalle de temps.
Faible : un petit intervalle de temps.

Les opérations sont : Augmenter et Diminuer.

Augmenter : basculer vers le niveau supérieur.
Diminuer : basculer vers le niveau inférieur.

Les cas possibles sont les suivants :

N° cas	La stabilité d'un nœud dan un cluster	la fréquence de changement de voisinage	opération
1	Petite	Petite	Augmenter si le niveau est Moyen ou Faible
2	Petite	Grande	Diminuer si le niveau est Haut ou Moyen
3	Grande	Petite	Augmenter si le niveau est Moyen ou Faible
4	Grande	Grande	Diminuer si le niveau est Haut ou Moyen

Table 4.1 : Les opérations correspondent aux différents cas possibles

4.4 Les agents mobiles

Nous utilisons deux types d'agents :

4.4.1 Agent adaptateur

Le rôle de l'agent mobile adaptateur est de déterminer le degré de mobilité de chaque nœud visité afin d'agir sur le protocole de routage utilisé pour améliorer ces performances.

Un cluster peut contenir plusieurs agents adaptateurs, chaque agent est le responsable d'un ensemble des nœuds.

L'agent mobile migre d'un nœud à un autre pour collecter des informations, calculer des métriques et sauvegarder ces informations dans la table d'état.

4.4.1.1 Architecture de l'agent adaptateur

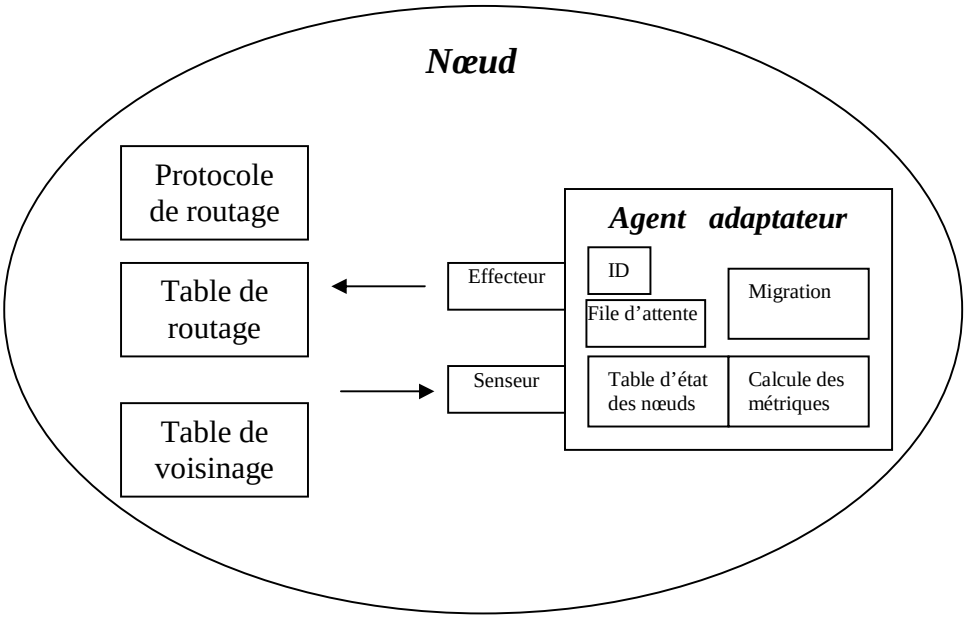

Figure 4.2 : Un nœud hébergeant un agent adaptateur.

Comme il est illustré dans la figure 4.2 chaque agent possède :

- Un identificateur ID :

Cet identificateur est unique pour chaque agent. Il est défini pour distinguer les agent les uns des autres.

- Un table état des nœuds :

Chaque entrée dans cette table correspond à un des nœuds rentrons sous la responsabilité de l'agent, cette table est dynamique et met à jours lorsqu'il y a des changements dans l'environnement. Chaque colonne de la table correspond à une métrique pour sauvegarder les valeurs des métriques calculer pour chaque nœud en plus d'une colonne pour mémoriser les voisins de chaque nœud.

Métrique / Nœuds	SNC	FCV	FNVC	voisins
N1				
…				
N2				

Figure 4.3 : Table d'état des nœuds

- Moteur de calcul des métriques

On s'est basé sur les informations extraites des tables de routage est des voisinages, l'agent utilise ce module pour calculer les métriques et mettre à jour l'état protocolaire du nœud ; ce moteur est en interaction avec la table d'états des nœuds.

- Les Senseurs et les Effecteurs

Ce sont respectivement les outils de perception et d'action sur l'environnement. Les senseurs sont utilisés pour l'extraction des données des tables de routage ou de voisinage et les effecteurs pour agir sur le fonctionnement de protocole et de mise à jour les informations de la table d'état des nœuds.

- Module de Migration

Il est utilisé pour sélectionner la direction suivante de l'agent. La politique de migration est expliquée dans les paragraphes suivants.

- Une fille d'attente

Cette file est utilisée pour mémoriser les nœuds à visiter.

4.4.1.2 Fonctionnement de l'agent adaptateur

L'agent adaptateur se déplace d'un nœud à un autre pour accomplir ses tâches, lorsqu'il arrive à un nœud il fonctionne comme suit :

1. il fait un parcours de la table d'état pour vérifier l'existence du nœud,
   Si oui :

- incrémenter la métrique de stabilité d'un nœud dans un cluster (SNC),
- calcule le voisinage et comparer avec sel de la dernière visite (FCV),
- incrémenter (FNVC) la Fréquence qu'un Nœud Visite un Cluster,
- mettre à jour la table d'état.

   Si non :

- créer une nouvelle entrée dans la table d'état.
- calculer les métriques et le voisinage.
- mettre à jour la table.

2. prendre une décision est faire la modification nécessaire du paramètre du protocole.
3. en utilisant le métrique de la fréquence qu'un nœud visite un cluster (FNVC) et la file d'attente, l'agent sélectionner et migrer vers le nœud suivant.

4.4.1.3 Stratégie de migration de l'agent adaptateur

La stratégie de navigation de l'agent mobile doit assurer que tous les nœuds du réseau sont visités périodiquement. Notre approche propose de diviser un réseau en cluster, chaque cluster est géré par des agents et chaque agent est le responsable d'un ensemble des nœuds. Cette division assure que les agents sont bien distribués de telle façon à éviter le regroupement des agents seulement dans une partie du réseau.

La stratégie choisie pour la migration est : High_mobility_least_viseted_node_first , cette stratégie consiste à visiter les nœuds périodiquement, mais par ordre. En utilisant les informations qu'il possède sur les nœuds, l'agent crée une liste des nœuds à visiter selon l'ordre décroissant de la mobilité de chaque nœud dans le cluster, l'agent se déplace d'un nœud à un autre, recalcule les métriques et sauvegarde les résultats pour le prochain cycle, à chaque fin de cycle l'agent doit rétablir la liste de visite.

La migration entre deux nœuds se fait comme suit : après la détermination du prochain nœud à visiter, l'agent envoi une copie de lui-même (code, données,…), puis il attend la confirmation de la réception et ensuite il exécute une procédure de suppression de l'agent.

4.4.2 Agent manager

Dans notre approche la construction des clusters virtuels joue un rôle très important pour le contrôle de la mobilité des nœuds, les métriques proposées sont basées sur cette topologie pour garantie un bon suivie du nœud. La fonction principale de l'agent manger est d'assurer la création et la maintenance des clusters.

4.4.2.1 Architecture de l'agent manager

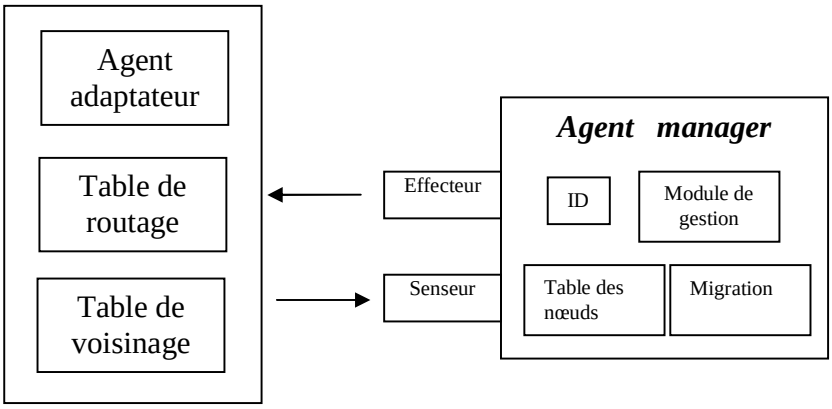

Figure 4.4 : L'agent manager.

L'agent manager est constitué des éléments suivants (figure 4.4) :

- Un identificateur ID :

Cet identificateur est unique pour chaque agent. Il est défini pour distinguer les agents les uns des autres.

- Module de Migration :

Utiliser pour sélectionner le nœud le plus stable dans le cluster, avant la migration.

- Module de Gestion

Utiliser pour sélectionner le nœud le plus stable dans le cluster, avant la migration.

- Une table des nœuds

Chaque entrée dans cette table correspond à un des nœuds formant le cluster, combiner avec la valeur de stabilité de chaque noeud, cette table est dynamique est mise à jours lorsqu'il y a des changements dans l'environnement.

- Les Senseurs et les Effecteurs :

Les senseurs sont utilisés pour l'extraction des données de la table de routage ou de voisinage et les effecteurs pour l'échange des messages avec l'agent adaptateur.

4.4.2.2 Fonctionnement de l'agent manager

Chaque agent manager est le responsable de la création et la gestion d'un cluster, il collecte des informations, calcule la densité de cluster, crée et envoie des agents adaptateurs aux différents nœuds.

4.4.2.2.1 Création et maintenance des clusters

La recherche de voisinage est utilisée dans de nombreux domaines, tels la reconnaissance de formes, le *clustering*, l'approximation de fonctions, la prédiction de séries temporelles et même les algorithmes de compression (recherche d'un groupe de données le plus proche possible du groupe de données à compresser pour minimiser l'apport d'information).

- **Recherche des plus proches voisins (ou des *k* plus proches voisins)**

Le problème de la recherche des plus proches voisins (ou des *k* plus proches voisins) est très courant en algorithmique et de nombreux auteurs ont proposé des algorithmes efficaces pour le résoudre rapidement.

Soient :

- un espace *E* de dimension *D* ;
- un ensemble *A* de *N* points dans cet espace ;
- un entier *k* plus petit que *N*.

La recherche des plus proches voisins consiste, étant donné un point *x* de *E* n'appartenant pas nécessairement à *A*, à déterminer quels sont les *k* points de *A* les plus proches de *x*. On parle alors de trouver un voisinage de taille *k* autour du point *x*.

Dans notre approche nous utilisons une version modifiée de l'algorithme de k-plus proche voisins. Les agents managers doivent coordonner et coopérer pour la division du réseau en clusters selon la distribution des nœuds au niveau de réseau dont chaque cluster est sous la responsabilité de l'un des agents managers. La description de l'algorithme est la suivante :

Au début, un seul agent manager est lancé à partir d'un nœud quelconque. Il ajoute tous les voisins à un saut dans sa table des nœuds, ces nœuds sont considérés comme le premier cluster, ensuite l'agent manager se duplique et envoie des copies vers les nœuds hors de cluster (les voisins a deux sauts) et chaque agent commence à créer son propre cluster et ainsi de suit.

Lorsqu'un agent manager arrive à un nœud, il test le voisinage et s'il y a des voisins libres, il collecte pour créer un cluster. Dans le cas contraire, il choisi le plus proche cluster pour ce nœud. Un nœud est plus proche à un cluster implique qu'il possède le plus grand nombre de connexion avec les nœuds de ce cluster.

Algorithme création et maintenance des clusters

```
Début
- Lancement de l'agent manager A1 sur un nœud n ;
- Construction de premier cluster C1 = {ni} ; /* voisins <= k saut */
- Sélectionner C2={nj} l'ensemble des voisins de k+1 saut ;
- Duplication de A1 et envoi d'une copie à chaque élément de nj ;
- Pour tout agent manager sur nj faire
        * sélectionner E l'ensemble des voisins libres de nj ;
            Si vide(E) alors          /* pas de voisins */
                Ajouter nj au plus proche cluster ;
            Sinon
                Créer un nouveau cluster ;
             Fin si ;
   Fin pour ;
Fin début.
```

Lorsqu'un nœud réside à la frontière entre deux clusters, sa position est à négocier entre les agents responsables de ces clusters.

Algorithme de négociation

Début
- Si un nœud N appartient à un cluster C1 est détecté comme plus proche à un autre cluster C2 alors N est ajouté à C2
- Si le nombre des liaisons est égal alors
    Si un voisin de N dans la même situation alors
        Fusionner les clusters ;
    Sinon
        Riens a changé ;
    Fin si ;
Fin si ;
Fin début ;

4.4.3 Flexibilité de population d'agents

La fusion et la duplication des clusters sont des cas prévus à cause de la dynamique du réseau. Les agents doivent fournir des mécanismes pour s'adapter à ces situations, par exemple la duplication d'un cluster nécessite des fois la création des nouveaux agents ou la redistribution des agents sur les nouveaux clusters résultants….etc.

4.4.4 Adaptation structurelle de l'agent

Lorsque un cluster se duplique en plusieurs sous cluster, les agents doivent s'adapter leur structure pour correspondre à la situation actuelle (changer ID, table d'état des nœuds…), aussi l'agent mobile se duplique pour créer un autre agent mobile similaire en terme de code, mais sans mémoire et sans données.

4.4.5 Adaptation comportementale de l'agent

Comme nous avons déjà dit, lorsque la détection d'un nouveau cluster l'agent mobile se duplique pour créer un autre agent mobile similaire en terme de code, mais sans mémoire et de données, ce dernier migre vers le nouveau cluster pour l'apprentissage de son environnement est la construction de leurs données.

La population d'agents dans chaque cluster est dynamique, sa progression est en fonction de nombre des nœuds inclus dans le même cluster. Dans quelque cas, il est possible de supprimer des agents lorsque le nombre est plus que le nombre nécessaire pour gérer le cluster. Contrairement, l'ajout des agents est recommandé lorsque le nombre des nœuds est augmenté.

Notre étude expérimentale montre que le meilleur nombre des agents est le quintupler du nombre des nœuds au sein de même cluster.

4.4.6 Communication entre les agents

La communication entre agents se fait à l'aide des messages, pour faire face au problème de déconnexion de lien de communication nous recourons vers un mécanisme de rendez-vous pour assurer une communication fiable. L'échange des messages entre un agent manager et ces agents adaptateurs se fait au niveau de même nœud.

4.4.7 L'interaction entre les agents

4.4.7.1 Interaction générale entre les agents

Pour bien comprendre le fonctionnement de notre architecture, nous présentons le protocole d'interaction dans une séquence d'étapes suivantes :

1- l'agent manager lance une opération de création ou de maintenance d'un cluster. Supposons que le résultat de cette opération est un cluster de trois nœuds N1, N2 et N3

2- L'agent manager prépare les rôles est distribue sur le nombre des agents adaptateurs nécessaire pour ce cluster. Dans ce cas égale à 1.

3- L'agent adaptateur, après la réception de ses rôles, il se déplace d'un nœud à un autre,

4- Au niveau de chaque nœud, il exécute des opérations pour réaliser ses rôles.

5- L'agent adaptateur retourne vers l'agent manager pour échanger des données.

6- L'agent manager réceptionne les données, détruit l'agent adaptateur et commence à nouveau avec les nouvelles informations.

L'interaction globale entre les différentes classes d'agents dans le système est présentée sous forme d'un modèle SDM (Sequence Diagram for Mobility) [51].

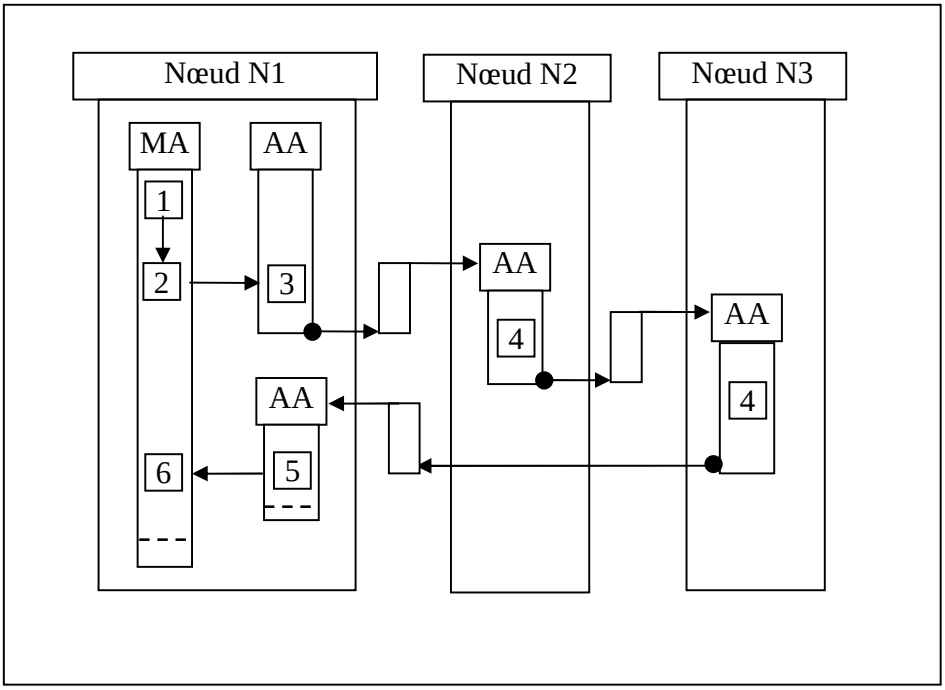

Figure 4.5 : Interaction générale entre les agents.

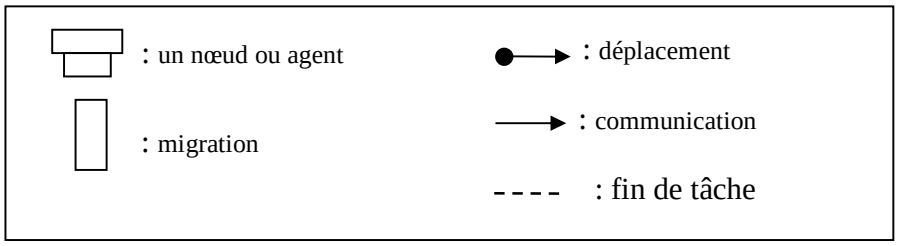

Figure 4.6 : Signification des éléments graphiques

4.4.7.2 Interaction détaillée entre les agents

Pour bien comprendre le fonctionnement de notre architecture nous présentons le protocole d'interaction en AUML entre un agent manager avec un agent adaptateur au sein d'un même cluster est un agent manager d'un autre cluster.

La notation AUML [52] est fortement centrée sur l'interaction entre les agents. Les diagrammes de séquence de AUML sont très similaires à ceux de UML, mais avec l'addition des nouveaux artefacts graphiques avec sa sémantique associée. Ces opérateurs sont principalement de deux types: les opérateurs d'exécution et les opérateurs d'interaction. Tous se réduisent à trois: XOR, OR, et AND, tant pour contrôler l'exécution comme pour l'envoi de messages.

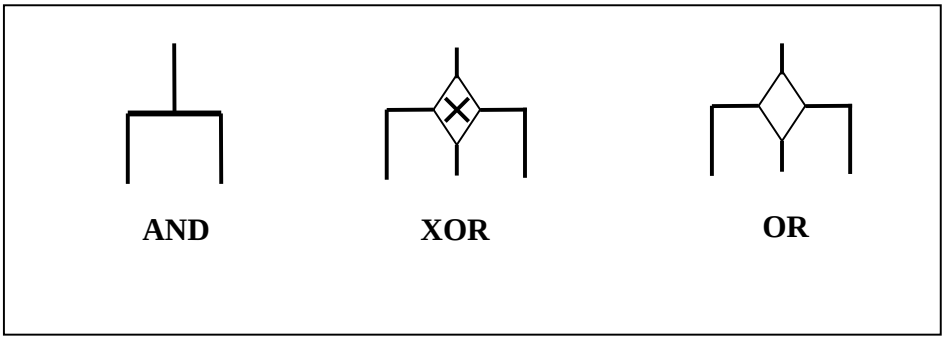

Figure 4.7 : Les opérateurs AND, XOR et OR en AUML

Ils ont été ajoutés pour essayer de modéliser et d'exprimer d'une façon plus complète les interactions entre les agents. AND (qui représente le parallélisme), XOR (un seul message parmi une liste à envoyer) et OR (choix non déterministe entre une ou plusieurs alternatives).

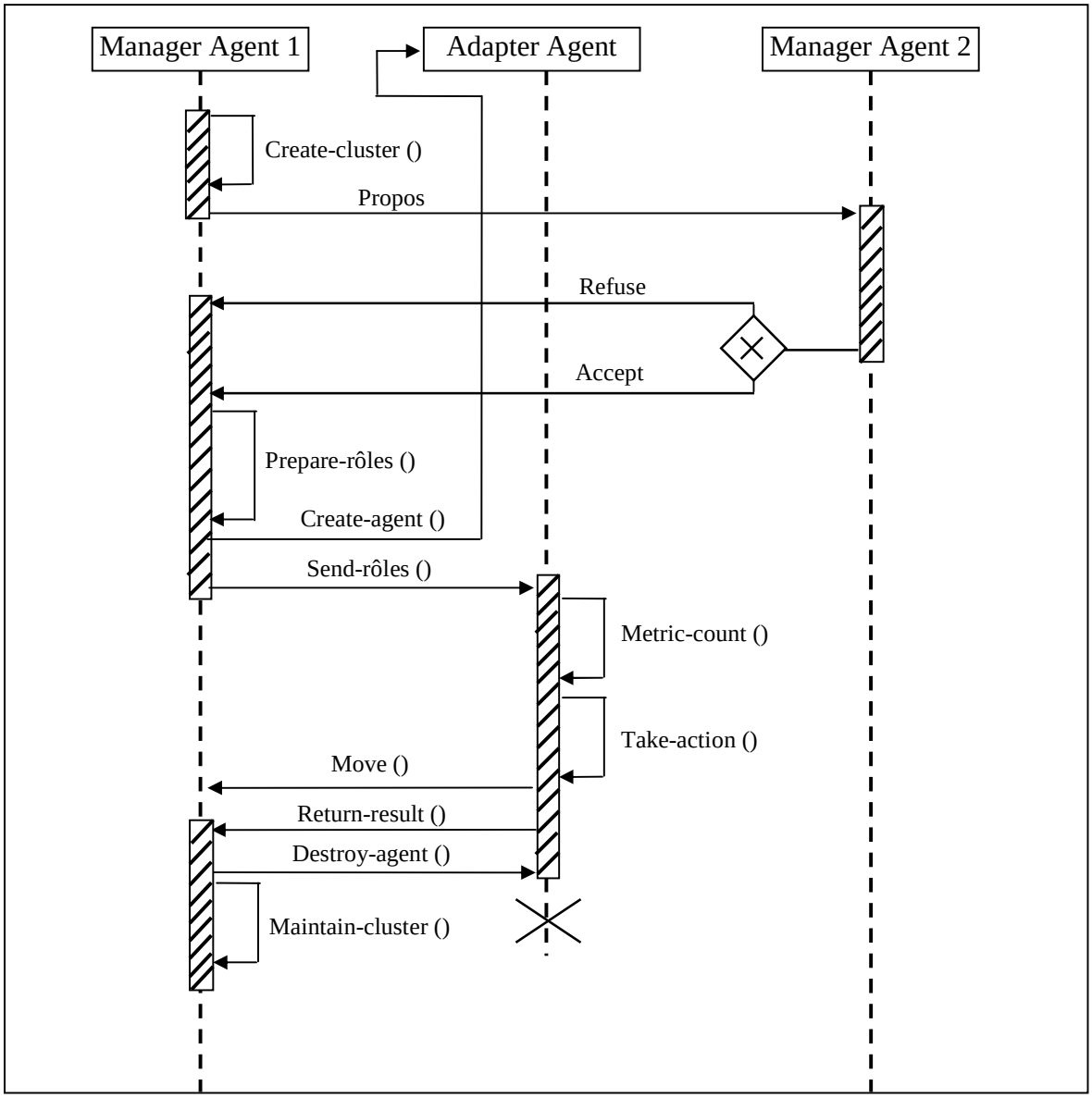

Figure 4.8 : Diagramme de séquence qui représente l'interaction entre les agents du système

4.5 Conclusion

Dans ce chapitre, nous avons présenté une architecture basée agent mobile pour l'adaptation de réseaux ad hoc à la mobilité des nœuds. Cette architecture comporte les concepts nécessaires pour assurer les exigences que nous avons prises en compte afin d'assurer le bon fonctionnement du système proposé. Alors, l'idée proposée pour l'adaptation de réseau ainsi que les structures internes des différents agents sont illustrées, les mécanismes d'interaction, de migration et de communication sont discutés.

Dans le prochain chapitre, et afin d'évaluer le système proposé, nous allons procéder à la simulation à l'aide de l'outil JiST/SWANS (Java in Simulation Time/ Scalable Wireless ad-hoc Network Simulator) et la plateforme JADE(Java Agent DEvelopement framework). Ceci nous permettra d'effectuer un ensemble de tests pour mesurer les performances de notre système par rapport aux approches traditionnelles des protocoles de routage.

Chapitre V

Implémentation

5.1 Introduction

Dans le chapitre précédent de ce mémoire, nous avons proposé une approche basée agents mobiles pour l'adaptation à la mobilité dans un réseau mobile ad hoc. Afin d'évaluer notre proposition, nous faisons appel à des simulations pour interpréter des expériences et réaliser une étude comparative avec un protocole existant.

Dans un premier temps, nous commençons par une description de notre travail. Nous détallions le protocole que nous avons choisi (ZRP), son fonctionnement et ses sous-protocoles. Par la suite, nous avons décrit l'implémentation de notre architecture pour montrer comment nous l'exploitons dans le cadre de notre travail pour améliorer le fonctionnement de ce protocole.

Ensuite nous présentons notre environnement de travail, c'est-à-dire les outils et les logiciels que nous avons utilisé à savoir : le simulateur JiST/SWANS , la plateforme JADE (Java Agent DEvelopement framework) et l'environnement java NetBeans IDE 6.7.

Et enfin en conclus par la comparaison des résultats obtenus à partir de la simulation entre le protocole ZRP (fonctionnement traditionnel) et ce améliorer avec notre approche.

5.2 Description de travail

Les protocoles hybrides tirent avantage des méthodes réactives et proactives et limitent leurs inconvénients. Ils découpent généralement le réseau en zone à l'intérieur desquelles ils appliquent une politique de routage proactive. De cette manière, le processus de routage intra zone est accéléré. Le nombre de nœuds d'une zone est limité pour réduire la charge engendrée par la maintenance des tables de routage.

Lorsqu'un noeud souhaite communiquer avec un noeud faisant partie d'une autre zone, une politique réactive est alors mise en place pour le routage interzone.

5.2.1 Zone Routing Protocol (ZRP)

Dans ZRP [7], une zone $Z(k; n)$ pour un noeud n avec un rayon k, est définie comme l'ensemble des noeuds à une distance inférieure ou égale à k sauts :

$Z(k; n) = fi\ j\ H(n; i) \cdot kg,$ où $H(i; j)$ est la distance en nombre de sauts entre le noeud i et le noeud j. Le noeud n est appelé le noeud central de la zone de routage, alors que le noeud b tel que $H(n; b) = k$ est appelé le noeud frontière de n.

La taille d'une zone affecte les performances de communication et doit être optimisée en fonction du degré de mobilité, de trafic, ainsi que du diamètre du réseau.

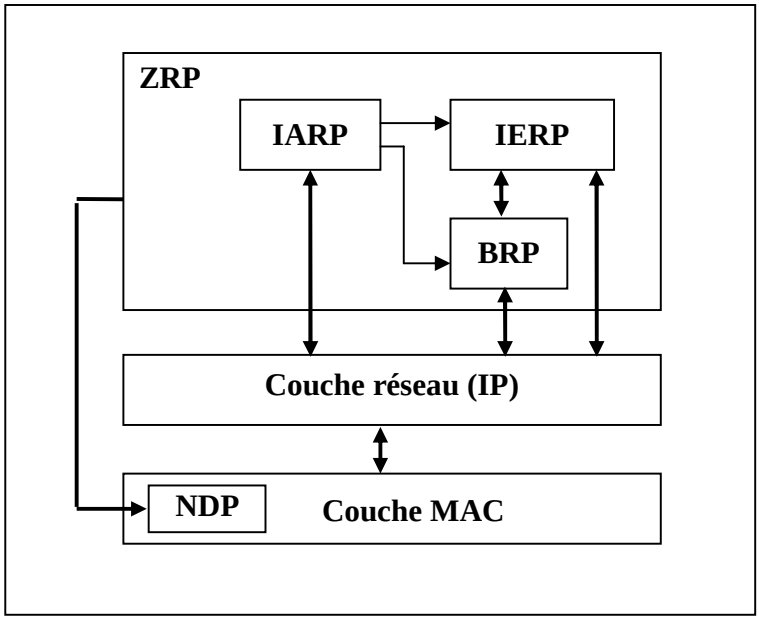

Figure 5.1 : Interaction de ZRP avec les autres couches.

L'architecture du protocole ZRP (Figure 5.1) est composée de quatre sous-protocoles :

1. l'IntrAzone Routing Protocol (IARP)
2. l'IntErzone Routing Protocol (IERP)
3. le Bordercast Resolution Protocol (BRsP) et
4. le Neighbor Discovery/Maintenance Protocole (NDP) situe au niveau de la couche 2.

IARP fournit de manière proactive les routes aux noeuds situés dans la même zone que la source. IARP repose sur NDP pour découvrir ses voisins, son rôle principal est d'assurer que chaque noeud au sein d'une zone possède une table de routage à jour reflétant la route à emprunter pour chaque noeud de la même zone.

IERP, quant à lui, repose sur les nœuds de frontières afin de découvrir de manière réactive les routes interzones. D'une manière plus précise, IERP émet des paquets de requêtes aux noeuds de frontières via BRP, une sorte d'algorithme multicast, afin qu'ils vérifient si le noeud destinataire fait partie de leur zone respective. Si c'est le cas, il génère un paquet de réponse vers la source. Sinon, ils propagent la requête vers leurs nœuds de frontières.

Le problème de ce protocole est qu'il n'y a pas de coordination entre les noeuds, il en résulte que les zones se chevauchent et un noeud peut être à la fois membre d'une zone et noeud frontière de plusieurs zones. Dans ces conditions, l'algorithme de recherche peut conduire à des résultats moins bons qu'une diffusion standard.

5.2.2 Application de notre approche

Notre approche repose sur les informations contenues dans la table de routage et sur l'état des voisins de chaque nœud. Pour cela nous précisons notre travail au tour des sous protocoles IARP et NDP de ZRP. Comme nous avons déjà dit : le rôle principal de IARP est d'assurer que chaque noeud au sein d'une zone possède une table de routage à jour reflétant la route à emprunter pour chaque noeud de la même zone. Dans l'autre côté, la fonction principale de NDP est de découvrir les voisins.

Notre approche consiste à utiliser un ensemble des agents mobiles pour la perception de l'environnement et la détection de niveau de mobilité des nœuds en basant sur IARP et NDP.

IARP est utilisé comme la partie proactive du protocole ZRP. En combinant avec les informations sur les voisins fournis par le protocole NDP de la couche 2 (MAC), les tables de routage sont utilisées pour l'extraction des informations sur l'état actuel du réseau afin de calculer l'ensemble des métriques pour prendre la décision nécessaire pour la régulation des paramètres du protocole lui-même afin d'adapter le réseau (fonctionnement de protocole) à la mobilité des nœuds.

5.2.3 Fonctionnement des agents

Il est à Noter ici que la procédure de création des clusters par les agents Managers annoncer dans le chapitre précédent est indépendante des clusters gérer par le ZRP, la seule relation entre eux est que les agents doivent limiter aux nœuds au sein de chaque zone ZRP tout simplement parce que IARP est limité à une seule zone.

Chaque agent manager commence, tout d'abord, par créer ou maintenir son propre cluster à partir de sa position, cette position est calculable à chaque fois(le nœud le plus stable), en utilisant un algorithme de k-voisins ou k=2, le chois de la valeur 2 est par expérimentation (meilleur résultat avec la valeur 2).

Pour réaliser cette fonction, l'agent manager utilise les tables de routage IARP et de voisinage NDP.

Ensuite, selon la densité de cluster (nombre des noeuds) l'agent manager créer un nombre des agents Adaptateurs, selon notre expérimentation chaque agent Adaptateur correspond à cinq (5) nœuds.

Nombre des agents Adaptateur = nombre des nœuds dans le cluster / 5 si le reste=0

ou

Nombre des agents Adaptateur = (nombre des nœuds dans le cluster / 5)+1.

L'agent Manager distribue les rôles de chaque agent Adaptateur, lance et attend le retour des agents Adaptateurs pour recommencer son travail.

Chaque agent Adaptateur est responsable de parcourir tous les nœuds enregistrés au niveau de sa file d'attente, calcule les métriques et prend la décision de modifier les paramètres de protocole. Ici nous parlons de la période de détection des voisinages ou la période des messages hello. Cette décision concerne l'augmentation ou la diminution de la période estimée en second.

Dans l'implémentation de ZRP dans le simulateur JIST/SWANS cette période est fixée à 10 Seconds.

Pour notre approche nous avons définit trois niveaux : haut, moyen et faible.

Haut : un grand intervalle de temps soit 15 Seconds.

Moyen : un moyen intervalle de temps soit 10 Seconds

Faible : un petit intervalle de temps soit 5 Seconds.

Lorsqu'il arrive à un nœud, l'agent adaptateur fonctionne comme suit :

4. il fait un parcours de la table d'état pour vérifier l'existence du nœud (encore dans le même cluster),

   Si oui :

   - incrémenter la métrique de stabilité d'un nœud dans un cluster (SNC), c'est-à-dire que l'agent arrive sur un nœud qui est toujours dans le même cluster.

     SNC=SNC+1.

     Remise à 0 : lorsque c'est la première visite.

   - Calculer le voisinage et comparer avec ce de la dernière visite (FCV),

     FCV= nombre de liens changer depuis la dernière visite.

     FCV= card ({voisins a l'instant t}- {voisins a l'instant t-1 }).

   - Incrémenter (FNVC) la Fréquence qu'un Nœud Visite un Cluster,

     FNVC=FNVC+1.

   - Mettre à jour la table d'état en ajoutant les informations propres au nœud visité.

     (FNC, FCV, FNVC et l'ensemble des voisins)

   Si non :

   - Créer une nouvelle entrée dans la table d'état.
   - Calculer les métriques et le voisinage.
   - Mettre à jour la table.

5. Prendre une décision est faire la modification nécessaire de paramètre du protocole en fonction de FNC et FCV. En combinant les deux métriques, l'agent Adaptateur décide quelle opération (Augmenter ou Diminuer) qui va s'exécuter.

   - Comme nous avons déjà indiqué dans le chapitre précédent, le degré de la mobilité est déterminé par rapport à FNC et FCV. Nous avons défini deux niveaux : petit et grand. Empiriquement nous proposons que :

     Pour FNC :

     - petit si FNC < =2
     - grand si FNC >2

Pour FCV :

- petit si FCV <=5
- grand si FCV >5

6. En utilisant la métrique de la fréquence qu'un nœud visite un cluster (FNVC) et la file d'attente, l'agent sélectionne et migre vers le nœud suivant.
   Les nœuds sont triés dans un ordre croissant, de cette façon l'agent visite les nœuds les plus mobiles en premier. La motivation de ce choix c'est quand un nœud devient de plus en plus mobile, il est plus empoisonné par les changements de l'environnement.

Le scénario de la simulation et les résultats obtenus sont expliqués plus loin dans ce chapitre.

5.3 Environnement de travail

Dans cette section, nous présentons les outils et les logiciels que nous avons utilisé : le simulateur JiST/SWANS , la plateforme JADE (Java Agent DEvelopement framework) et l'environnement java NetBeans IDE 6.7.

5.3.1 Le simulateur JiST/SWANS

La plateforme JiST/SWANS [53], développée à l'université de Cornell, forme la base de notre simulateur java. JiST (Java in Simulation Time) est un moteur de simulation à événements discrets qui tourne au-dessus d'une java virtuelle machine (JVM). SWANS (Scalable Wireless Ad-hoc Network Simulator) est un simulateur de réseau qui s'exécute au-dessus de JiST.

Le système de Jist, comme illustré à la Figure 5.2, se compose de quatre éléments distincts [54] : un compilateur, un byte code et un rewriter et le noyau de simulation avec la machine virtuelle.

On écrit un programme de simulation en Java, ensuite compilé en classes bytecode en utilisant un compilateur de langage Java ordinaire. Ces classes compilées sont ensuite modifiées avec le rewriter ou graveur de bytecode et en fin lancer la simulation.

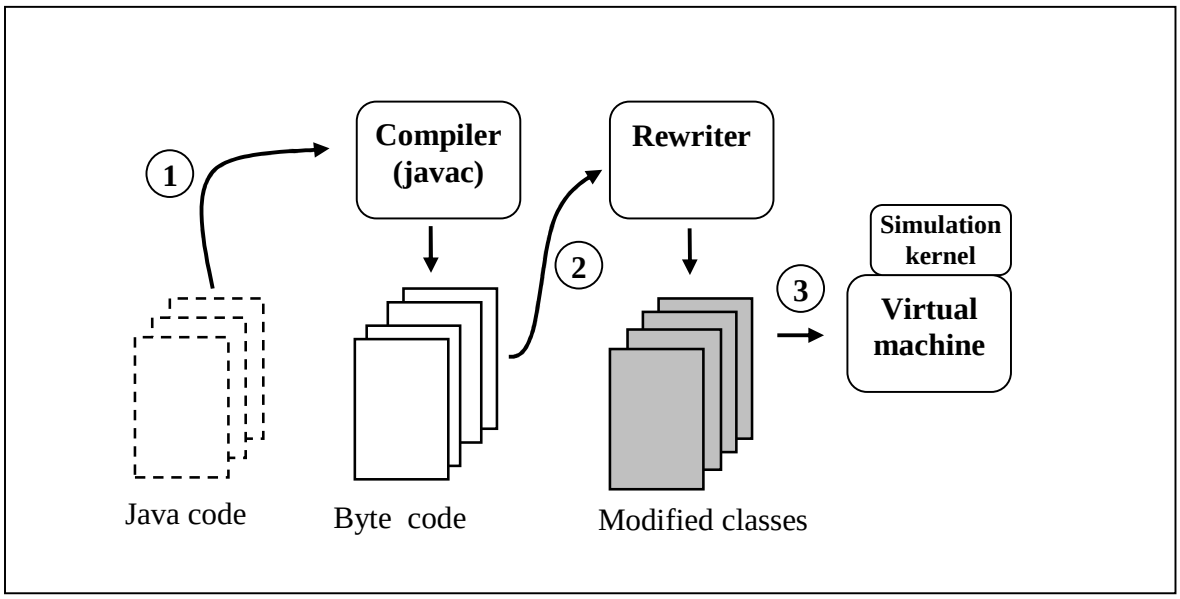

Figure 5.2 : Les composants de Jist.

Notre choix de ce simulateur est justifié par les points suivants :

1. Le plus grand nombre des appareils mobiles fournissent une plateforme de programmation en java. Par conséquent, simplifier la mise en œuvre de l'application dans un cas réel.
2. En second lieu, JiST/SWANS contient une pile assez robuste de réseaux avec des réalisations assez complètes de 802.11 et de TCP/IP.
3. Troisièmement, JiST/SWANS permet l'exécution d'un programme java sans modifications importantes.
4. Enfin, JiST/SWANS permet de manipuler un nombre énorme de noeuds lors de la simulation et ceci dans des conditions Temps/espace raisonnables.

D'après la Figure 5.3, JiST/SWANS fournit un modèle en couche de réseau. Nous récapitulons les principaux composants de chaque couche telle qu'elles sont implémentées dans SWANS comme suit :

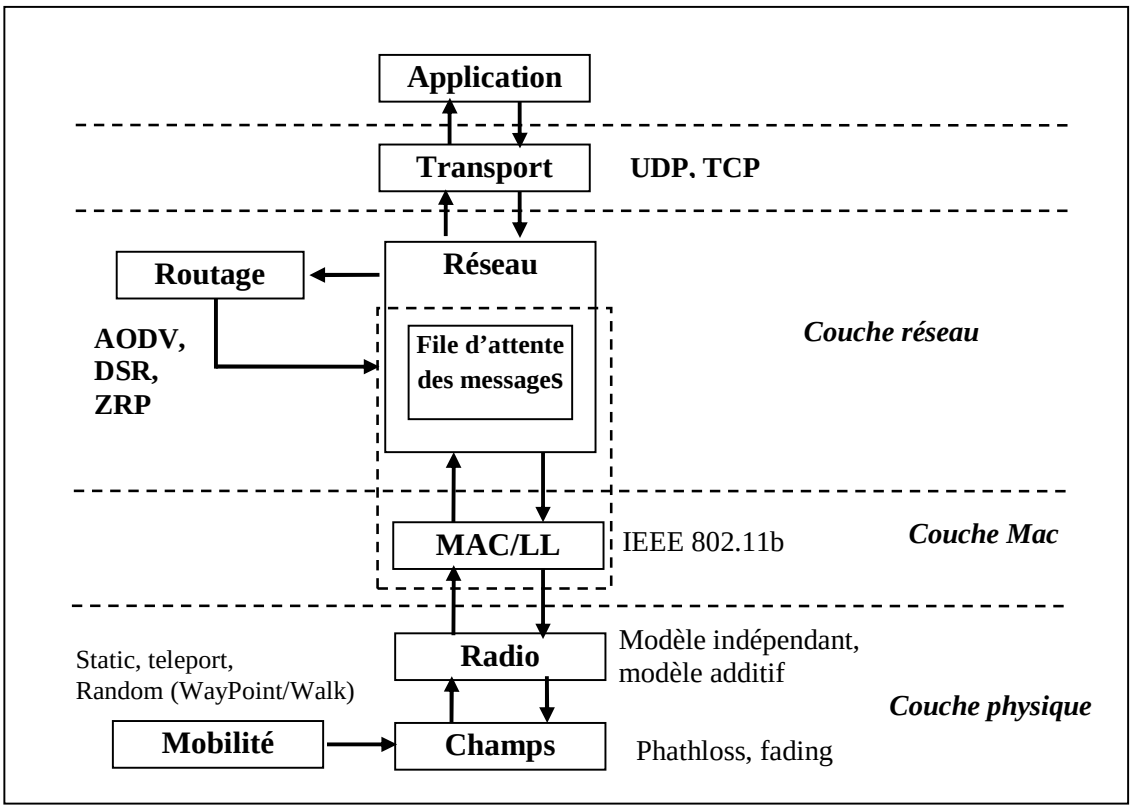

Figure 5.3 : Architecture de simulateur JIST/SWANS.

A. Couche physique

Les composants de la couche physique sont responsables à la modélisation de la propagation du signal entre les entités radios et de la mobilité des noeuds. Cette couche contient deux principales entités :

- ***Radio*** : Chaque entité radio est identifiée par un identificateur ID et elle est paramétrée par une fréquence, une puissance de transmission, une bande passante,...etc. Ceux-ci peuvent être différents pour chaque noeud. Il y a deux modèles de bruit fournis par l'entité radio: le modèle d'interférence indépendant qui considère seulement les signaux destinés à la radio cible comme interférence. Cependant, le modèle d'interférence additif considère que tous les signaux contribuant à l'interférence.
- ***Champs*** : Il représente l'espace dans lequel existent les radios. C'est là également que les modèles de perte de chemin et la mobilité (statique, aléatoire,...) sont implémentés.

Notons que SWANS implémente quatre modèles de mobilité à savoir : Static, Random (Waypoint/ Walk) et Teleport.

- ***Static Mobility*:** Tous les noeuds sont immobiles tout au long de la simulation.
- ***Random Waypoint Mobility* :** Un mobile commence par rester un certain temps à la place où il se trouve puis il détermine une destination ainsi qu'une vitesse (m/s) appartenant à l'intervalle [*SPEED_MIN, SPEE_DMAX*]. Une fois arrivé à destination, le mobile marque à nouveau un arrêt puis reprend le processus de recherche de destination et de calcul de vitesse.
- ***Random Walk Mobility* :** Un mobile consiste à sélectionner une direction, et se déplace sur une certaine distance en suivant cette direction, puis fait une pause. Le mobile reprend ce processus jusqu'à la fin de la simulation.
- ***Teleport Mobility*** : Un mobile se télé porte vers une destination aléatoirement, marque un temps de pause et reprend le processus.

Pour augmenter la vitesse de simulation, SWANS divise ce champ dans des grilles de cellules pour réduire le calcul requis quand les radios sont hors de portée l'un de l'autre.

B. Couche Liaison de données

La couche liaison de données est responsable de l'implémentation du protocole d'accès au médium et de l'encapsulation des paquets IP dans des trames. En effet, cette couche inclut l'implémentation de protocole IEEE 802.11b.

- Mac 802.11b: 802.11b est actuellement le seul protocole implémenté dans SWANS. Il inclut la fonctionnalité complète de la technique d'accès DCF avec des retransmissions.

C. Couche réseau

- **TCP/IP** : L'implémentation de protocole TCP/IP dans SWANS ne supporte pas la fragmentation d'un réseau mobile ad hoc. Elle détient une file d'attente avec priorité des paquets et ne gère pas les débordements. Elle génère, simplement, une exception lorsqu'elle devient pleine. Notons que IPv4 est le seul protocole implémenté dans la couche réseau.

- **Routage** : L'entité routage reçoit des messages envoies par l'entité réseau pour les paquets qui nécessitent l'information sur le prochain saut. Cette entité de routage offre trois protocoles, de à savoir : AODV, DSR et ZRP.

Pour situer les performances de ce simulateur par rapport aux autres, la table 5.1 [58] présente quelques résultats de performances (temps, espace mémoire) des benchmarks réalisés sur trois simulateurs qui sont : SWANS, GloMoSim [56] et NS-2 [57]. L'objectif des benchmarks consiste à mesurer le temps et l'espace nécessaires lors de la simulation du protocole de découverte d'un noeud (NDP).

Nodes	Simulator	Time	Memory
500	**SWANS**	54s	700 KB
	GloMoSim	82 s	5759 KB
	Ns 2	7136s	58761 KB
5,000	**SWANS**	3250s	4885 KB
	GloMoSim	6191s	27570 KB
50,000	**SWANS**	312019s	47717 KB

Table 5.1 : Performances de SWANS.

D'après ces résultats, JiST/SWANS peut simuler des réseaux de tailles très importantes que ceux possibles avec GloMoSim et NS-2 en utilisant la même durée de simulation et d'espace mémoire. Notons que JiST/SWANS peut simuler un réseau d'un million de nœuds sur une machine uni-processeur de 2.0 GHz avec 2 GB de RAM. Pour plus d'informations, les lectures peuvent se référer au guide de JiST/SWANS [55].

5.3.2 La plate-forme JADE (Java Agent DEvelopement framework)

À l'heure actuelle, il existe diverse plat-forme pour le développement d'agents, ils peuvent être classifiés selon la nature des agents supportés, l'interaction, la communication entre agents et les standards utilisés.

5.2.2.1 Description générale de la plate-forme JADE

JADE (Java Agent DEvelopement framework) [59] est une plate-forme multi-agent créée par le laboratoire TILAB. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA [60]. Elle est implémentée en JAVA et fournit des classes qui implémentent « JESS » pour la définition du comportement des agents. Toute la communication entre agents est exécutée par messages FIPA ACL [60].

JADE fournit les facilités suivantes :

- **Une plate-forme agent distribuée :** la plateforme peut être distribuée (partagée) entre plusieurs hôtes connectées via RMI de Java, de telle façon qu'une seule application Java, par conséquent une seule "Machine Virtuelle Java" est exécutée sur chaque hôte.
- **Une interface utilisateur graphique (GUI):** l'interface GUI assure un traitement plus commode de la plate-forme, elle permet à l'utilisateur d'exécuter plusieurs ordres tels que créer un nouvel agent dans la même plate-forme, cloner l'agent, le déplacer, le suspendre, le tuer, etc.…
- **Un support d'exécution :** pour les activités multiples, parallèles et concurrentes des agents via le modèle du comportement (behaviour).
- **Un transport efficace des messages ACL :** à l'intérieur de la même plate-forme.
- **Une bibliothèque de protocoles :** compatibles aux standards FIPA et prêts à être employés pour gérer l'interaction inter-agent.

Lorsqu'on lance la plate-forme, on remarque que JADE a trois modules qui sont actifs au démarrage de la plate-forme, ces modules sont (voir Figure 5.4) :

- **DF** « Director Facilitor » fournit un service de « pages jaunes» à la plate-forme ;
- **ACC** «Agent Communication Channel » gère la communication entre les agents ;
- **AMS** « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

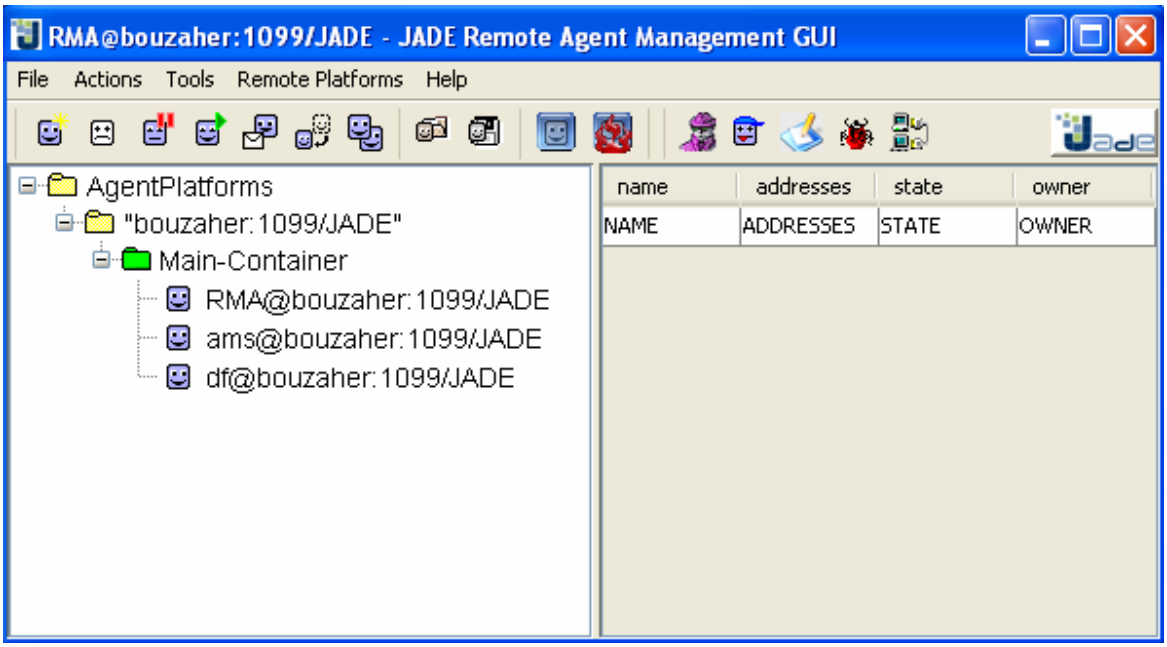

Figure 5.4: Les modules de JADE.

5.3.3 L'environnement NetBeans IDE 6.7

Parmi les environnements de développement intégré, nous choisissons NetBeans IDE sous la version 6.7 de constructeur Sun.

5.3.3.1 Description de NetBeans IDE

NetBeans [61] est un environnement de développement intégré (IDE) open source. Il est développé par Sun [62] et se trouve sous licence CDDL (Common Development and Distribution License). En plus de Java, il propose tous les outils nécessaires à la création d'applications professionnelles pour les particuliers, les entreprises, le web et les applications mobiles avec le langage C / C + +, et même les langages dynamiques tels que PHP, Javascript, Groovy, Ruby XML et HTML. NetBeans IDE est facile à installer et à utiliser. Il comprend toutes les caractéristiques d'un IDE moderne (coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web, etc).

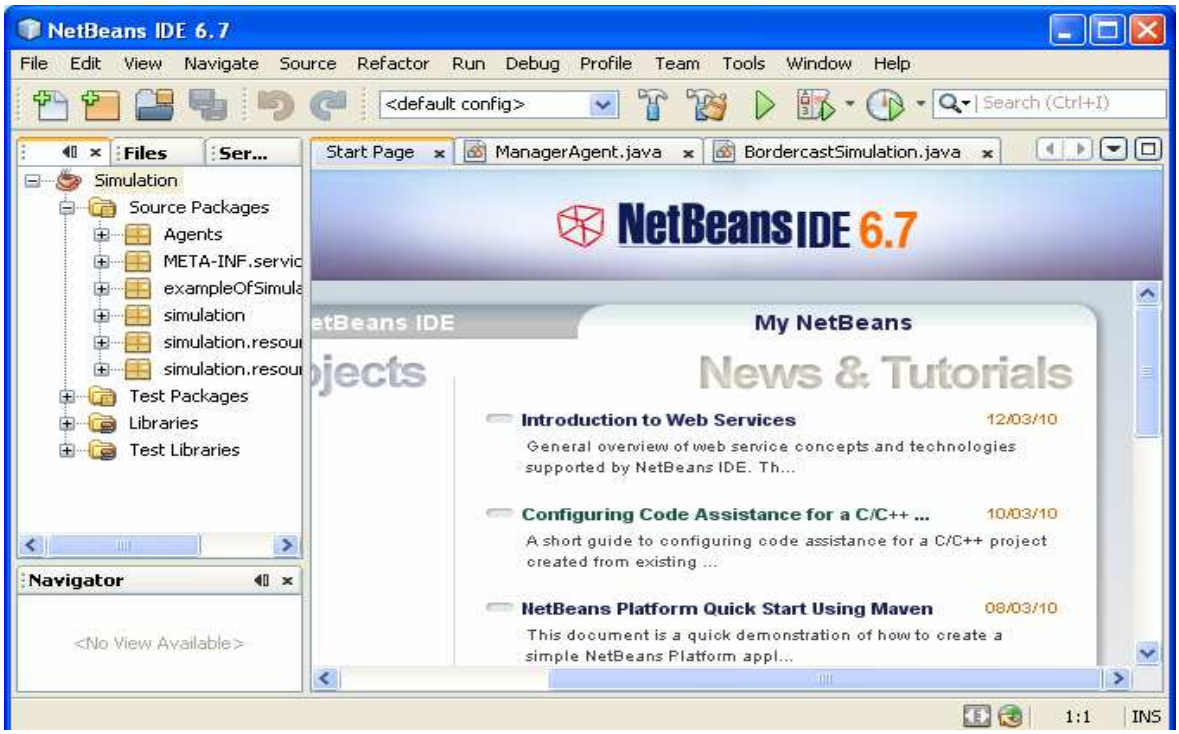

Figure 5.5 : NetBeans IDE 6.7

5.3.3.2 intégrations de JADE et JiST/SWANS

Pour intégrer la plateforme JADE et le simulateur JIST/SWANS dans l'environnement NetBeans IDE il suffit d'ajouter les fichiers jar nécessaire correspond a chacun des deux :

- **Les fichiers jar de JADE :**
  - jade.jar
  - http.jar
  - iiop.jar
  - jadeTools.jar
  - CalendarBean.jar

- **Les fichiers jar pour JIST/SWANS :**
  - Jist.jar
  - Swans.jar
  - bcel.jar
  - bsh.jar
  - checkstyle-all.jar
  - jargs.jar
  - jython.jar
  - log4j.jar

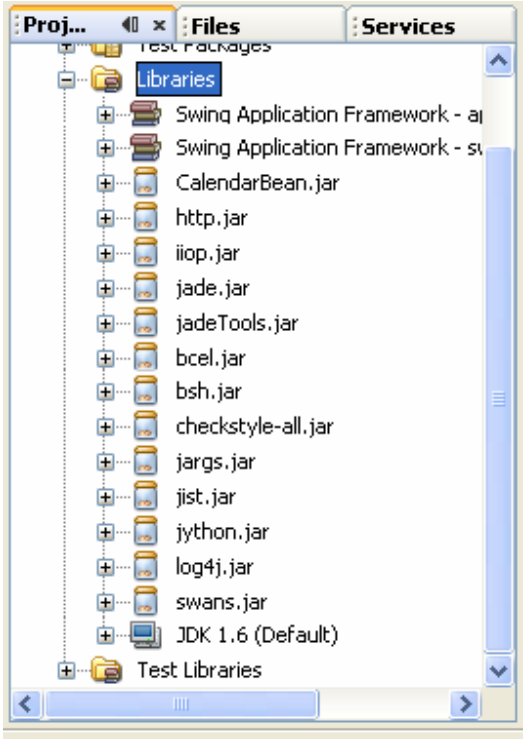

Figure 5.6 : Intégrations des bibliothèques.

5.4 Simulation et résultat

Dans cette section nous présentons un exemple de simulation avec les résultats obtenus

5.4.1 Scénario de simulation

Le scénario proposé est similaire à ce de " *Constant Bit Rate (CBR)* " l'un des programmes fournis avec les exemples de JIST/SWANS avec quelques modifications qui nous aident à montrer l'efficacité de notre approche. Ce programme crée un champ (field) de dimension x * y et place les nœuds dans différents lieux de façon aléatoire. Certains de ces nœuds sont désignés comme des paires de client-serveur, ces paires sont capables de transmettre un nombre N des paquets les uns aux autres pendant la durée de la simulation.

Il est possible de préciser, entre autres, le protocole de routage utilisé (nous choisissons ZRP), la probabilité de perte de paquets et le taux de mouvement des nœuds.

Figure 5.7 : Aperçus sur la page de démarrage de notre application.

Comme il est illustré dans la Figure 5.8, nous avons lancé une simulation de 30 minutes sous le protocole ZRP, le modèle de mobilité statique, un champ de simulation de 300x300 avec 100 nœuds placés aléatoirement sur ce champ et un nombre des clients/serveur fixé à 10 ou chacun des clients envoyer 50 messages au serveur correspondant. Cette simulation déroule deux fois avec et sans le système multi agents.

Les statistiques sur les paquets IARP perdus et le nombre des messages NDP sont enlevées chaque minute de simulation.

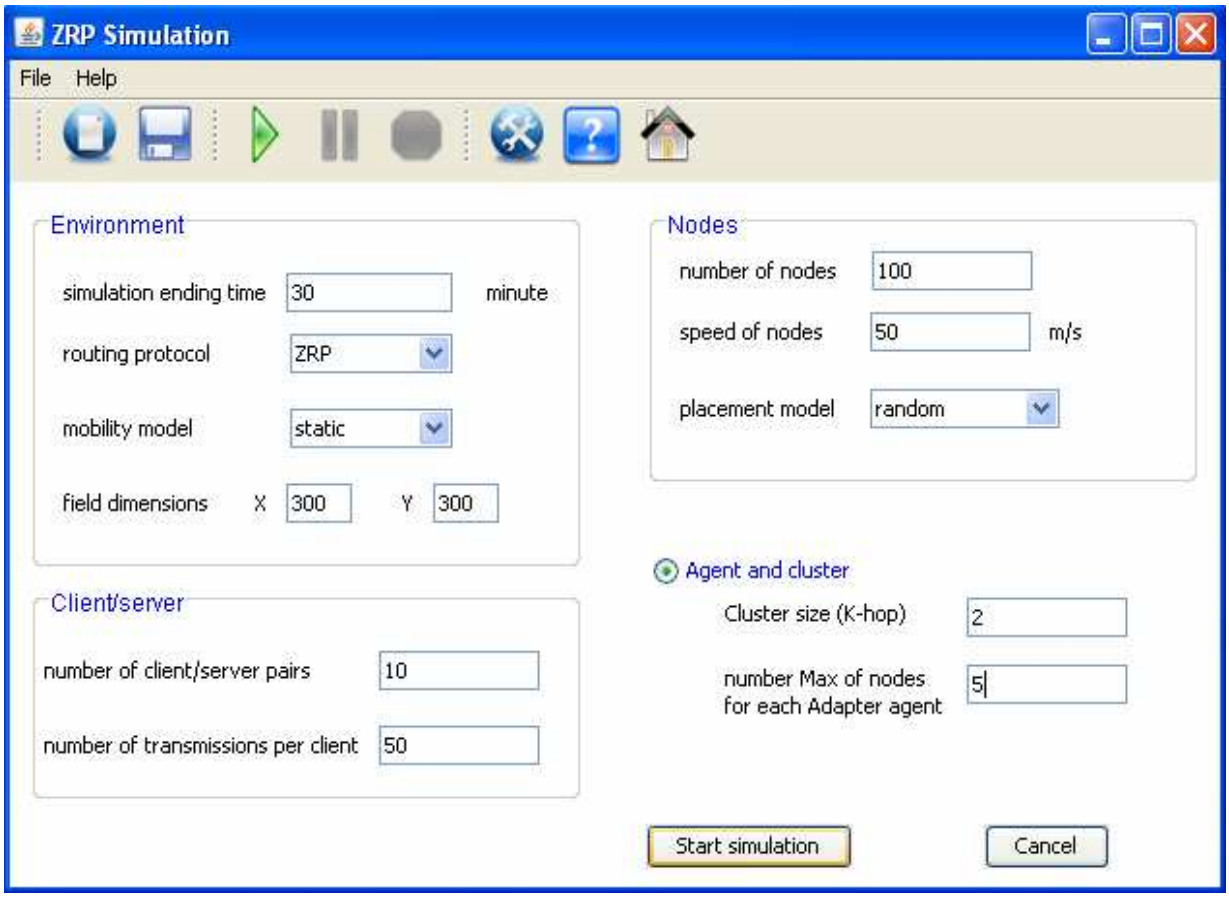

Figure 5.8 : Lancement d'un exemple de simulation.

Les résultats obtenus après la fin de la simulation sont enregistrés sous forme d'un fichier texte à la structure suivante :

Time(m)	IARP lost paquet	Number of NDP message
0	0	0
1	700	425

5.4.2 Discutions sur les résultats

Nous faisons une comparaison entre les performances de ZRP et le ZRP amélioré avec notre approche.

- **Nombre des paquets IARP perdus :**

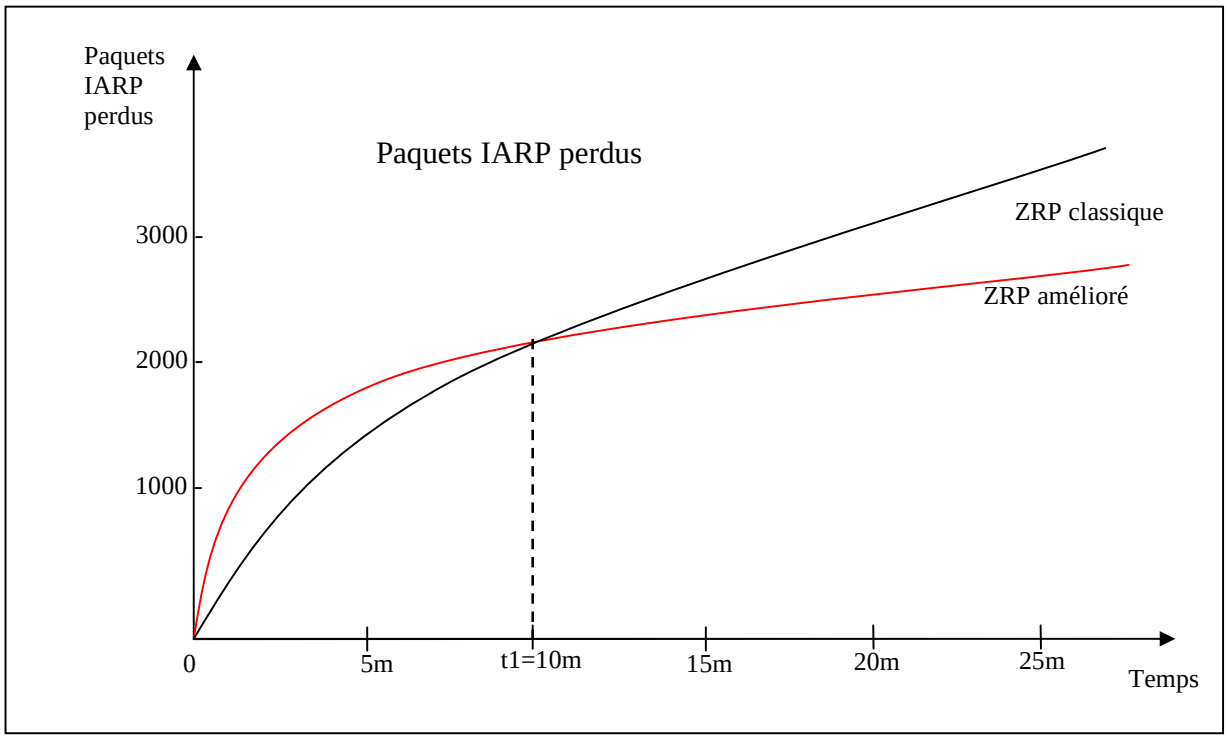

Figure 5.9 : Courbes montrons le nombre des paquets IARP perdus.

La figure 5.9 ci-dessus montre le nombre des paquets IARP perdus obtenus à partir d'une simulation de 100 nœuds à vitesse de 50 m/s. On désigne par paquets IARP perdus les messages de contrôle non acquis ou des données qui n'arrivent pas à la destination.

L'analyse et l'explication des courbes peut être comme suit :

On peut diviser la courbe en deux parties, dans la première partie (0 à t1=10m) on remarque que les deux courbes augmentent relativement avec le temps et de façons croissante forte, le fonctionnement de ZRP classique est mieux que ZRP amélioré.

En justifions ce point pour le ZRP classique par la convergence depuis l'état initiale (table de routage vide) vers l'état de réseaux à l'instant t1.

Pour le ZRP amélioré, en justifions cette augmentation, en plus de la convergence de réseau, par la stabilité partielle de système multi agent, ce dernier n'arrive pas encore à atteindre un bon degré de stabilité, pendant cette période, il est claire que le système multi agents est en

train de la création et que le nombre des agents n'est pas très approprié avec le nombre des nœuds et que les clusters ne sont pas complètement conçus ce qui implique plus de messages est automatiquement plus des paquets perdus.

Dans la deuxième partie, après l'instant t1 en remarque que ZRP amélioré devient mieux que ZRP classique ce qui prouve que les performances de protocole ZRP sont améliorés avec notre approche après la période de stabilité, le changement dans le réseau est de plus en plus contrôler est l'échange des messages est en fonction de la mobilité des nœuds.

- **Nombre des messages NDP échangés :**

Les messages NDP sont les messages qui jouent le rôle des messages Hello pour la détection des voisins. Les courbes suivantes (Figure 5.10, Figure 5.11) montrent le nombre de messages NDP échangés dans la même simulation précédente (100 nœuds à vitesse de 50 m/s puis avec 100m/s).

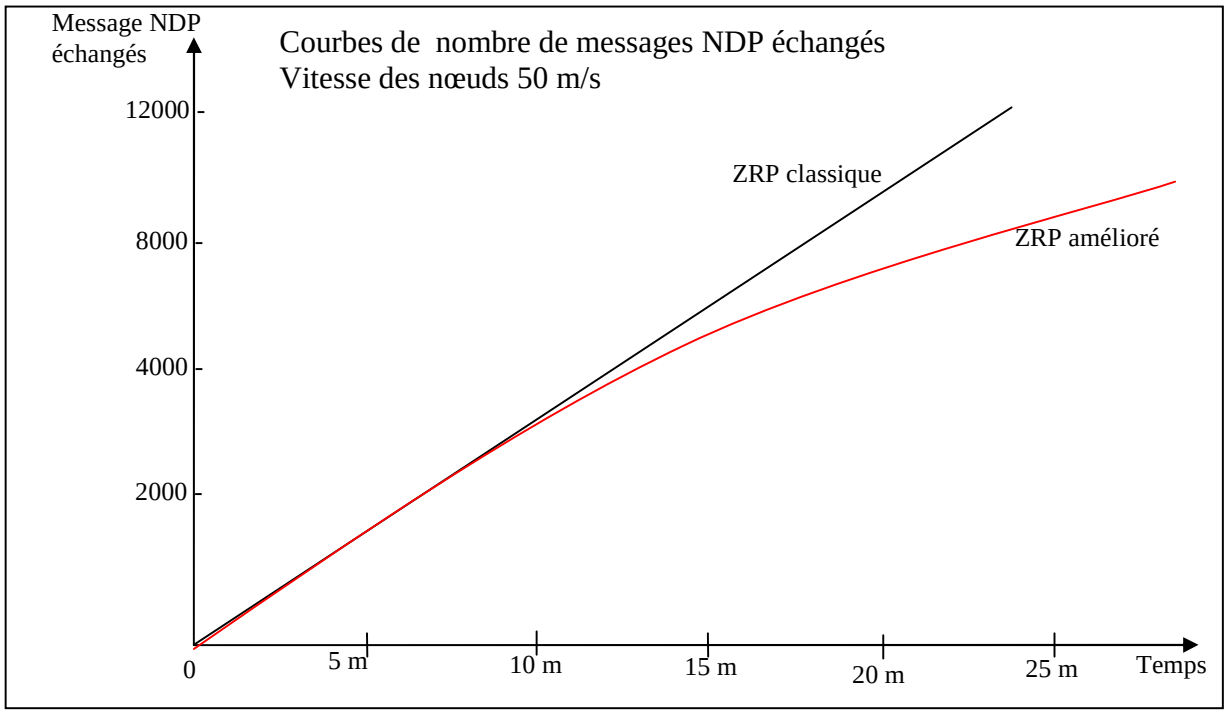

Figure 5.10 : Nombre de messages NDP échangés pour une vitesse de 50 m/s

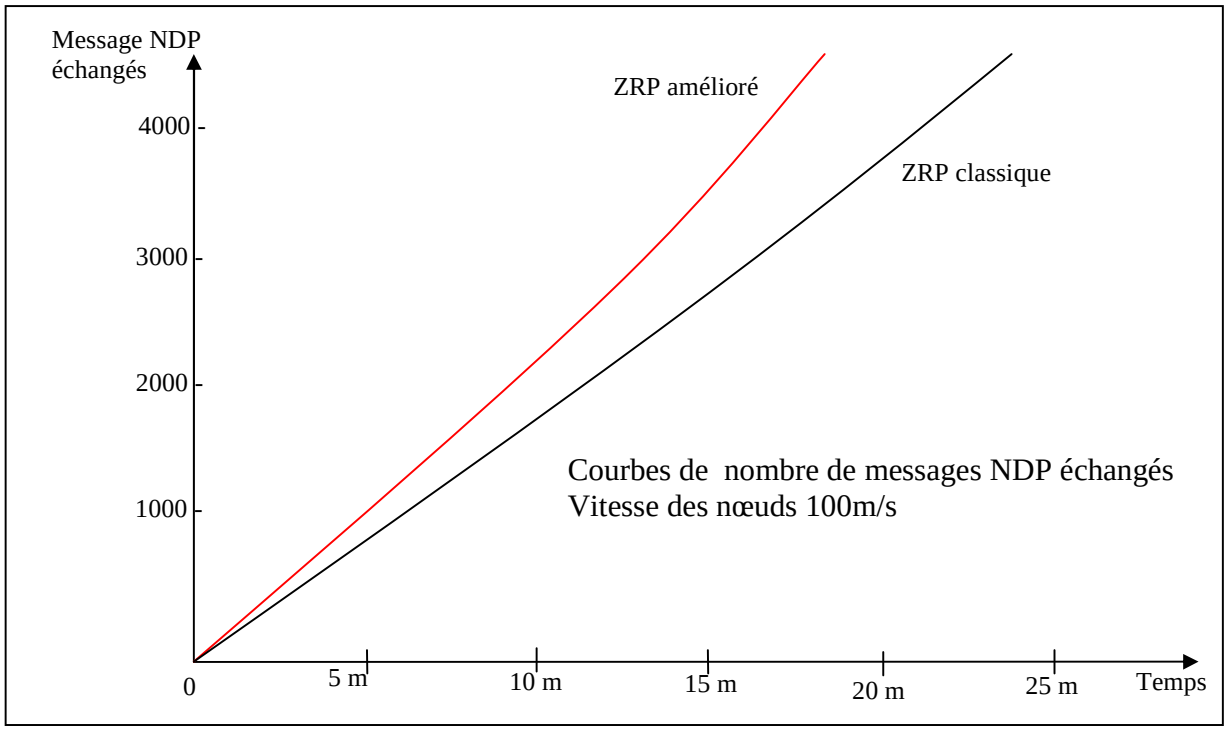

Figure 5.11 : Nombre de messages NDP échangés pour une vitesse de 100 m/s.

Lorsque l'analyse des résultats obtenus dans les deux cas (50 ou 100 m/s) sur les courbes si dessus, la première remarque c'est la forme de la courbe qui est, à peu près, linaire pour le ZRP classique. La période d'envoi des messages est régulière et le nombre de messages NDP augmente exponentiellement avec le temps.

Pour le ZRP amélioré, on remarque que lorsque la vitesse est petite (50m/s) le nombre de messages NDP a diminué par rapport au ZRP classique. Par contre, lorsque la vitesse est grande le nombre de messages NDP a augmenté. Ce point est justifié par le choix de notre politique de varier l'intervalle des messages NDP pour la détection des voisins en fonction de la mobilité des nœuds.

Une petite vitesse correspond à un grand intervalle de temps ce qui implique un nombre plus petit des messages NDP.

Une grande vitesse correspond à un petit intervalle de temps ce qui implique un nombre plus grand des messages NDP.

5.5 Conclusion

Dans ce chapitre, nous appliquons notre approche sur un exemple consiste en un ensemble des clients/serveurs qui échangent des messages entre eux au sein d'un réseau ad hoc. Nous choisissons la partie proactive de protocole ZRP (IARP) et le protocole NDP déployé par ZRP pour la détection des voisins.

Les résultats obtenus montrent que notre approche a des avantages comme la réduction de nombre des paquets perdus et assure une connexion plus efficace et fiable. D'un autre côté, notre approche assure une adaptation à la mobilité des nœuds par l'augmentation ou la réduction de l'intervalle de l'échange des messages de contrôle en fonction de cette mobilité. ☒

L'inconvénient est illustré lorsque la vitesse est très grande. Le nombre de messages de contrôle augmente ce qui consomme de l'énergie en plus de temps de processeurs.

Conclusion générale

La mobilité est une caractéristique inhérente de l'informatique moderne, lorsqu'il s'agit de réseaux sans fil ad hoc, il apparaît clairement que la mobilité doit être prise en charge par les protocoles de routage. Ces derniers doivent s'adapter à l'état de l'environnement.

Dans ce mémoire, nous nous sommes intéressés à la problématique de l'utilisation des agents mobiles pour l'adaptation dynamique au changement de l'environnement d'un réseau ad hoc, ou plus précisément, le protocole déployé dans ce réseau.

Tout au long de ce mémoire, nous avons présenté les différentes technologies nécessaires pour la réalisation de notre proposition. Dans un premier temps, nous avons exposé le concept d'adaptation d'un réseau ad hoc au niveau protocolaire. L'énergie, le débit des liens, la qualité de service et la topologie sont les principales caractéristiques sur lesquelles les travaux de recherche proposant d'améliorer les performances des protocoles de routage.

Pour s'adapter dynamiquement à leur environnement, les protocoles de routage font des mesures à l'aide des métriques. Une métrique est une mesure indiquant l'état d'un noeud, de son voisinage, ou bien de l'ensemble du réseau. Les valeurs de ces métriques permettent aux protocoles de routage soit de changer la manière de leur fonctionnement ou mode (exp : de proactif vers réactif, changer l'algorithme de sélection de chemin selon les exigences de la qualité de service…etc.) soit de changer ses paramètres (exp : augmenté la fenêtre d'anticipation si le réseau est stable…etc.).

L'avantage de l'architecture proposée est qu'elle utilise les agents mobiles comme outil de perception de l'environnement. La plus part des chercheurs travaillons dans ce contexte utilisent les agents mobiles pour une implémentation intégrale d'un protocole, mais cette solution n'est pas vraiment une voie pour résoudre le problème. Chaque approche possède des avantages et des inconvénients.

Pour l'évaluation et l'étude de performance, nous avons appliqué notre approche sur un exemple de simulation (communication par message) entre un ensemble des clients/serveurs au sein d'un réseau ad hoc. Nous avons choisi la partie proactive de protocole ZRP (IARP) et le protocole NDP déployé par ZRP pour la détection des voisins.

Les résultats obtenus montrent que notre approche a des avantages comme la réduction de nombre des paquets perdus et l'assurance d'une connexion plus efficace et fiable. Dans un autre côté, notre approche assure une adaptation à la mobilité des nœuds par l'augmentation et la réduction de l'intervalle de l'échange des messages de contrôle.

Les inconvénients de notre approche sont la consommation d'énergie et de temps de processeur lorsque la vitesse est très grande et le nombre de messages de contrôle augmente.

Ce mémoire constitue une base de travail à partir du quelle, des nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté. Les perspectives que nous proposons peuvent donc s'orienter vers les directions suivantes :

- Etendre l'application de notre approche vers les protocoles de routage réactifs.
- La proposition des nouveaux métriques afin d'améliorer la qualité de perception de l'environnement.
- Intégration des autres facteurs pour l'adaptation de l'énergie, le débit des liens, la qualité de service et la topologie.

Bibliographie

[1] R. E. Kahn, “The organization of computer resources into a packet radio network,” IEEE Trans. Commun., vol. COM-25, no. 1, pp. 169-178, Jan. 1977.

[2] T. Clausen and P. Jacquet, “Optimized link state routing protocol (olsr),” RFC3626, October 2003.

[3] R. Ogier, F. Templin, and M. Lewis, “Topology Dissemination Based on Reverse-Path Forwarding (tbrf),” Network Working Group, Request for Comments : 3684.
http ://www.ietf.org/rfc/rfc3684.txt, February 2004.

[4] C.E. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers,” Comp. Comm. Rev., Oct. 1994, pp.234-244.

[5] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (aodv) routing,” RFC 3561, July 2003.

[6] D. Johnson, D. Maltz, and Y. Hu, “The dynamic source routing protocol for mobile ad hoc networks (dsr),” Internet-Draft, 16 April 2003.

[7] ZYGMUNT J. HAAS, MARC R. PEARLMAN, PRINCE SAMAR,” The Zone Routing Protocol (ZRP) for Ad Hoc Networks,” Internet Draft.
http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt

[8] M. Joa-Ng, I.-T. Lu, “ A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks,” IEEE Journal on Selected Areas in Communications 17 (8) (1999) 1415–1425.

[9] Cholatip YAWUT, “Adaptation à la mobilité dans les réseaux ad hoc,“ thèse du doctorat, Institut de Recherche en Informatique de Toulouse (IRIT), Novembre 2009

[10] V. T. Raisinghani and S. Iyer., “ Cross-layer design optimizations in wireless protocol stacks,” Computer Communications, vol. 27(8), pp. 720_725, May 2004.

[11] V. T. Raisinghani and S. Lyer., “ Eclair : An efficient cross layer architecture for wireless protocol stacks”, 5th World Wireless Congress, San Francisco, USA, May 25-28 2004., May 2004.

[12] W. Su, S. Lee, and M. Gerla, “Mobility prediction in wireless networks,” IEEE Military Communication Conference (MILCOM), vol. 1, 2000, pp.491–495.

[13] C. Gentile, J. Haerri, and R. E. V. Dyck, “Kinetic minimum-power routing and clustering in mobile ad-hoc networks,” Proceedings of the IEEE Vehicular Technology Conference (VTC’02 Fall) Conference, 2002.

[14] A. Agarwal and S. R. Das, “ Dead reckoning for mobile ad hoc networks ” Proceedings of the 2003 IEEE Wireless Communications and Networking Conference (WCNC’03), 2003.

[15] N.Wang and S. Chang, “A reliable on-demand routing protocol for mobile ad hoc networks with mobility prediction,” Elsevier Computer Communications, vol. 29, pp. 123-135, 2005.

[16] H. Menouar, M. Lenardi, and F. Filali, “A movement prediction-based routing protocol for vehicle-to-vehicle communications,” Proceedings of the 1st International Vehicle-to-Vehicle Communications Workshop (V2V-COM’05), 2005.

[17] F. Bai, N. Sadagopan, and A. Helmy, “Important : a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks,” INFOCOM 2003, April 2003.

[18] M. Zonoozi and P. Dassanayake, “User Mobility Modeling and Characterization of Mobility Patterns,” IEEE Journal on Selected Areas on Communication, vol. 15, no.7, pp: 1239-1252, September 1997.

[19] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa, “Stochastic Properties of the Random Waypoint Mobility Model: Epoch Length, Direction Distribution, and Cell Change Rate,” ACM MSWiM’02, 2002.

[20] E. Royer, P.M. Melliar-Smith, and L. Moser, “An Analysis of the Optimum Node Density for Ad hoc Mobile Networks,” Proceedings of IEEE ICC, 2001.

[21] L. Blazevic, S. Giordano, and J.Y. Le Boudec, “Self Organized Terminode Routing,” Journal of Cluster Computing, Vol. 5, No.2, pp. 205–18, 2002.

[22] R. Groenevelt · E. Altman · P. Nain, “Relaying in mobile ad hoc networks: The Brownian motion mobility model,”, Wireless Networks, Volume 12, Number 5 / octobre 2006.

[23] ETSI, Universal Mobile Telecommunications System (UMTS), “Selection Procedures for the Choice of Radio Transmission Technologies of the UMTS,” Technical Report, TR 101 112 v 3.2.0 (1998).

[24] Z. Haas. “A new routing protocol for the reconfigurable wireless networks,” Proceedings of the Institute of Electrical and Electronics Engineers (IEEE) Inter-national Conference on Universal Personal Communications (ICUPC), pages 562–565, October 1997.

[25] B. Liang and Z. Haas, “Predictive Distance-Based Mobility Management for PCS networks,” Proceedings of IEEE INFOCOM, March 1999.

[26] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” Wireless Communications and Mobile Computing (WCMC), vol. 2, no. 5, no. 5, pp. 483–502, 2002.

[27] V. Davies, “Evaluating Mobility Models within an Ad Hoc Network,” MS thesis, Colorado School of Mines, 2000.

[28] [JAR 03] JARDOSH A., BELDING-ROYER E. M., ALMEROTH K. C., Suri S. “ Towards Realistic Mobility Models for Mobile Ad hoc Networks” Proceedings of MobiCom ’03, 2003.

[29] M. Sanchez and P. Manzoni, “Anejos: A Java-based Simulator for Ad-hoc Networks,” Future Generation Computer Systems - Elsevier Science, vol. 17, no. 5, pp. 573-583, 2001.

[30] X. Hong, M. Gerla, G. Pei, and C. Chiang, “A group mobility model for ad hoc wireless networks,” proceedings of the ACM International workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM), August 1999.

[31] J. Ferber , “ Les systèmes multi-agents - Vers une intelligence collective, ” InterEditions, 1995.

[32] Jeff Rulifson, “ RFC5 - Decode Encode Language (DEL), ”, rfc-5, June 2, l969.

[33] D.S. Milojicic, F. Douglis, Y. Paindaveine, R.Weeler, and S. Zhou, “ Process migration, ”, ACM Compting Surveys, 32(3) : 241–299, Septembre 2000.

[34] J. E. White, “ Telescript technology : The foundation for the electronic marketplace, ” White paper, General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040, 1994.

[35] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna, “ Understanding Code Mobility, ”, IEEE Transactions on Software Engineering, 24(5) :342–361, May 1998.

[36] G. Bernard et L. Ismail, “ Apport des agents mobiles à l’exécution répartie, ”, Revue des sciences et technologies de l’information, série Techniques et science informatiques, 21(6):771–796, 2002.

[37] Sara Bouchenak, “ Mobilité et Persistance des Applications dans l’Environnement Java,” thèse de doctorat de l’institut national polytechnique de GRENOBLE, 19 Octobre 2001.

[38] Salah El Falou et François Bourdon, “ Agent mobile et recherche d'information sur le Web : une solution basée sur le MDP, ” Reconnaissance des Formes et Intelligence Artificielle (RFIA06), 2006.

[39] Christophe CUBAT DIT CROS, “ Agents Mobiles Coopérants pour les Environnements Dynamiques,” thèse de doctorat de l’institut national polytechnique de Toulouse, 2 décembre 2005.

[40] Peter Braun and Wilhelm Rossak, “ Mobile Agents Basic Concepts, Mobility Models, and the Tracy Toolkit, ” livre , Morgan Kaufmann Publishers is an imprint of Elsevier, 500 Sansome Street, Suite 400, San Francisco, CA 94111, édition 2005.

[41] Robert S. Gray, George Cybenko, David Kotz, and Daniela Rus, “Agent Tcl. Itinerant Agents: Explanations and Examples with CDROM, ” William Cockayne and Michael Zypa (editors), Manning Publishing and Prentice Hall, 1997.

[42] D. B. Lange et O. Mitsuru, “Programming and Deploying Java Mobile Agents Aglets,” Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[43] Systems Laboratory of Mitsubishi, “Concordia: An Infrastructure for Collaborating Mobile Agents,” Mitsubishi Electric ITA Horizon Systems Laboratory 1432 Main Street Waltham, MA 02154, USA.

[44] J. Baiumann, F. Hohl, K. Rothermel, M. Straer, ”Mole, Concepts of a Mobile Agents System,” World Wide Web, vol. 1, No. 3, pp. 123-137, 1998.

[45] Anurag Acharya , M. Ranganathan , Joel H. Saltz, “Sumatra: A Language for Resource-Aware Mobile Programs,” Selected Presentations and Invited Papers Second International Workshop on Mobile Object Systems - Towards the Programmable Internet, p.111-130, July 08-09, 1996.

[46] G. Glass, ”Object Space Voyager - The Agent ORB for Java”, Lecture Notes in Computer Science, no.1368, pp. 38-55, 1998.

[47] N. Kawaguchia,*, K. Toyamaa,b, Y. Inagakia , “MAGNET: ad hoc network system based on mobile agents,” A Department of Information Engineering, Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan.
Computer Communications 23 (2000) 761–768.

[48] Kaizar A. Amin *, Armin R. Mikler, “Agent-based distance vector routing: a resource efficient and scalable approach to routing in large communication,” Department of Computer Sciences, University of North Texas, 320 General Academic Building Mulberry, Avenue B, Denton, TX 76203-1366, USA. The Journal of Systems and Software 71 (2004) 215–227.

[49] S.S. Manvi *, M.S. Kakkasageri, “Multicast routing in mobile ad hoc networks by using a multiagent system,” Department of Electronics and Communication Engineering, Basaveshwar Engineering College, Bagalkot-587 102, India Information Sciences 178 (2008) 1611–1628 .

[50] G. Di Caro, M. Dorigo, “*Mobile Agents for Adaptive Routing*,” Technical Report, IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.

[51] Piotr Kosiuczenko : « Sequence Diagrams for Mobility » Institute of Computer Science, Ludwig-Maximilian-University Oettingenstr. 67, Munich Germany 2001.

[52] Agent UML - AUML Home Page. Available at http://www.auml.org.

[53] R. Barr and Z. J. Haas : JiST/SWANS. http://jist.ece.cornell.edu/, March 2005.

[54] Rimon Barr. JiST– Java in Simulation Time User Guide, http://jist.ece.cornell.edu/jist-user/node3.html.

[55] R. Barr and Z. J. Haas : SWANS Java Documentation. Septembre 2006. http://jist.ece.cornell.edu/javadoc/index.html.

[56] X. Zeng, R. L. Bagrodia, and M. Gerla. "GloMoSim: a library for parallel simulation of large-scale wireless networks." In PADS, May 1998.
Site officiel de GloMoSim : http://pcl.cs.ucla.edu/projects/glomosim/.

[57] Fall, K., Varadhan, K. (2001), "The NS Manual," UC Berkeley, LBL, USC/ISI, and Xerox PARC.
Site officiel de ns2 : http://www.isi.edu/nsnam/ns/.

[58] Rimon Barr , " SWANS– Scalable Wireless Ad hoc Network Simulator User Guide", http://jist.ece.cornell.edu/docs/040319-swans-user.pdf.

[59] JADE (Java Agent DEvelopement framework) : http://jade.tilab.com.

[60] FIPA, IEEE Foundation for Intelligent Physical Agents, http://www.fipa.org/ ,2000.

[61] NetBeans IDE : www.netbeans.org/.

[62] Sun Microsystems : http://fr.sun.com/.