République Algérienne Démocratique et populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université M'hamed BOUGARA de BOUMERDES

mémoire

Faculté des Sciences

Département d'informatique

MEMOIRE DE MAGISTER

Spécialité : système informatique et génie des logiciels

Option : spécification de logiciel et traitement de l'information

Ecole doctorale

Présenté par :

RIAHLA Med Amine

Thème

Conception et mise en œuvre d'un nouveau protocole de routage Multi chemins sécurisé pour les réseaux ad hoc basé sur les colonies de fourmis

Soutenu le : 30/06/2008, devant le jury de soutenance composé de :

Mr. M. MEZGHICHE	UMBB	Professeur	Président
Mr. K. TAMINE	Université de LIMOGE	Maître de conférences	Promoteur
Mr. Y. DJOUADI	UMMTO	Maître de conférences	Examinateur
Mr. J. DJENOURI	CERIST	Maître de recherches	Examinateur

Année Universitaire : 2007/2008

Dédicaces

A la mémoire de mon collègue de magistère MAGRAMANE sofiane

A la mémoire de mes grands-parents paternels

A mes grands-parents maternels

A celle qui m'est plus chère au monde, qui m'a soutenu durant toute ma vie grâce à son amour et son affection, que Dieu te bénisse maman

A mon très chère père qui grâce à ses sacrifices, je suis devenu ce que je suis aujourd'hui

A mes chers frères Yacine, Samir, Smail, Mounir et à ma chère sœur Amel

A mes oncles et mes tantes

A tous mes cousins et cousines

A tous mes amis

khelifa, Hocine et nabil

A toute l'Equipe de biunet

Je dédie ce mémoire.

Remerciements

Je tiens à remercier, Monsieur TAMINE KARIM, pour ses conseils judicieux, sa grande disponibilité et les précieuses discussions que nous avons eues ensemble. Je lui exprime ma profonde gratitude pour m'avoir fait profiter de ses connaissances, mais aussi de ses méthodes de travail. Grâce à lui, j'ai découvert un domaine de recherche qui aujourd'hui me passionne.

J'adresse mes plus vifs remerciements à Monsieur M. MEZGHICHE Professeur à l'UMBB, pour avoir accepté de présider ce jury et pour m'avoir accueilli dans son laboratoire LIFAB.

Je remercie également et pleinement Dr J.DJENOURI, maître de recherches au Centre de Recherche sur l'Information Scientifique et Technique (CERIST) et Monsieur Y.DJOUADI Maître de conférences à l'université de TIZI-OUZOU qui m'on fait l'honneur d'accepter de juger ce modeste travail.

Enfin je souhaite aussi témoigner toute mon amitié à l'ensemble de mes collègues de poste graduation.

A mes parents

Résumé

Les réseaux ad hoc sont composés d'unités mobiles communiquant via un média sans fil, sans la nécessité d'infrastructure physique. Dans ce genre de topologie, tous les nœuds coopèrent afin d'assurer la bonne gestion du réseau (contrôle, routage,…). La nature complètement distribuée de ce type de réseau pose le problème de performances (dûes aux calculs des routes) ainsi que les problèmes liées à la sécurité des échanges entre les nœuds. En ce qui Concerne les performances dues au routage des paquets, les protocoles actuels se divisent en deux catégories, les protocoles proactifs et les protocoles réactifs. Les protocoles proactifs établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage comme dans le cas des réseaux classiques, alors que les protocoles réactifs cherchent les routes à la demande. Les deux classes souffrent d'un même problème qui est le nombre important de message de contrôle qui influence négativement sur la bande passante du réseau et les ressources énergétique des nœuds. Une troisième catégorie de protocoles de routage, dits hybrides, combinant les deux approches (réactive et proactive) a aussi été développée. Parmi les protocoles appartenant à cette catégorie, nous distinguons des protocoles inspirés des colonies de fourmis qui présentent, dans la plupart des cas, de meilleurs résultats. Tous les protocoles existant qui se basent sur le principe des fourmis utilisent une diffusion de fourmis (agents) par les nœuds sources pour la découverte des routes, ce qui augmente de façon exponentielle le nombre de messages de contrôle inutile. En ce qui concerne l'aspect sécurité, De nombreux travaux se focalisent sur les comportements malveillants pour contrer les attaques actives en négligeant les comportements égoïstes des nœuds qui peuvent avoir des conséquences dramatiques dans le cas d'un réseau ad hoc; par exemple un hôte peut tout simplement rejeter les paquets en transit qu'il reçoit afin d'économiser son énergie. L'étude, qui a porté sur l'analyse des protocoles existants, nous a permis de faire ressortir un ensemble de problèmes; pour cela nous avons conçu un nouveau protocole de routage, le protocole AntTrust, afin d'apporter quelques solutions aux problèmes initiaux. Le travail consiste à formaliser et à mettre en œuvre un protocole de routage Multi chemins pour les réseaux ad hoc basé sur la réputation des nœuds. Ce protocole est basé sur le fonctionnement d'une colonie de fourmis pour le calcul dynamique des routes et sur un mécanisme de coopération pour contrer les mauvais comportements des nœuds. Les résultats obtenus par le simulateur ns2, comparés aux autres protocoles, encouragent et montrent la validité de notre protocole.

Abstract

The ad hoc networks are composed of mobile units communicating via a wireless media, without the need for physical infrastructure. In this kind of topology, all the nodes cooperate to ensure the proper management of the network (control, routing,…).The fully distributed nature of this type of network poses the problem of performance (due to the calculations of roads) as well as problems related to the security of exchanges between the nodes. As Concerns performance due to routing packets, existing protocols are divided into two categories, proactive protocols and reactive protocols. The protocols establish proactive roads in advance based on the periodic exchange of routing tables as in the case of traditional networks, while the reactive protocols search roads at request. Both classes suffer from the same problem which is the large number of control messages which influence negatively on network bandwidth and energy resources nodes. A third category of routing protocols, known as hybrids, combining the two approaches (reactive and proactive) has also been developed. Among protocols there posters this category, we distinguish protocols inspired by the colonies of ants which introduce, in most cases, the best results. All existing protocols which are based on the principle of ants use a broadcasting of ants (agents) by nodes sources for the discovery of roads, which exponentially increases the number of control messages unnecessary. Regarding the security aspect, Numerous jobs are focused on malicious behaviours to counter the active attacks by neglecting selfish behaviours of the nodes which can have dramatic consequences in while network ad hoc; for example, a host may simply reject the package in transit it receives in order to save his energy. The study, which focused on analysis of existing protocols, allowed us to bring to light group of problems; why we designed a new routing protocol, the protocol AntTrust to provide solutions to some initial problems. The job is to formalize and implement a protocol routing Multi paths for ad hoc networks based on the reputation of nodes. This protocol is based on how a colony of ants for calculating dynamic roads and a cooperative mechanism to counter the bad behavior of nodes. The results obtained by the simulator ns2, compared to other protocols, promote and demonstrate the validity of our protocol.

ملخص

تتكون شبكات Ad Hoc من وحدات متنقلة تتصل عبر وسائط لاسلكية، من دون الحاجة إلى البنية التحتية المادية. في مثل هذا النوع من الطوبولوجيا ، كل العقد تتعاون لضمان الإدارة السليمة للشبكة (الرقابة ، التوجيه ،...).' طبيعة التوزيع الكامل لهذا النوع من الشبكة يطرح مشكلة الأداء (نظرا لحسابات الطرق)، بالإضافة إلى المشاكل المتعلقة بأمن تبادل المعلومات بين العقد. فيما يتعلق بالأداء الراجع لتوجيه الرزم، البروتوكولات القائمة تنقسم إلى فئتين، البروتوكولات المبادرة وبروتوكولات رد الفعل. البروتوكولات المبادرة تنشئ الطرق مقدما على أساس تبادل دوري لجدول المسالك كما هو الحال في الشبكات التقليدية، في حين أن بروتوكولات رد الفعل تبحث عن الطرق حسب الطلب. الفئتين تعانيان من نفس المشكلة التي هي العدد الكبير من رسائل التوجيه التي تؤثر سلبا على عرض النطاق الترددي للشبكة و على الموارد الطاقوية للعقد. فئة ثالثة من بروتوكولات التوجيه ، المعروفة باسم الهجينة ، تجمع بين الفئتين (رد الفعل والمبادرة) .من بين البروتوكولات التي تنتمي لهذه الفئة نميز البروتوكولات المستوحاة من مستعمرات النمل الذي يعرض، وفي معظم الحالات، أفضل النتائج. جميع البروتوكولات القائمة والتي تقوم على مبدأ النمل تستخدم نشر النمل (وكلاء) من عقد المصدر لاكتشاف الطرق، مما يزيد بصفة مضاعفة عدد رسائل التوجيه الغير ضرورية. وفيما يتعلق بالجانب الأمني ، العديد من الأعمال تركز على التصرف السيئ للتصدي للهجمات مع إهمال السلوكيات الأنانية من العقد التي يمكن أن يكون لها عواقب مأساويه في حالة شبكة Ad Hoc ؛ على سبيل المثال ، يمكن ببساطة للمضيف أن يرفض الرزم التي يتلقاها من اجل اقتصاد طاقته. الدراسة، التي ركزت على تحليل البروتوكولات الموجودة ، سمحت لنا بإلقاء الضوء على مجموعة من المشاكل؛ لهذا السبب صممنا بروتوكول جديد للتوجيه ، البروتوكول AntTrust من اجل توفير حلول لبعض المشاكل الأولية. العمل يستلزم صياغة وتنفيذ بروتوكول للتوجيه متعدد المسارات لشبكات Ad Hoc القائمة على سمعة العقد. هذا البروتوكول مبني على أساس كيفية عمل مستعمرة النمل لحساب فعال للطرق والية التعاون لمكافحة التصرفات السيئة للعقد . النتائج التي تم الحصول عليها من قبل جهاز المحاكاة NS2، بالمقارنة مع غيره من البروتوكولات،تعزز و تثبت صحة البروتوكول.

Sommaire

Table des matières

II Contributions : AntTrust : Un protocole de routage Multichemins basé sur la réputation des noeuds

---

Introduction Générale

Les réseaux ad hoc sont composés d'unités mobiles communiquant via un média sans fil, sans la nécessité d'infrastructure physique. Dans ce genre de topologie, tous les nœuds coopèrent afin d'assurer la bonne gestion du réseau (contrôle, routage,…). Les réseaux ad hoc sont idéals pour des applications liées à des opérations de secours (militaires, pompiers, tremblement de terre, etc..) ainsi que les missions d'exploration.

La nature complètement distribuée de ce type de réseau pose le problème de performances (dues aux calculs des routes) ainsi que les problèmes liées à la sécurité des échanges entre les nœuds.

En ce qui Concerne l'aspect performances, les protocoles actuels se divisent en deux catégories, les protocoles proactifs et les protocoles réactifs. Les protocoles proactifs établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage comme dans le cas des réseaux classiques, alors que les protocoles réactifs cherchent les routes à la demande. Les protocoles réactifs soufrent de coûts élevés lors de l'établissement des routes. Ceci provient en grande partie du caractère massif des inondations des paquets de recherche de route. Le trafic généré pour le contrôle et la mise à jour de la table de routage dans les protocoles proactifs peut être important et partiellement inutile et le coût en bande passante est important.

Une troisième catégorie de protocoles de routage, dits hybrides, combinant les deux approches (réactive et proactive) a aussi été développée. Parmi les protocoles appartenant à cette catégorie, nous distinguons des protocoles inspirés des colonies de fourmis qui présentent, dans la plupart des cas, de meilleurs résultats. Tous les protocoles existant qui se basent sur le principe des fourmis utilisent une diffusion de fourmis (agents) par les nœuds sources pour la découverte des routes, ce qui augmente de façon exponentielle le nombre de messages de contrôle inutile.

En ce qui concerne l'aspect sécurité, l'essentiel des propositions de protocoles de routage sécurisés présupposent une phase de distribution de clefs pour protéger le routage. Cependant, ces travaux reposent implicitement sur une infrastructure de sécurité, ce qui est contradictoire avec la nature même d'un réseau ad hoc.

De nombreux travaux se focalisent sur les comportements malveillants pour contrer les attaques actives en négligeant les comportements égoïstes des nœuds qui peuvent avoir des conséquences dramatiques dans le cas d'un réseau ad hoc. Il est donc nécessaire de traiter la sécurité dans les réseaux ad hoc de manière plus globale et en tenant compte des spécificités de tels réseaux. En particulier, un protocole de routage ad hoc doit s'appuyer sur un modèle de confiance réaliste et doit intégrer des mécanismes contrant les attaques actives, forçant la coopération entre les nœuds, et détectant les comportements défaillants.

Ces travaux se heurtent souvent à la même difficulté qui est de proposer des mécanismes relativement robustes face aux différentes attaques possibles, sans pour autant affecter les performances du réseau ad hoc de manière trop prononcée. En effet beaucoup de propositions s'appuient sur un protocole de routage ad hoc existant; puis pour assurer la sécurité des échanges on y ajoute à posteriori des mécanismes de sécurité. Cependant, dans un but de simplification, les auteurs de ces travaux sont amenés à supprimer des optimisations présentes dans le protocole initial au détriment des performances du protocole résultant.

La première partie de mon travail consistait à faire un état de l'art des différents protocoles de routage pour les réseaux ad hoc, à étudier les performances de ces différents protocoles et enfin à étudier les notions et les méthodes qui ont été utilisées pour sécuriser les échanges dans les réseaux Ad hoc.

Dans la deuxième partie de ce mémoire, je présente mes contributions : La proposition et la mise en œuvre d'un protocole de routage Multichemins pour réseaux ad hoc basé sur la réputation des nœuds. Ce protocole est basé sur le fonctionnement d'une colonie de fourmis pour le calcul dynamique des routes. Le protocole permet à chaque nœud du réseau d'établir une ou plusieurs routes vers d'autres nœuds, soit de manière proactive en évitant les diffusions des messages de contrôle comme dans le cas des protocoles proactifs classiques ou, réactive mais en minimisant le nombre de messages de contrôle pour l'établissement des routes (diffusion de la demande de route) en proposant même une autre approche qui diffère de tous les protocoles existants (demande locale des routes).

Le protocole réalisé, va au delà du simple calcul des routes les plus courtes, en calculant la route la plus sûre. Afin de juger de la sûreté d'une route qui n'est rien d'autre qu'une succession de nœuds, le protocole vise à faire évoluer au cours du temps une relation entre un nœud et tous ses voisins à travers l'établissement d'une réputation mutuelle entre les nœuds. A cet effet, le protocole intègre un mécanisme permettant à chaque nœud d'avoir à tout moment une note de réputation de tous ses voisins, ceci permet à chaque nœud de décider de router ou non des messages vers ou à travers un voisin donné.

Ce mémoire a été organisé de la manière suivante :

Première partie : Les réseaux AdHoc

- Généralités sur les réseaux Ad-Hoc.
- Le routage dans les réseaux Ad-Hoc.
- La sécurité des réseaux Ad-Hoc.

Deuxième partie : Contributions : AntTrust - Un protocole de routage Multichemins basé sur la réputation des nœuds -

- Présentation du routage dans AntTrust.
- Présentation du modèle de réputation dans AntTrust.
- Mise en œuvre de AntTrust à l'aide du simulateur Ns2

Partie I : Etat de l'art

Routage et sécurité dans les réseaux Ad hoc

Chapitre I

Introduction aux réseaux ad hoc

La technologie des réseaux informatiques sans fil, en pleine expansion, suscite actuellement un grand intérêt vu les facilités qu'elle apporte : grande liberté de mouvement, économie d'installation en cout et en temps, interopérabilité et fusion avec le monde de la téléphonie mobile, etc.

Cette famille de réseaux se divise en deux classes de topologies :

. Les topologies avec infrastructure : elles sont constituées d'un ensemble de stations de bases fixes connectées par un réseau filaire. La zone de couverture de chaque station de base définit une cellule. Les hôtes mobiles communiquent entre eux via le réseau des stations de base. Le réseau GSM est un exemple typique des réseaux sans fil avec infrastructure. Les réseaux WLAN basés sur la norme IEEE802.11 sont un autre exemple plus récent de cette famille de réseaux.

. Les topologies sans infrastructure : ces réseaux sont constitués d'unités mobiles communiquant entre eux sans l'aide d'infrastructure fixe. Appelées communément ad hoc, elles ne nécessitent aucune structure physique (backbone) pour être déployées et sont opérationnelles instantanément. Dans ce type de réseaux, tous les hôtes doivent coopérer pour gérer les communications entre eux (routage, contrôle de l'accès au media, etc).

La rapidité et les coûts quasi nuls de déploiement, la flexibilité et le support de la mobilité sont les principaux avantages des réseaux ad hoc qui peuvent être utiles dans une large variété d'applications.

Dans ce chapitre, je vais présenter le concept des réseaux mobiles ad hoc, leurs caractéristiques et quelques exemples d'applications.

I.1. Concept

Les réseaux mobiles ad hoc ou Mobile Ad hoc Networks (MANet) consistent en une collection de terminaux (ou nœuds) capable de s'auto organiser, et qui communiquent les uns avec les autres sans le support d'une quelconque infrastructure de gestion prédéfinie. Du fait

du champ de transmission limité des nœuds, les MANets sont des réseaux pair à pair, multi sauts, qui s'appuient sur les nœuds intermédiaires en tant que relais pour acheminer les paquets. Ceci signifie que les nœuds jouent à la fois le rôle d'hôte et de routeur : ils sont d'une part responsables de l'émission et de la réception de leurs propres données, et ils assurent d'autre part la retransmission du trafic des autres nœuds.

Les applications des réseaux ad hoc représentent l'une des causes principales de l'importance de la sécurité dans de tels réseaux. A cause de leurs flexibilités, comme on va le voir par la suite, les réseaux ad hoc peuvent être déployés dans des scénarios critiques et de secours tels que les opérations militaires ou catastrophes naturelles, etc.

Pour préciser les domaines d'utilisation, nous donnons ci-après les principales caractéristiques des réseaux ad hoc.

I.2 Les caractéristiques des MANET

Un réseau ad hoc est donc un système autonome constitué de nœuds mobiles. Ces derniers communiquent avec leurs voisins par des liaisons sans fil point à point. Quand les zones d'émission/réception de deux nœuds en communication sont disjointes, les nœuds intermédiaires sont alors sollicités pour assurer le routage. A partir de cette définition générale nous présentons les caractéristiques principales qui différencient un réseau ad hoc d'un réseau doté d'une architecture fixe.

- Pour Les nœuds dans un réseau ad hoc
  - ✓ ***La mobilité de tous les nœuds*** est une caractéristique intrinsèque des MANET. Le déplacement des nœuds provoque des modifications aléatoires et non prédictibles de l'architecture du réseau.
  - ✓ ***L'équivalence des nœuds*** est une spécificité des MANET. Dans un réseau classique, il existe une distinction nette entre les nœuds terminaux (stations, hôtes) qui supportent les applications et les nœuds internes du réseau (routeurs), chargés de l'acheminement des données. Cette différence n'existe pas dans les réseaux ad hoc car tous les nœuds peuvent être amenés à assurer des fonctions de routage.

- ✓ ***Le nombre de nœuds mobiles*** présents dans un MANET varie selon les besoins. D'une façon plus générale aucune limitation n'est faite sur la taille ou le nombre de nœuds d'un réseau ad hoc.
- ✓ ***Les ressources énergétiques*** des nœuds mobiles, alimentés par des sources d'énergies autonomes (batteries) sont limitées tel que les Personal Digital Assistant (PDA) qui sont des équipements typiquement limités en énergie, en puissance de calcul et en capacité mémoire. Ces équipements intègrent des modes de gestion d'énergie et il est important que les protocoles mis en place dans les réseaux ad hoc prennent en compte cette caractéristique.

- L'absence de serveur centralisé rend complexe le contrôle et la gestion d'une architecture qui se forme et évolue au gré de l'apparition et des déplacements des nœuds. En conséquence, il n'existe aucune hiérarchie entre les nœuds et aucun service réseau ne peut prétendre être centralisé.
- Les liaisons physiques s'appuient sur *les technologies* de communications sans fil, indispensables à la mise en place d'un réseau ad hoc. Malgré des progrès très importants, leurs performances sont encore aujourd'hui en deçà de celles des technologies des réseaux LAN filaires.
- Les vulnérabilités des réseaux sans fil sont par nature plus sensibles aux problèmes de sécurité. Pour les réseaux ad hoc, le principal problème ne se situe pas tant au niveau du support physique seulement mais principalement dans le fait que tous les nœuds sont équivalents et potentiellement nécessaires au fonctionnement du réseau. Les possibilités de s'insérer dans le réseau sont plus grandes, la détection d'une intrusion ou d'un déni de service est plus délicate et l'absence de centralisation rend plus complexe la collecte d'informations pour la détection d'intrusions.

I.3 Applications

A l'origine, les MANets ont été étudiés pour les déploiements dans les environnements militaires et tactiques. Un exemple d'application est l'établissement de communications, indépendamment de toute infrastructure, entre les groupes armés sur un champ de bataille. Ces dernières années, les applications civiles ont commencé à tirer parties des caractéristiques de ce type de réseau.

Des scénarios plus complexes dans le domaine industriel appelés réseaux de capteurs (Sensor Networks) peuvent former un MANET pour s'adapter à différents environnements. Un exemple d'une telle application est la formation d'un MANet pour la surveillance médicale, les poursuites militaires, la détection des Feux de forêt, la surveillance des volcans et entre les véhicules mobiles sur un réseau routier, ceci dans le but de gérer et réguler le trafic.

De nombreuses autres situations de la vie courante sont adaptées à l'usage des MANET. Nous considérons, par exemple, un réseau créé pour les besoins et la durée d'une réunion regroupant des participants de différents organismes, ou un réseau créé entre les élèves et leur professeur dans une salle de classe pour la durée d'un cours.

Les caractéristiques des réseaux ad hoc soulèvent de nouvelles problématiques qui sont spécifiques à ce type de réseau; afin de satisfaire les besoins de toutes ces applications, de nouvelles fonctionnalités doivent être réalisées, plus particulièrement au niveau du routage des données et de la sécurité du routage.

I.4. Le routage dans les réseaux ad hoc

Les techniques de routage des réseaux classiques, basées sur des routes préétablies par des équipements spécialisés et dédiés, ne peuvent plus fonctionner correctement sur un réseau ad hoc. Pour cela les réseaux ad hoc utilisent des protocoles de routage spécifiques.

Dans le prochain chapitre, je vais présenter les principaux protocoles de routage des réseaux ad hoc en donnant des exemples détaillés pour chaque catégorie.

Chapitre II Les protocoles de routage dans les réseaux ad hoc

Un protocole de routage a pour fonction de déterminer le chemin entre deux nœuds en fonction d'une stratégie prédéfinie. Dans les réseaux multi-sauts[1], les fonctions fondamentales du routage servent à trouver et à maintenir les routes entre une source et une destination pour permettre les communications. Dans les réseaux filaires, le routage s'appuie généralement soit sur des algorithmes vecteurs de distance [26], soit sur des algorithmes à état de liens [44]. Ces deux algorithmes nécessitent la diffusion périodique par chaque routeur de paquets d'annonce (ou de contrôle) de routage. Dans le routage vecteur de distance, chaque routeur diffuse périodiquement à l'ensemble des routeurs sa vue logique des distances vers tous les autres nœuds en nombre de sauts; à partir de ces informations, les routeurs voisins calculent localement le chemin le plus court vers chaque nœud. Dans le routage état de liens, chaque routeur diffuse à ses routeurs voisins sa vue sur l'état de ses liens adjacents; à partir de ces informations, les routeurs calculent le chemin le moins coûteux (mais pas nécessairement le plus court).

Ces algorithmes de routage conventionnels ne sont pas adaptés aux environnements mobiles où des changements dans la topologie du réseau sont fréquents. En effet, un haut degré de mobilité des nœuds conduit à des calculs de routes et des échanges de paquets de contrôle fréquents, or cette convergence complète des routes stables cohérentes ne peut être fiable car elle est coûteuse en termes de consommation de la bande-passante et d'énergie et en puissance de calcul.

Une autre caractéristique des protocoles de routage dans les réseaux filaires est que les liens sont considérés comme étant bidirectionnels, ce qui signifie que les transmissions entre deux hôtes sont réciproques (si *A* peut communiquer avec *B,* alors *B* peut communiquer avec *A).* Or cette réciprocité n'est pas toujours vérifiée dans les réseaux sans-fil.

Une première question qui se pose dans le contexte des MANets est de déterminer s'il est nécessaire de maintenir des tables de routage. Dans une approche non-routée, des

[1] Le chemin entre les nœuds source et destination nécessite la présence de plusieurs nœuds intermédiaires

protocoles par inondation (ou flooding) où le principe est que chaque nœud retransmet les paquets qu'il reçoit; à cet effet des mécanismes complémentaires de contrôle peuvent être utilisés pour éviter les bouclages ou la duplication des paquets [10][11]. L'approche diamétralement opposée consiste à s'appuyer sur une couche de routage pour trouver et maintenir les routes entre une source et une destination.

Une autre question qui survient est de déterminer comment maintenir les routes : vaut-il mieux maintenir constamment la cohérence des tables par envoi périodique de paquets de contrôle alors que celles-ci changent sans-arrêt (du fait de la mobilité), ou vaut-il mieux construire les tables de routage au dernier moment? C'est à la base de ces deux questions que deux grandes classes de protocoles de routage dans les réseaux ad-hoc ont été définies. Nous distinguons les protocoles dits *proactifs* où chaque terminal mobile maintient une table de routage cohérente, même en l'absence de communication, et les protocoles dits *réactifs* où chaque terminal mobile calcule les routes vers les destinations désirées à la demande.

D'autres critères tels que l'organisation fonctionnelle et structurelle des nœuds, les sources d'informations (logique ou géographique) permettent de classifier les protocoles de routage (figure II.1); ainsi lorsque tous les nœuds ont des fonctionnalités identiques, le protocole est qualifié *d'uniforme*, si certains nœuds ont des fonctionnalités particulières dans la diffusion des messages, le protocole est *non uniforme et* ce type de protocole est utilisé *p*our diminuer le nombre de messages de contrôle nécessaires à la découverte des routes. Les protocoles de routage *non uniformes* sélectionnent certains nœuds pour créer des architectures hiérarchiques et dynamiques; parmi ces protocoles on distingue les protocoles à sélection de voisins, où chaque nœud décharge la fonction de routage à un sous ensemble de voisins directs, et les protocoles à partitionnement où le réseau est découpé en zones dans lesquelles le routage est assuré par un unique nœud maître.

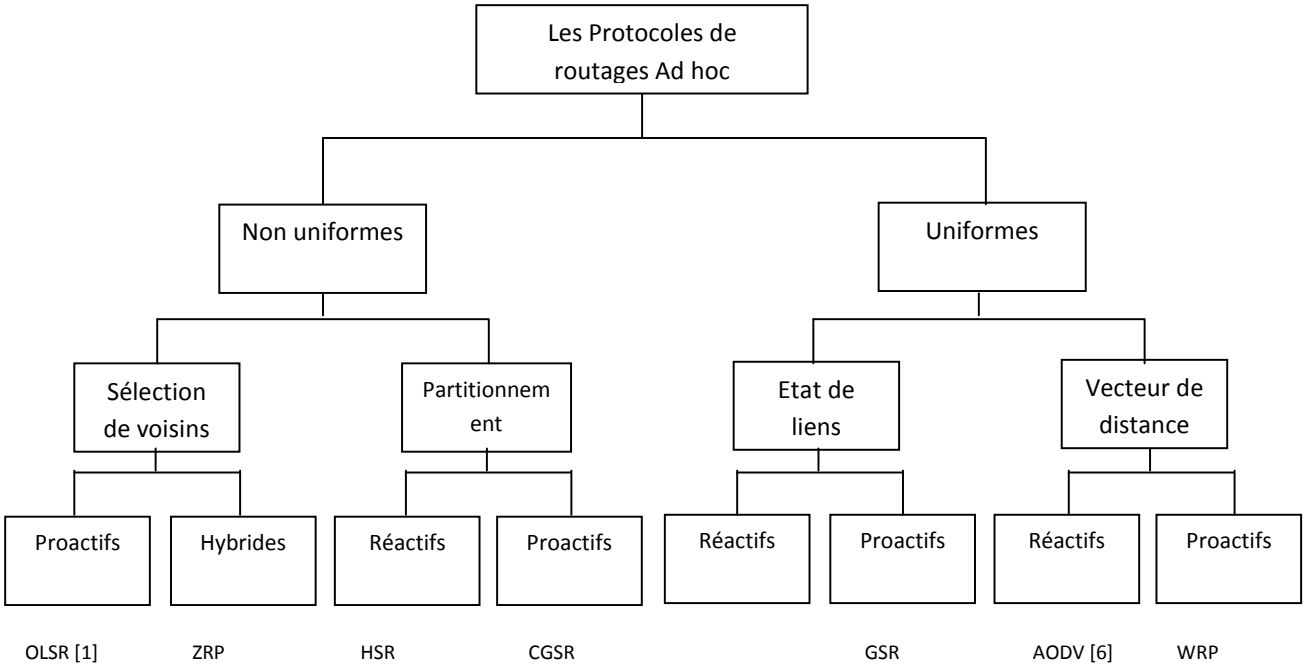


Figure. II.1. Les protocoles de routage Ad hoc

II.1. Le routage proactif

L'objectif est de maintenir une table de routage locale dans chaque nœud. Cette table contient un chemin vers tous les nœuds du réseau. Ainsi, lors de l'établissement d'une communication, le chemin est déjà connu, ce qui réduit les délais de communication [12].

La gestion de la table de routage peut être effectuée de deux manières [13] :

- ✓ Etat de lien (link state) ex : OLSR.
- ✓ Vecteur de distance (distance vector) ex : DSDV.

II.1.1. Le protocole DSDV

DSDV (Destination Sequence-Distance Vector) [7] est un protocole proactif, uniforme et orienté destination. Chaque nœud possède en permanence dans sa table de routage le chemin le plus récent découvert vers chaque nœud du réseau. Parmi plusieurs chemins possibles, DSDV privilégie le chemin le plus court. Chaque nœud maintient une table de routage dont les entrées mémorisent pour une destination: l'identifiant du prochain nœud vers

cette destination, le nombre de nœud jusqu'à cette destination (la distance), le plus grand numéro de séquence reçu pour cette destination.

Chaque nœud diffuse périodiquement sa table de routage à travers le réseau. Lors d'une nouvelle diffusion, le nœud incrémente un numéro de séquence et le transmet avec sa table de routage. Celui-ci est utilisé par les autres nœuds pour valider la mise à jour de leur table de routage et éviter les boucles. Pour limiter les informations échangées, des mises à jours incrémentales sont générées pour informer les nœuds des derniers changements intervenus (nouvelles destinations- routes perdues, etc) depuis la dernière mise à jour complète. Cette dernière est générée périodiquement par chaque nœud.

II.1.2. Le protocole OLSR

OLSR (Optimized Link State Routing Protocol [1][4]) est l'autre proposition retenue par l'IETF[43] pour le routage dans les réseaux ad hoc. OLSR est un protocole proactif qui repose sur l'échange régulier d'informations sur la topologie du réseau. L'algorithme est optimisé par la réduction de la taille et du nombre des messages échangés: seuls des nœuds particuliers, les MPR, MultiPoint Relay, diffusent des messages de contrôle sur la totalité du réseau.

Les définitions suivantes sont utilisées dans la description du protocole:

Nœud: hôte d'un réseau ad hoc implémentant le protocole OLSR.

Interface: point d'accès au réseau ad hoc. Un nœud peut avoir plusieurs interfaces, chacune ayant une adresse IP propre.

Voisin immédiat: le nœud X est un voisin immédiat du nœud Y si Y est à portée du nœud X (une des interfaces de X peut envoyer des messages sur l'une des interfaces de Y).

MPR (MultiPoint Relay): nœud sélectionné par un de ses voisins immédiats (appelé MS, MPR Selector) pour retransmettre ses messages de mise à jour. L'ensemble des MPR d'un nœud est choisi parmi les voisins immédiats, de manière à permettre d'atteindre tous les nœuds situés exactement à 2 sauts.

Lien: couple d'interfaces capables de communiquer (i.e. recevoir des messages). L'état d'un lien peut être :

- **Symétrique** SYM si les deux interfaces peuvent s'entendre.

- **Asymétrique** ASYM ou HEARD, si les nœuds ont des puissances d'émission différentes.
- **MPR**, si l'émetteur a sélectionné le nœud comme MPR, dans ce cas le lien doit être symétrique.
- **Lost**, quand le lien est perdu

Tous les nœuds envoient périodiquement des messages HELLO à leurs voisins immédiats (temporisateur HELLO INTERVAL) sur chacune de leurs interfaces. Ces messages ne sont pas relayés vers d'autres nœuds.

OLSR utilise un seul format de message. L'entête précise si le message doit être seulement transmis au voisinage immédiat ou bien à l'ensemble du réseau.

Chaque nœud garde en mémoire la description de son voisinage: interfaces voisines, voisins à 2 sauts, MPR et MS. Cette description est mise à jour à chaque réception d'un message HELLO, et les informations obsolètes sont effacées.

Le routage OLSR est basé sur l'acheminement des messages par les nœuds qui ont un voisinage symétrique. Un lien ne peut participer à une route que s'il est symétrique. Le routage vers les stations éloignées de plus d'un saut (1+N sauts) se fait grâce aux MPR, qui diffusent périodiquement des messages TC (Topology Control) contenant la liste de leurs MS. Un numéro de séquence permet d'éliminer les doublons. Ces messages servent à maintenir dans chaque station une table de la topologie du réseau. La table de routage est construite et mise à jour à partir des informations contenues dans la table des interfaces voisines et la table de la topologie, en utilisant un algorithme de plus court chemin. La métrique prise en compte est le nombre de sauts.

L'inconvénient de ce type de protocole est dû au surcoût en bande passante grâce à l'émission régulière et surtout la diffusion des paquets à travers tout le réseau.

II.2. Le routage réactif

Une solution au problème précédent consiste à n'établir une route que lorsqu'elle est demandée. Ainsi, on économise la bande passante et l'information sur la topologie est plus fraîche. Cette approche est donc plus adaptée aux environnements ad hoc. Les protocoles réactifs les plus connus sont : AODV [6] et DSR [14].

Je vais à présent expliquer brièvement le fonctionnement de cette méthode.

II.2.1. Création de la requête

Lorsqu'un nœud S veut communiquer avec un autre nœud D, S émet une requête *RouteRequest* qui va se propager de nœud en nœud jusqu' à arriver à la destination (ou à un nœud connaissant une route vers D).

II.2.2. Relais de la requête

Chaque nœud recevant une *RouteRequest*, consulte sa table de routage locale pour une route vers D. Si aucune entrée n'est trouvée, le paquet est relayé vers les voisins. Sinon, une réponse *RouteResponse* est envoyée à la source S.

II.2.3. Création de la réponse

Lorsque le paquet *RouteRequest* atteint la destination D, elle émet un paquet *RouteResponse* qui sera acheminé vers la source S, en empruntant le chemin inverse du paquet *RouteRequest*. Ainsi, le nœud source en recevant le paquet *RouteResponse* assure l'existence d'un chemin duplex vers D.

On distingue deux approches pour la gestion des routes :

II.2.4. Routage par source

Chaque nœud relayant une requête ajoute son adresse à l'entête du paquet. Ainsi, lorsque le paquet atteint la destination, il contiendra le chemin entre S et D. Dans ce cas, il n'y a plus nécessité d'une table de routage locale dans chaque nœud. C'est le cas du protocole **DSR** qui est un protocole de routage par source.

II.2.4.1. Le protocole DSR

DSR (Dynamic Source Routing) [5] est un protocole réactif, ou la source indique le chemin complet jusqu'à la destination dans l'entête du paquet. La route retenue correspond au plus court chemin, en nombre de nœuds, entre la source et la destination. En plus de cela, chaque nœud intermédiaire utilise un cache de routes, c'est-à-dire qu'il consulte en premier

lieu son cache, s'il y a une route qui mène à la destination il l'utilise, sinon il lance une opération de découverte.

Dans l'opération de découverte, le nœud source inonde un paquet de requête (RREQ Route REQuest). Lorsqu'une demande atteint la destination (ou un nœud intermédiaire qui trouve dans son cache de routes un chemin qui mène à la destination), elle contient la liste complète des nœuds traversés. La destination peut ainsi choisir la route optimale et la retransmettre à la source en empruntant le chemin inverse.

Chaque paquet possède un identificateur unique et une liste initialement vide; si un nœud s'aperçoit qu'il a déjà rencontré cet identificateur ou bien qu'il trouve que son adresse est présente dans la liste, il ignore le paquet et arrête l'inondation, sinon, il vérifie s'il n'y a pas de chemin vers la destination dans son cache. Dans le cas où il ne trouve pas de route, il met son adresse dans la liste et envoie le paquet à ses voisins, dans le cas contraire, il envoie un paquet de réponse (RREP Route REPly) au nœud source sans propager le paquet de requête.

Chaque nœud intermédiaire peut apprendre des routes par les paquets qui passent à travers lui-même et les met à jour dans son cache.

Quatre types de messages de routage sont utilisés: *demande de route, réponse à une demande, erreur* et *information.* Le cumul des options de demande et de réponse permet, par exemple, à une destination d'initier une demande de route vers le nœud demandeur tout en lui transmettant la réponse. Ce protocole exploite ainsi les liens unidirectionnels du réseau *[5]*.

Une défaillance de route se détecte par le protocole de la couche liaison ou bien implicitement si aucune émission n'est reçue à partir d'un nœud pendant un moment donné. Le nœud détecteur de la défaillance génère un message d'erreurs afin de mettre à jours les informations stockées dans les caches; il envoie pour cela un paquet d'erreur (RERR Route ERRor) à la source qui lance à son tour une nouvelle opération de découverte [17].

Pour des raisons d'économie d'énergie, les nœuds peuvent se mettre en veille. Ils signalent leur réapparition sur le réseau en diffusant un message d'information pour rétablir les liens avec leurs voisins. Certaines optimisations apportent une composante proactive à cet algorithme.

Avantages

- Peu de messages de contrôle sont échangés dans le cas ou un nombre limité de nœuds sources communiquent avec des destinations rarement ciblées.
- En cas de modification de la topologie du réseau, on peut tenter d'acheminer le paquet.

Inconvénients

- Dans le cas ou les chemins reliant deux nœuds deviennent longs, le paquet devient lui aussi grand car il doit porter les adresses de tous les nœuds intermédiaires ce qui influence négativement sur la bande passante.

II.2.5. Routage par table

Chaque nœud possède une table de routage locale contenant les routes récentes connues. Chaque entrée contient l'adresse de la destination du chemin, le prochain nœud dans cette route, ainsi qu'une métrique comparative (par exemple : le nombre de sauts). A la réception d'une *RouteRequest*, le nœud met à jour sa table, car il vient de connaitre une nouvelle route vers la source S. Lors de la réception d'une *RouteResponse*, le nœud met à jour l'entrée relative à la Destination D. Le protocole AODV est un exemple protocole de routage par table.

II.2.5.1. Le protocole AODV

Le protocole AODV (Ad hoc On demand Distance Vector) [6] n'est en réalité qu'une combinaison entre les deux protocoles DSDV et DSR.

En effet, c'est une combinaison de la demande de base de la découverte et de l'entretien de DSR et le routage de saut par saut et des numéros de séquence de DSDV.

La route retenue est bidirectionnelle et correspond au plus court chemin [6] (en nombre de sauts) entre la source et la destination. Chaque nœud maintient une table de routage dont les entrées mémorisent, pour une destination:

- L'identifiant de cette destination.
- L'identifiant du prochain nœud vers cette destination.
- Le nombre de nœuds jusqu'à cette destination.

Quand un nœud source S veut atteindre la destination D, il inonde le message de demande de route à ses voisins, ainsi que le numéro de séquence pour cette destination jusqu'à ce que le paquet atteint un nœud qui a une route à la destination ; chaque nœud intermédiaire expédie la demande et crée une route inversée vers S (mémoriser une route vers la source). Quand un nœud intermédiaire a une route vers D, il produit une réponse qui contient le nombre de saut et le numéro de séquence pour D (le plus récent). Les nœuds qui portent la réponse vers S créent une route vers D mais seulement avec le prochain saut et non pas la route toute entière.

Pour la mise à jour des routes, le protocole AODV exige des messages HELLO toutes les secondes. Un lien est considéré invalide si trois messages HELLO consécutifs ne sont pas été reçus (à travers ce même lien).

AODV suggère qu'un nœud puisse employer des méthodes de la couche physique ou de la couche lien pour détecter les ruptures.

Quand un lien devient invalide, tout nœud expédiant à travers celui-ci est informé par une réponse avec une métrique égale à l'infini, ce qui conduit au lancement d'une opération de découverte [17].

L'inconvénient de ce type de protocole est le délai avant l'envoi des messages dû au temps nécessaire pour la découverte de la route.

II.3. Protocoles hybrides

En plus des protocoles de routage proactifs et réactifs, il existe une famille de protocole de routage qui est une combinaison des deux précédents et appelés protocoles hybrides'.

Ils utilisent un protocole proactif, pour apprendre à chaque nœud son voisinage à un, deux ou trois sauts, etc. Ainsi, ces protocoles disposent des routes immédiatement dans le voisinage. Au delà de cette zone prédéfinie, le protocole hybride fait appel aux techniques des protocoles réactifs pour chercher des routes. Avec ce découpage, le réseau est partagé en

plusieurs zones, et la recherche de route en mode réactif peut être améliorée. A la réception d'une requête de recherche réactive, un nœud peut indiquer immédiatement si la destination est dans le voisinage ou non, et par conséquent savoir s'il faut aiguiller la dite requête vers les autres zones sans déranger le reste de sa zone. Ce type de protocole s'adapte bien aux grands réseaux, cependant, il cumule aussi les inconvénients des protocoles réactifs.

Parmi les protocoles appartenant à cette catégorie, nous distinguons des protocoles inspirés des colonies de fourmis qui présentent, dans la plupart des cas, de meilleurs résultats[32].

II.4. Routage basés sur le fonctionnement des colonies de fourmis

Ces derniers temps, des chercheurs se sont penchés sur l'étude du comportement des insectes afin de s'en inspirer pour l'organisation de réseaux. En effet, l'intelligence collective chez les insectes sociaux se traduit par l'émergence d'un comportement collectif intelligent macroscopique dû à des interactions simples au niveau microscopique. Le fonctionnement des colonies de fourmi en est le meilleur exemple.

Le comportement des fourmis est un comportement collectif et collaboratif. Chaque fourmi a pour priorité le bien être de la communauté.

Chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre (système complètement distribué)). Ce concept est appelé Hétérarchie (s'opposant à la Hiérarchie), où chaque individu est aidé par la communauté dans son évolution et en retour il aide au bon fonctionnement de celle-ci. La colonie est donc autocontrôlée par le biais de mécanismes relativement simples à étudier.

En faisant une projection du comportement de ces insectes sur les caractéristiques des réseaux ad hoc, on remarque que le comportement des fourmis est bien adapté à ce type de réseaux et particulièrement lors du calcul des routes.

Une nouvelle approche de routage dans les réseaux ad-hoc a vu le jour : cette approche est basée sur des algorithmes inspirés du fonctionnement des colonies de fourmis**.** Ces

algorithmes sont fondés sur la capacité des simples fourmis à résoudre des problèmes complexes par coopération.

Afin de mieux comprendre le principe des algorithmes basés sur les colonies de fourmis dédiés au routage ad-hoc, nous allons tout d'abord présenter le comportement des fourmis réelles d'une fourmi, le comportement d'une fourmi virtuelle (agent fourmi), une comparaison entre les deux aspects de la fourmi , le fonctionnement général d'un algorithme basé sur une colonie de fourmis et enfin présenter en détail le protocole de routage ARA (Ant-Colony Based Routing Algorithm) [32] utilisé pour le calcul des routes dans les réseaux ad hoc.

II.4.1. Aspect biologique : la fourmi réelle

En se dirigeant du nid à la source de nourriture et vice-versa, et ceci de façon aléatoire, les fourmis déposent au passage sur le sol une substance odorante appelée phéromones. Cette substance permet ainsi donc de créer une piste chimique, sur laquelle les fourmis s'y retrouvent. En effet, d'autres fourmis peuvent détecter les phéromones grâce à des capteurs sur leurs antennes.

Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par les autres fourmis pour retrouver les sources de nourritures trouvées par leurs congénères.

Ce comportement permet de trouver le chemin le plus court vers la nourriture (voir Fig.II.2) lorsque les pistes de phéromones sont utilisées par la colonie entière. Autrement dit, lorsque plusieurs chemins marqués sont à la disposition d'une fourmi, cette dernière peut connaitre le chemin le plus court vers sa destination. Cette constatation essentielle est la base de toutes les méthodes que l'on va développer plus loin.

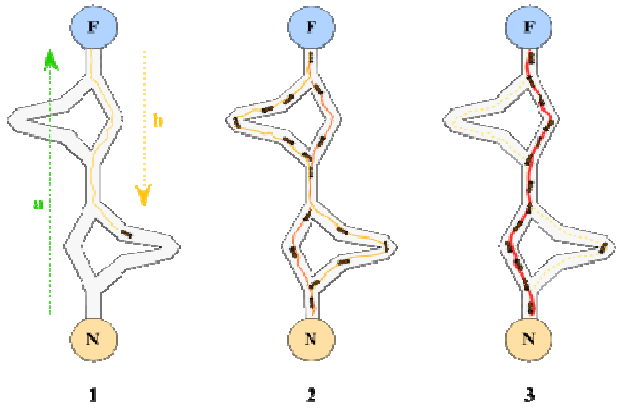


Figure.II.2.

1) la première fourmi trouve la source de nourriture (F), via un chemin quelconque (a), puis revient au nid (N) en laissant derrière elle une piste de phéromone (b).

2) les fourmis empruntent indifféremment les quatre chemins possibles, mais le renforcement de la piste rend plus attractif le chemin le plus court.

3) **les fourmis empruntent le chemin le plus court, les portions longues des autres chemins perdent leur piste de phéromones.**

II.4.2. Aspect Informatique : l'agent fourmi

Il s'agit d'un système d'agents capables de ressentir leur environnement et d'en prélever des informations qui lui permettent d'exécuter des actions simples. Le nombre d'agents est variable (il est fonction du taux de mortalité et de naissance des agents) mais cela n'entraîne aucun changement au niveau du fonctionnement global. Tous les agents fonctionnent (recherche de nourriture) en parallèle en communiquant indirectement par dépôt de phéromones.

II.4.3. Fourmis virtuelles vs fourmis réelles

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières. Nous allons maintenant synthétiser ces ressemblances et différences.

II.4.3.1. Les points communs

II.4.3.1.1. La coopération

Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui coopèrent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

II.4.3.1.2. Le support de communication

Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.

II.4.3.1.3. Le phénomène d'évaporation

La méta-heuristique ACO (voir section plus loin) comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ses anciennes décisions.

II.4.3.1.4. Le but en commun

Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

II.4.3.1.5. Les déplacements

Les vraies fourmis ne sautent pas les cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.

II.4.3.1.6. Le choix lors des transitions

Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi

du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

II.4.3.2. Les Différences

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

II.4.3.2.1. La mémoire de la fourmi

Les fourmis réelles ont une mémoire très limitée. Tandis que les fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

II.4.3.2.2. La nature des phéromones

Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifient des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

II.4.3.2.3. La qualité de la solution

Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles découvrent.

II.4.3.2.4. Le délai d'attente (dépôt de phéromone)

Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré.

II.4.3.2.5. Capacités supplémentaires

Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

- *l'anticipation* : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.
- *le retour en arrière* : une fourmi peut revenir à un état déjà exploré si la décision prise en arrivant à cet état a été mauvaise.

II.4.4. L'algorithme ACO: Simple ant colony optimization meta-heuristic algorithm

Le point commun entre les protocoles de routage ad-hoc basés sur le fonctionnement des fourmis est qu'ils utilisent tous une adaptation de l'heuristique ACO que nous détaillerons ci-dessous.

Soit G = (V, E) un graphe connexe avec n = |V | nœuds. ACO est utilisée pour trouver le plus court chemin entre une source v_s et une destination v_d d'un graphe G. Tous les arcs de G possèdent une quantité de phéromones (e (i, j) $\in$ E reliant v_i et v_j , e(i, j) $\leftarrow \psi_{i,j}$) modifiée à chaque passage d'une fourmis ; La quantité de phéromone, nous renseigne sur le nombre de fourmis qui ont empreinté cette arc.

Une fourmi se trouvant sur le nœud v_i utilise la phéromone $\psi_{i,j}$ associée au nœud j $\in N_i$ pour calculer la probabilité de choisir v_j comme nœud suivant. N_i est l'ensemble des voisins à un saut de i :

$$P_{i,j} = \frac{\psi_{i,j}}{\sum_{j \in Ni} \psi_{i,j}} \quad \text{si } j \in N_i ; \quad 0 \text{ sinon}, \quad \sum p_{i,j} = 1$$

La fourmi modifie la quantité de phéromones d'un arc e(v_i, v_j) lorsqu'elle quitte le nœud v_i pour aller vers le nœud v_j comme suit :

$$\psi_{i,j} \leftarrow \psi_{i,j} + \Delta\psi$$

Comme dans le cas des fourmis réelles, la concentration de phéromone diminue avec le temps. Dans l'heuristique ACO, elle se fait de manière exponentielle :

$$\psi_{i,j} \longleftarrow (1-q)\,\psi_{i,j} \qquad q \in [0,1]$$

Il existe de nombreux protocoles de routage fonctionnant suivant le principe des colonies de fourmis, tel que : **AntHocNet** : (An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks) [31] et **ARA** : The Ant-Colony Based Routing Algorithm for MANETs [32]. Dans ce qui suit nous donnons une présentation détaillée du protocole ARA**.**

II.4.5. Le protocole ARA: The Ant-Colony Based Routing Algorithm for MANETs

The Ant-Colony Based Routing Algorithm [32] est basé sur l'heuristique ACO et comme de nombreux autres protocoles de routage ad-hoc, on distingue trois phases :

II.4.5.1. Découverte de routes

Durant cette phase une nouvelle route est créée. La création d'une nouvelle route requiert l'utilisation de deux agents : un forward ant (FANT) et un backward ant (BANT). Un FANT est un petit paquet avec un numéro de séquence unique, qui va permettre aux nœuds de détecter les éventuelles copies d'un même FANT.

Quand un nœud source veut commencer une communication avec un nœud destination, et qu'il n'y a pas de route vers cette destination dans sa table de routage, il diffuse un FANT vers tous ses voisins à un saut. Un nœud qui va recevoir pour la première fois un FANT, crée une entrée dans sa table de routage. Une entrée dans la table de routage est un triplet (adresse destination, nœud suivant, quantité de phéromones). Le nœud interprète l'adresse de la source comme étant une destination, le nœud précédent comme étant le nœud suivant, et calcule la quantité de phéromones en prenant en considération le nombre de sauts entre la source et la destination. Ensuite, le nœud diffuse à son tour le FANT.

Quand le FANT atteint la destination, la destination crée un BANT et l'envoie vers le nœud source. Le BANT procède de la même façon que le FANT, i.e. il établit une route vers

le nœud destination. Quand le nœud source reçoit le BANT, il aura la route vers la destination, il ne lui restera qu'à envoyer le paquet de données vers la destination.

II.4.5.2. Maintenance de la route

ARA n'a pas besoin de paquets spéciaux affin de maintenir les routes. Une fois le FANT et le BANT ont établi les routes vers les nœuds source et destination, les paquets de données sont utilisés pour la maintenance de la route. Quand un nœud v_i relaye un paquet de données vers v_d à travers v_j, il augmente la quantité de phéromones de l'entrée (v_d, v_j, ψ) par $\Delta\psi$, et le nœud suivant v_j, quant à lui, augmente la quantité de phéromone de l'entrée (v_s, v_i, ψ) par $\Delta\psi$.

Le processus d'évaporation est simulé par une diminution régulière de la quantité de phéromone de chaque entrée.

II.4.5.3. Traitement des ruptures de liens

La dernière phase est le traitement des ruptures de liens détectés en l'absence d'acquittement, lors de l'envoi d'un paquet de données. Si un nœud reçoit un message de type ROUTE_ERROR via un certain lien, il désactive en premier lieu ce lien en mettant la quantité de phéromone à 0. Ensuite il recherche dans sa table de routage un autre lien lui permettant d'atteindre cette destination. Si ce lien existe, alors il envoie le paquet suivant cette route, sinon il informe ses voisins s'ils peuvent relayer ce paquet. Le processus est réitéré par chaque voisin. Le paquet peut, soit atteindre la destination ou bien, dans le cas contraire, la source renouvelle sa demande de découverte de la route.

Chapitre III

La sécurité dans les réseaux ad hoc

III. 1. Concepts de base sur la sécurité

III.1.1. Définitions

Nous présentons dans ce qui suit quelques notions utilisées dans le domaine de la sécurité des réseaux en général et des réseaux mobiles en particulier :

III.1.1.1. Cryptographie symétrique

Dans la technique de cryptographie symétrique, chaque entité possède une clé de chiffrement/déchiffrement. C'est-à-dire qu'un message chiffré (ou crypté) avec une clé donnée par un nœud expéditeur, doit être déchiffré avec la même clé par un nœud récepteur. Dans cette technique, nous pouvons trouver une clé partagée par toutes les entités d'un groupe, ou bien une clé par chaque pair d'entités, mais le problème dans cette deuxième approche est qu'il faut gérer un grand nombre de clés [18].

III.1.1.2. Cryptographie asymétrique

En utilisant la technique de cryptographie asymétrique, chaque entité possède une paire de clés : une clé publique, connue par toutes les autres entités et utilisée pour chiffrer un message donné, et une clé privée qui ne doit être connue que par l'entité qui possède la paire en question, et qui est utilisée pour déchiffrer un message. Notons qu'un message chiffré avec une clé publique ne peut être déchiffré qu'avec la clé privée correspondante.

Le chiffrage asymétrique est utilisé, par exemple, dans la distribution des clés qui seront utilisées, par la suite, pour le chiffrage symétrique [18].

III.1.1.3. Signature digitale

Appelée aussi signature numérique ou bien électronique. C'est une chaîne de données qui associe un message (dans sa forme numérique) à l'entité dont il est originaire. Les signatures digitales sont largement utilisées dans la sécurité informatique, par exemple dans

l'authentification ou l'intégrité des données, et permettent d'assurer aussi la non répudiation, que nous allons présenter après. Une des applications les plus répandues des signatures digitales est la certification des clés publiques dans les réseaux [18].

III.1.1.4. Fonction de hachage

Les fonctions de hachage prennent un message comme entrée et produisent un résultat appelé code haché ou simplement valeur hachée. Plus précisément, une fonction de hachage h transforme des chaînes de bits de longueur arbitraire finie en chaînes de longueur fixe.

L'idée fondamentale des fonctions de hachage est qu'une valeur hachée sert d'image représentative compacte (parfois appelée empreinte digitale) d'une chaîne d'entrée, et peut être employée comme si elle était uniquement identifiable avec cette chaîne.

Une classe distincte des fonctions de hachage est le code d'authentification de message (MAC Message Authentification Code), qui permet l'authentification de message par des techniques de cryptographie symétrique. Le code d'authentification de message peut être vu comme une fonction de hachage qui prend deux entrées distinctes : un message et une clé secrète, et produit une sortie d'une taille fixe, avec l'intention qu'il ne soit pas possible de produire la même chaîne de sortie sans connaître la clé secrète.

Un code d'authentification de message peut être employé pour fournir l'intégrité des données et l'authentification de l'origine des données [18].

III.1.1.5. Cryptographie à seuil (threshold cryptography)

La technique de la cryptographie à seuil est utilisée pour permettre à n parties de partager la capacité d'effectuer des opérations cryptographiques conjointement, ou bien de reconstruire un secret donné en participant avec leur propre partie du même secret [21].

Cet exploit est réalisé en divisant la clé privée en n différentes clés partielles, de telle sorte que toute association de k clés partielles parmi les n est suffisante pour effectuer une opération cryptographique donnée. Mais aussi, aucune association de k-1 clés partielles ne peut le faire [22].

III.1.1.6. TESLA

TESLA (Timed Efficient Stream Loss-tolerant Authentication) [20] est un mécanisme puissant qui offre une sécurité (authentification) pour le flux de données. Il utilise des primitifs cryptographiques symétriques et les codes d'authentification de messages, et il est basé sur la génération de clés synchronisée par l'envoyeur de messages. L'idée de base de TESLA est la suivante : L'expéditeur choisit une clé aléatoire K sans l'indiquer, et l'utilise à la génération du code d'authentification des messages (notés Pi) qu'il envoie. Dans un dernier paquet Pj, l'expéditeur révèle la clé K, ce qui permet aux récepteurs de vérifier cette clé et les MACs des paquets Pi. Si les deux vérifications sont correctes, et s'il y a une garantie que le paquet Pj n'a pas été envoyé avant que les paquets Pi ne soient reçus, alors un récepteur saura que les paquets Pi sont authentiques.

III.1.2. Conditions de sécurité

Le service de sécurité dans un réseau mobile ad hoc n'est pas différent de ceux des autres réseaux. Le but c'est de protéger l'information et les ressources des attaques et des mauvais comportements [15].

III.1.2.1. Disponibilité

Ce principe permet de s'assurer que les services réseau désirés sont toujours disponibles en dépit des attaques. Le système qui assure la disponibilité dans un réseau ad hoc cherche à combattre les dénis de service et les nœuds qui se comportent mal tels que les nœuds égoïstes.

III.1.2.2. Authentification

Ce principe permet de s'assurer que la communication d'un nœud à un autre est authentique, en d'autres termes : s'assurer qu'il n y a pas de nœud malveillant masqué (usurpation d'identité).

III.1.2.3. Confidentialité des données

Ce principe permet de s'assurer que le message ne peut être compris que par la source et la destination, ce qui implique l'utilisation d'un chiffrement symétrique ou asymétrique des données.

III.1.2.4. Intégrité

Il dénote l'authentification des données envoyées d'un nœud vers un autre, c'est à dire qu'un message envoyé de A vers B n'est pas modifié par un nœud malicieux C.

III.1.2.5. Non répudiation

Ce principe permet de s'assurer qu'un nœud ne peut pas refuser un message dont il est la source; ce principe est réalisé en appliquant une méthode basée sur la signature électronique.

III.1.3. Menaces

Dans les réseaux mobiles ad hoc, les menaces sont divisées en deux classes [15] :

- Les attaques et les mauvais comportements.

III.1.3.1. Attaques

Cette catégorie inclue n'importe quelle action qui vise volontairement à causer des dégâts dans le réseau. Elles sont classées suivant leur origine (les attaques internes et les attaques externes), ou bien suivant leur type (les attaques passives et celles actives). Avant de définir les différents types d'attaques, nous donnons les définitions suivantes :

III.1.3.1.1. Nœud malicieux

Dans les réseaux mobiles ad hoc, un nœud malicieux n'est qu'une unité mobile malveillante, ayant pour but d'écouter clandestinement le trafic, et qui peut même supporter des coûts énergétiques afin de lancer des attaques qui perturbent le fonctionnement correct du réseau [19].

III.1.3.1.2. Attaquant actif-n-m

Un attaquant noté par actif-n-m est un attaquant qui possède m nœuds (m nœuds malicieux) et qui compromis n nœuds. Par exemple, actif-0-1 est un attaquant qui possède un seul nœud, et actif-1-2 est un attaquant qui possède deux nœuds et qui compromis un seul.

III.1.3.1.3. Attaques externes

Cette catégorie inclue les attaques lancées par un nœud qui n'appartient pas au réseau ou bien qui n'est pas autorisé à y accéder [15][16].

III.1.3.1.4. Attaques internes

Cette classe inclut les attaques lancées par des nœuds internes compromis ou malveillants. C'est le type de menace le plus sévère, du moment où les mécanismes proposés pour lutter contre les attaques externes sont inefficaces devant ce type d'attaques.

III.1.3.1.5. Attaques passives

C'est une collection continue d'informations qui peuvent être utilisées par la suite lors du lancement d'une attaque active. Pour cela, l'attaquant écoute clandestinement les paquets et il les analyse pour tirer les informations voulues. Vu la nature du support de communication sans fil, il est facile qu'un attaquant lance une attaque pareille dans un tel réseau contrairement aux réseaux filaires. La solution de sécurité doit assurer dans ce cas la confidentialité. Dans ce genre d'attaque, le nœud malicieux n'utilise pas son énergie, mais opère d'une manière passive.

III.1.3.1.6. Attaques actives

Les attaques actives permettent à un adversaire de détruire des messages, d'injecter des messages erronés, de modifier des messages et d'usurper l'identité d'un nœud et par conséquent de violer la disponibilité, l'intégrité, l'authentification et la non répudiation qui sont les éléments de base de la sécurité des réseaux.

Dans un réseau ad hoc les attaques peuvent être dirigées contre les services d'une station ou ceux du réseau. Dans ce dernier cas, elles ciblent principalement ***le protocole de routage*** afin de perturber les communications entre les nœuds [9], [2], [8].

Les principales conséquences de ces attaques, présentées dans [3], sont résumées ci-dessous:

- L'introduction d'une boucle de routage.
- La création d'un *trou noir* qui consiste à rediriger le trafic vers un nœud qui ne retransmet pas les informations.
- La division du réseau en plusieurs sous-réseaux afin de bloquer les échanges entre les nœuds appartenant à des sous réseaux différents.
- Le non retransmission par un nœud de certains messages.
- L'arrêt d'un nœud en raison de son manque d'énergie.
- Le déni de service d'un nœud qui est empêché d'émettre ou de recevoir des paquets.

Exemple Routing table poisoning

Certaines optimisations des protocoles réactifs ont été développées afin d'optimiser la connaissance des chemins. Lorsqu'un nœud entend (en mode promiscuous) une RouteRequest ou une RouteResponse, il met à jour sa table de routage locale en conséquence. Un nœud malicieux peut émettre un nombre important de fausses RouteRequest et RouteResponse, remplissant ainsi les tables de routage des nœuds. Comme ces tables possèdent des tailles limitées, cela va engendrer un débordement, et les tables ne contiendront que de fausses routes.

III.1.3.2. Mauvais comportement

Nous définissons la menace par un mauvais comportement comme étant un comportement non autorisé d'un nœud interne qui peut causer involontairement des dégâts aux autres nœuds, c'est-à-dire que le but du nœud n'est pas de lancer une attaque mais il peut y avoir d'autres buts comme l'obtention d'un avantage injuste (par rapport aux autres nœuds) par exemple : un nœud n'exécute pas correctement le protocole MAC avec l'intention d'avoir une bande passante plus élevée, ou bien il peut laisser tomber les paquets des autres nœuds pour conserver ses ressources. Ce dernier comportement peut être utilisé pour lancer une

attaque du type DoS. Il est connu sous le nom : paquet dropping attack [15], ou bien sous le nom : blackhole attack.

Exemple

Pour préserver son énergie, l'attaquant peut abandonner son rôle de routeur. Par conséquent, toutes les routes passant par ce nœud seront inutilisables, engendrant ainsi une attaque de déni de service (DoS). De tels hôtes sont appelés selfish nodes ou neuds goistes

III.2. Les différents types d'attaques (Avec une projection sur quelques protocoles de routage) [46]

Dans cette section, nous allons présenter les différents types d'attaques menées contre les protocoles de routage des réseaux ad hoc, plus précisément, contre les deux protocoles DSR et AODV. Nous avons choisis ces deux protocoles car ils font parti des protocoles réactifs qui sont plus adaptés aux environnements mobiles. En plus de cela, tous les autres protocoles réactifs ont les mêmes vulnérabilités que le protocole AODV s'ils utilisent la méthode du saut par saut, ou bien celles du protocole DSR s'ils utilisent la méthode du routage de source.

III.2.1. Attaques en utilisant des modifications

Le trafic dans le réseau peut être réorienté et une attaque de déni de service peut être lancée en modifiant les informations de routage tel que l'altération des champs des messages de contrôle, des paquets de données ou bien acheminer des messages de routage avec des valeurs falsifiées [15].

III.2.1.1. Redirection en modifiant le numéro de séquence

Quelques protocoles de routage tel que l'AODV instancient et maintiennent les routes en incrémentant de façon monotone un numéro de séquence de route. Par conséquent, n'importe quel nœud peut détourner le trafic à travers lui-même en diffusant une route avec un numéro de séquence plus grand que la vraie valeur. Supposons qu'un nœud malicieux M reçoit un RREQ originaire de S afin d'atteindre la destination X, après avoir été acheminé par B. M peut réorienter le trafic à travers lui-même en répondant à B par un RREP qui contient

un numéro de séquence pour X plus grand que le dernier déclaré par X même. Eventuellement, le RREQ inondée par B atteint un nœud qui a une route valide vers X et par conséquent, un RREP valide va être envoyé vers S. Cependant, B a déjà reçu le faux RREP de M, si le numéro de séquence pour X (que M a utilisé) est plus grand que le numéro du vrai RREP, B ignore ce vrai RREP. Cette route ne va pas être corrigée jusqu'à ce qu'un vrai RREQ ou bien un vrai RREP pour X, avec un numéro de séquence plus grand, rentre au réseau.

III.2.1.2. Redirection en modifiant le nombre de sauts

Le protocole AODV utilise le nombre de sauts pour déterminer le chemin optimal, par conséquent, les nœuds malicieux peuvent augmenter leur chance d'être inclus dans le nouveau chemin crée en mettant un zéro dans le champ de calcul de sauts du RREQ qu'ils acheminent. L'attaque par redirection est possible même si le protocole utilise d'autres métriques que le nombre de sauts, dans ce cas, tout ce que l'attaquant a à faire est de changer la valeur du champ utilisé pour calculer la métrique.

III.2.1.3. Modification de la route source

Comme nous l'avons déjà vu, le protocole DSR utilise la stratégie du routage de source, ce qui fait que les nœuds seront explicitement nommés dans les paquets de données. Cependant, il y a une absence de contrôle d'intégrité, ce qui veut dire que l'altération de la route source peut facilement se faire par un nœud malicieux.

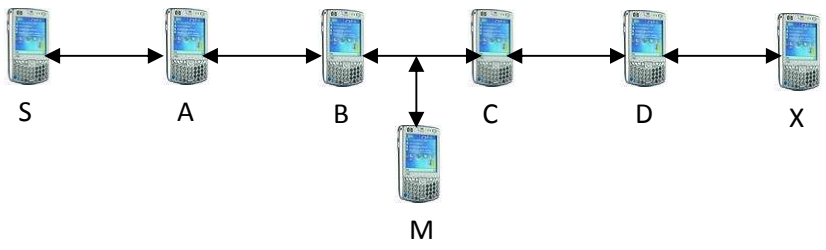


Figure III.1 Attaque en utilisant des modifications

Admettons qu'il existe une route de S vers X (voir Figure *III.1*), et admettons aussi que C et X sont hors de la portée l'un de l'autre et que M est un nœud malicieux qui essaie de

lancer une attaque de type déni de service. Supposons que S veut communiquer avec X et il a une route non expirée dans son cache. S transmet un paquet de données vers X avec la route source (S,A,B,M,C,D,X). Quand M reçoit le paquet, il peut altérer la route source, par exemple, en supprimant D de la route. Par conséquent, quand C reçoit le paquet altéré, il essaie de l'acheminer vers X, mais X est hors de la portée de C, donc le paquet ne lui parviendra jamais. C considère que le lien avec X est défaillant et envoie un RERR à S à travers M. Quand M reçoit le RERR il l'ignore. Ainsi, S continue à utiliser la route contenant M ce qui cause un déni de service. Cette attaque peut être utilisée pour causer une privation de sommeil ou bien une inanition d'énergie, puisque les paquets seront transmis et retransmis à travers les routes compromises.

III.2.1.4. Attaques par le trou de ver

Appelée aussi le tunneling, cette attaque est menée par deux nœuds malicieux distants qui collaborent pour encapsuler et échanger des messages à travers des routes existantes, et donner l'impression qu'ils sont adjacents. Cependant, ils collaborent pour falsifier la représentation de la longueur des chemins disponibles en encapsulant et en s'échangeant entre eux les messages de routage légitimes générés par les autres nœuds prévenant ainsi les nœuds intermédiaires de l'incrémentation correcte de la métrique utilisée pour mesurer la longueur des chemins.

Exemple : deux nœuds malicieux coopèrent afin de mal représenter la longueur des routes disponibles [23].

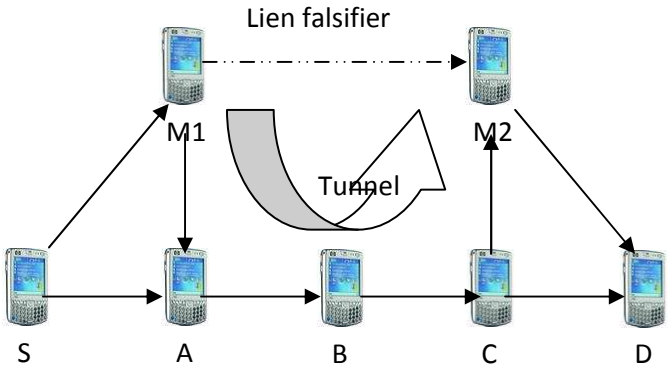


Figure III.2. Attaque par un trou de ver

Supposons que S veut créer une route vers D (voir Figure *III.2.*), pour cela, il initie une découverte de route. Quand M1 reçoit un RREQ venant de S, il l'encapsule et il le « tunnel » à M2 à travers la route (M1,A,B,C,M2). Quand M2 reçoit le RREQ encapsulé, il l'achemine vers D comme s'il avait seulement traversé S, M1, M2, D. Ainsi, ni M1 ni M2 ne mettent à jour l'entête du paquet comme quoi il a traversé aussi A, B et C. Après la découverte, il apparaît qu'il y a deux routes qui relient S à D (S,A,B,C,D et S,M1,M2,D), si M2 « tunnel » le RREP de nouveau à M1 alors S considère (faussement) que le chemin menant à D à travers M1 est le meilleur (en terme de longueur) par rapport à celui passant par A. De cette façon, les deux nœuds malicieux M1 et M2 ont créé une fausse route.

III.2.2. Attaques par personnification (spoofing attacks)

Pour mener cette attaque, un nœud représente mal son identité dans le réseau en altérant son adresse MAC ou IP dans les massages sortants, cette attaque peut être facilement combinée avec les attaques par modifications. Quand ces deux attaques sont combinées, cela donne lieu à une sérieuse désinformation telle que la création de boucles de routes.

Exemple : création d'une boucle en menant une attaque par personnification [23]

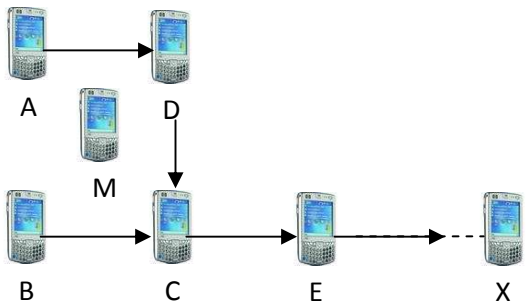


Figure III.3

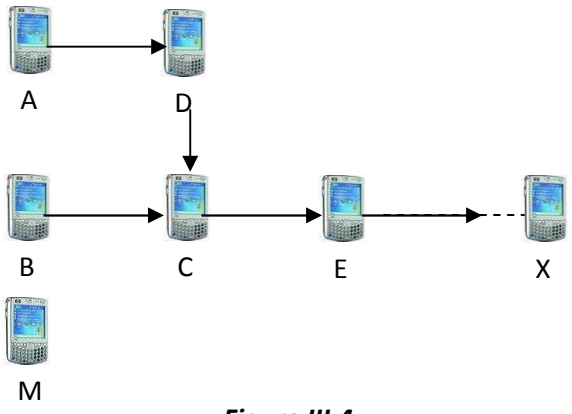


Figure III.4

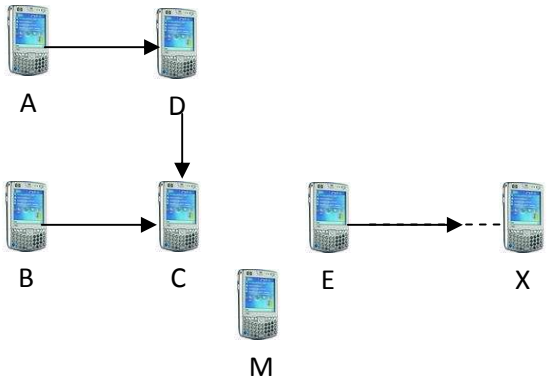


Figure III.5

Supposons qu'une route existe entre les cinq nœuds illustrés dans la Figure *III.3* vers une certaine destination X. Dans cet exemple, A est à la portée de B et D, B est à la portée de A et C, D est à la portée de A et C, et C est à la portée de B, D, et E. M est à la portée de A, B, C, et D. E est à la portée de C et le prochain nœud sur le chemin vers X.

Un nœud malicieux M, peut apprendre cette topologie en écoutant les échanges des RREQs/RREPs pendant la découverte de route. M peut former alors une boucle de telle sorte qu'aucun des quatre nœuds (A, B, C, D) ne puisse atteindre la destination. Pour commencer l'attaque, M change son adresse pour la rendre identique à celle de A, se positionne près de B et hors de la portée de A (voir Figure *III.4*). Il envoie alors un RREP à B qui contient un nombre de sauts vers X moins que celui envoyé par C. B change donc sa route vers X, pour acheminer les paquets à travers A (comme illustré dans la figure *III.4*). M change de nouveau son adresse pour la rendre identique à celle de B, se positionne près de C et hors de la portée de B (voir Figure *III.5*), et envoie ensuite à C un RREP avec un nombre de sauts vers X moins que celui qui a été annoncé par E. par conséquent, C achemine les paquets vers X à travers B

(voir figure *III.5*). Une boucle est formée et X est devenue inaccessible à partir des quatre nœuds.

III.2.3. Attaques en utilisant la fabrication

Cette classe inclue les attaques basées sur la génération de faux messages de routage. Des attaques pareilles sont difficiles à détecter [15].

III.2.3.1. Falsifier les routes erreur

Les protocoles de routage à la demande comme AODV et DSR implémentent une maintenance de chemins pour éliminer les chemins défaillants causés par la mobilité des nœuds. Quand un lien d'une route active de S vers D est défaillant, le nœud prédécesseur du lien en question envoie un paquet RERR à S, si ce dernier n'a pas d'autres routes vers D et il a toujours besoin de cette route, il lance une nouvelle opération de découverte. La vulnérabilité c'est que cette attaque peut être lancée en disséminant un faux RERR, ce qui donne une destruction de routes valides qui cause un déni de service et privation de sommeil.

III.2.3.2. Diffusion de routes falsifiées

Dans le protocole DSR, un nœud peut mettre à jour son cache par les informations qu'il trouve dans les entêtes des paquets qu'il achemine. La vulnérabilité c'est qu'un attaquant peut exploiter cette méthode d'apprentissage de route et empoisonne les caches de ses voisins en diffusant des paquets contenants de fausses routes.

III.2.4. Attaques par précipitation (Rushing attacks)

Dans la plupart des protocoles à la demande, pour limiter le coût de la découverte de route, chaque nœud achemine un seul RREQ originaire de n'importe quelle découverte de route, généralement le premier reçu. Cette propriété peut être exploitée en précipitant (Rushing) l'acheminement des RREQs reçus [15].

Pour une découverte de route, si les RREQs acheminés par l'attaquant sont les premiers à atteindre chaque voisin de la cible alors quelque soit la route obtenue par cette découverte, elle contiendra l'attaquant. De là, quand un voisin de la cible reçoit le RREQ précipité il l'achemine et n'achemine aucun autre RREQ pour cette découverte. Par

conséquent, l'initiateur va être incapable de découvrir une route qui n'inclut pas l'attaquant. En général, l'attaquant qui peut acheminer les RREQs plus rapidement que les nœuds légitimes peut lancer une attaque et s'inclure lui même dans toutes les routes qui seront découvertes.

Méthode de précipitation d'un paquet RREQ

Pour lancer une attaque par précipitation, un nœud malicieux peut utiliser une ou plusieurs de ces techniques :

- Supprimer le délai de la couche MAC et/ou réseau lors de l'acheminement des paquets. Les protocoles des couches MAC et réseau utilisent des délais d'attente dans la transmission des paquets pour éviter les collisions.

- Transmettre les RREQs en utilisant des puissances élevées : un attaquant équipé d'un support de communication physique puissant peut utiliser une puissance de transmission élevée pour acheminer les RREQs, ainsi, il couvre une surface plus grande que les autres nœuds et les nœuds éloignés seront atteints en un nombre minimal de sauts.

Contrairement aux autres techniques, celle là ne permet pas en général à l'attaquant de s'inclure dans une unique route.

- En employant l'attaque du trou de ver, deux attaquants peuvent utiliser un tunnel de haute qualité pour faire passer un RREQ afin qu'il arrive à sa destination avant les autres RREQs. Cela peut se faire quand un nœud est proche de la source et l'autre est proche de la destination, et un chemin de haute qualité existe entre ces deux nœuds.

III.2.5. Attaques du type blackhole attacks

Pour mener cette attaque, un nœud malicieux commence par s'inclure dans la route qui va être utilisée pour acheminer les données, cette phase est réalisée en menant une des attaques présentées précédemment (rushing attack par exemple). Par la suite, le nœud en question ignore (drop) tous les paquets de données qu'il reçoit, ce qui mène à un déni de service.

Notons que cette attaque diffère de l'égoïsme dans le fait que le nœud égoïste n'ignore pas les paquets de données uniquement, mais aussi les paquets de contrôle. De là, les

solutions utilisées contre les nœuds égoïstes peuvent être utilisées pour contrer ce type d'attaques.

Attaques
Attaques en utilisant des modifications
Redirection en modifiant le numéro de séquence de route
Redirection en modifiant le nombre de sauts
Modification de la route source
Trou de ver
Attaques du type *spoofing attacks*
Attaques en utilisant la fabrication
Falsifier les routes erreur
Diffusion de routes falsifiées
Attaques par précipitation (*Rushing attacks*)
Attaques du type *blackhole attacks*

La figure III.6. Les différentes vulnérabilités de quelques protocoles de routage.

III.3. Les solutions

Dans cette section, nous présentons les différentes techniques qui peuvent être utilisées pour sécuriser les protocoles de routage [15].

III.3.1. Authentification pendant toutes les phases de routage

Cette solution consiste à utiliser des techniques d'authentification pendant toutes les phases du routage, ce qui donne l'exclusion des attaquants et des nœuds non autorisés de la participation au routage. La plupart des solutions proposées, qui appartiennent à cette classe, modifient des protocoles existants pour construire des protocoles basés sur l'authentification.

Puisqu'elles emploient les signatures numériques, ces solutions se fondent sur une autorité de certification (CA), ce qui exige l'utilisation d'un serveur (digne de confiance) de certificat dont la clé publique est connue par tous les nœuds valides. Cette dépendance dans un serveur fixe rend la solution centralisée et moins flexible. L'avantage principal de cette approche est qu'elle exclut les nœuds externes non autorisés de participer au routage, donc toutes les attaques présentées précédemment sont empêchées une fois lancées par un nœud externe.

III.3.2. Métrique de niveau de confiance

Une nouvelle valeur appelée métrique de confiance a été définie, elle gère le comportement du protocole de routage. Cette métrique doit être mise dans des paquets de contrôle pour refléter la valeur minimale de confiance exigée par l'expéditeur. De ce fait, un nœud qui reçoit n'importe quel paquet ne peut ni le traiter ni l'expédier à moins qu'il fournisse le niveau de confiance exigé, présenté dans le paquet. Un exemple typique d'un protocole utilisant cette technique est SAR (Security-Aware Routing), qui va être présenté par la suite, et qui est un protocole dérivé de l'AODV et basé sur la métrique de confiance hiérarchique et l'authentification. Dans SAR, cette métrique est également employée comme critère de choix quand beaucoup de routes satisfaisant la valeur de confiance exigée sont disponibles. Définir les valeurs de la confiance des nœuds reste un problème. L'avantage de cette solution est qu'elle empêche des attaques d'un nœud interne quand un niveau élevé de confiance est exigé.

III.3.3. Randomiser l'expédition de messages

Cette technique a été proposée pour réduire au minimum la chance qu'un adversaire utilisant la précipitation peut s'inclure dans toutes les routes retournées. Dans l'expédition traditionnelle du RREQ, le récepteur expédie immédiatement le premier RREQ reçu et ignore tous les suivants. En utilisant cet arrangement, un nœud rassemble d'abord un certain nombre de RREQs, et choisit un RREQ au hasard pour l'expédier. Il y a ainsi deux paramètres liés à cette technique : le nombre de paquets du RREQ à rassembler et l'algorithme utilisé pour sélectionner les intervalles de temps (*timeouts*).

L'inconvénient de cette solution est qu'elle retarde la découverte de route, puisque chaque nœud doit attendre pendant un intervalle de temps ou recevoir un nombre donné de RREQs avant d'expédier la requête. D'ailleurs, le choix aléatoire empêche la découverte de routes optimales. L'optimalité de route peut être définie comme étant le nombre de sauts,

l'efficacité énergétique, ou selon une autre métrique. Quoi qu'il en soit, une route choisie en utilisant cette approche n'est pas aléatoire.

III.3.4. Chiffrement en oignon

Cette méthode consiste en l'utilisation d'une stratégie de chiffrement asymétrique efficace pour protéger et assurer l'anonymat des routes sources en utilisant un protocole de routage de source (DSR par exemple). Cette stratégie consiste à chiffrer une route pendant la découverte dans une forme d'oignon, et de transmettre les paquets de données à l'aide de cette route « oignon-chiffré ». Pendant la phase de réponse de route (resp. la diffusion de demande), chaque nœud ajoute son adresse à la prochaine (resp. précédente) partie de la route découverte, et chiffre le résultat en utilisant la clé publique du nœud précédent (resp. sa propre clé publique). De cette façon, chaque nœud sera capable de lire seulement le prochain saut quand des paquets de données sont transmis.

Exemple:

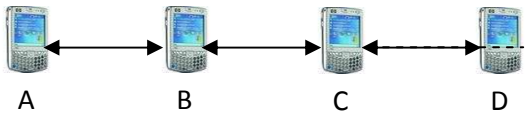


Figure III.7

Supposons qu'une route découverte (B, C, D), qui relie A à D, va être employé par A pour transmettre un paquet de données à D (voir *Figure III.7*). L'ordre « oignon-chiffré » de cette route est: [B,[C,[D]P_C]P_B]P_A (P_N dénote la clé publique du nœud N). En déchiffrant la route avec sa propre clé privée, le nœud A recherche l'adresse du B à lequel il transmet le paquet. Les autres adresses (C, D) sont cachées à A, et ne peuvent pas être déduites parce qu'elles sont asymétriquement chiffrées. De la même façon, B (resp. C) obtient l'adresse de C (resp. D) en utilisant sa propre clé privée, et à lequel il expédie le paquet.

Ce mécanisme assure que chaque nœud peut simplement identifier son successeur, et le reste de la route est maintenu anonyme. En conséquence, le déni de service obtenu en modifiant la route source est empêché. Une fois combiné avec l'authentification, ce mécanisme devient puissant et efficace, mais il souffre du coût élevé de calcul.

Le tableau suivant [15] résume pour chaque solution présentée les attaques détectées, ainsi que les inconvénients :

Solution proposée	**Type d'attaques**	**Inconvénient**
Authentification pendant toutes les phases de routage.	-Toutes les attaques externes. -La personnification. -Redirection en modifiant le numéro de séquence de route.	-Exige une autorité de certification ou un mécanisme de distribution de clés.
Métrique du niveau de confiance.	-Toutes les attaques contrées avec l'authentification. -Toutes les attaques menées contre des nœuds d'un niveau de confiance plus élevé.	-Exige une autorité de certification ou un mécanisme de distribution de clés. -Difficulté de définir un niveau de confiance.
Randomiser l'expédition de messages.	-Attaque par précipitation.	-temps d'attente.
Chiffrement en oignon.	-Toutes les attaques externes. - La personnification. -Déni de service causé en modifiant la route source.	-Exige une autorité de certification ou un mécanisme de distribution de clés. -coût élevé de calcul.

Figure *III.8.* Les attaques détectées et inconvénients de chaque solution

Les solutions citées ci-dessus sont réputés pour être de très grands consommateurs de calculs à cause de la complexité de la gestion des clés. Même s'ils sont très efficaces pour assurer la confidentialité des services et prévenir les attaques passives (eavesdropping), l'authentification des nœuds et l'intégrité des messages, il n'en demeure pas moins qu'ils restent des systèmes de prévention et ne permettent pas d'assurer la disponibilité et la robustesse du réseau. D'autres solutions complémentaires ont vu le jour notamment les modèles de renforcement de la coopération que nous allons présenter dans la suite de cette section.

III.4. Les modèles de coopération

Les protocoles de routage ad hoc existants dans la littérature et en cours de standardisation au sein de l`IETF (Internet Engineering Task Force) font l'hypothèse d'un environnement idéal dans lequel le fonctionnement du réseau n'est pas soumis à des attaques malveillantes concernant la disponibilité des services et l'intégrité des données. Ils font également l'hypothèse que tous les nœuds participent volontairement à l'opération du réseau sans tenir compte de leur tendance naturelle à s'abstenir dans le but de sauvegarder l'énergie de leur batterie. La sécurisation du routage ad hoc est particulièrement difficile en raison du manque d'entité administrative dans le coeur du réseau. Il existe de nombreuses vulnérabilités permettant à des nœuds mal intentionnés de corrompre la configuration des tables de routage, modifier les paquets en transit ou tout simplement de ne pas participer à l'effort routage dans le but d'économiser de l'énergie.

Pour ce dernier cas, La plupart des protocoles ad hoc font l'hypothèse que les nœuds coopèrent en routant les paquets des autres nœuds. Dans la réalité, cette hypothèse n'est pas toujours réaliste. Dans un réseau public, il est peu probable qu'un nœud, qui a des ressources limitées, coopère sans attendre en retour une récompense ou un gain.

Les modèles de renforcement de la coopération répondent principalement aux questions d'encouragement et de collaboration entre les nœuds d'un réseau ad-hoc. Ils sont classés selon deux catégories : reputation-based (modèles basés sur la réputation) et credit-based (modèles basés sur les crédits) ainsi décrit dans la figure.III.9.

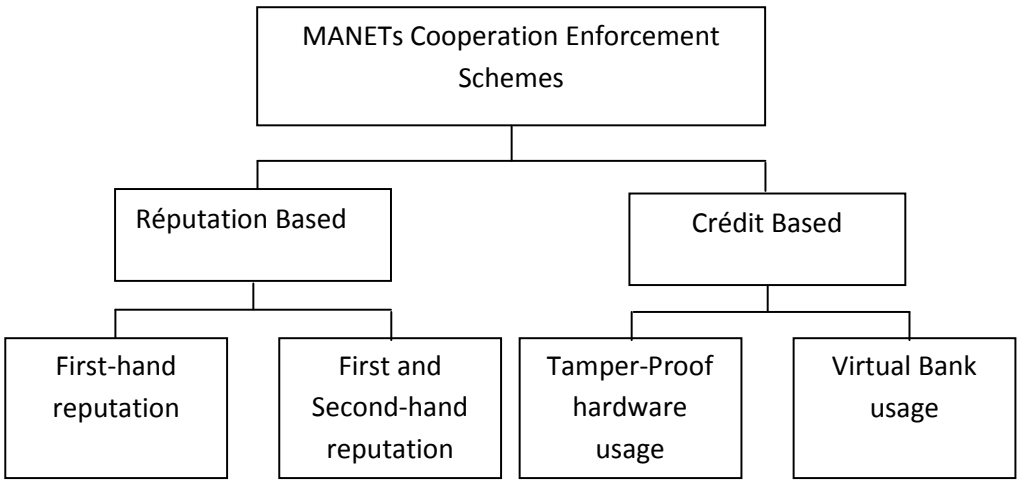


Figure. III.9. Taxonomie des modèles de renforcement de la coopération dans les MANET [33]

Les modèles basés sur la réputation utilisent la réputation des nœuds dans l'acheminement des paquets. La réputation d'un nœud augmente à chaque fois qu'il fait router correctement les paquets diffusés par ses voisins, sans leur porter atteinte. Les modèles appartenant à cette catégorie sont fournis avec un vrai mécanisme de mesure des réputations des nœuds et incorporent aussi des techniques d'isolation des nœuds ayant une faible valeur de réputation (NR : note de réputation). Ce modèle peut être divisé en deux sous classes. La première englobe les modèles dans lesquels les nœuds se basent seulement sur leurs propres observations des réputations de leurs voisins (*first-hand reputation information*) pour prendre une décision de routage.

La deuxième quant à elle, prend en considération les observations d'autres nœuds (*second-hand reputation exchange*). Les nœuds dans ce modèle échangent des informations relatives aux valeurs des réputations. Si un certain nœud voit qu'un autre nœud ne se comporte pas correctement, il en informe les autres nœuds du réseau de cette observation. Les modèles appartenant à cette sous-classe mettent en place de vrais mécanismes de distribution de ces informations.

Des Système opérationnels répondant à ce type de modèles existent déjà, nous pouvons citer OCEAN et SORI proposés dans [36], CORE [35] et CONFIDANT [34], et qui sont souvent des schémas d'extensions des protocoles de routage existants.

Dans les modèles basés sur les crédits, le routage des paquets par les nœuds est considéré comme un service pouvant être évalué et rémunéré. Ce modèle incorpore une sorte de monnaie virtuelle afin de réguler le trafic entre les nœuds dans le routage des paquets.

Nous pouvons trouver dans la littérature plusieurs systèmes fonctionnant suivant ce principe, il y a par exemple Sprit[37], TOKEN BASED[41] et Ad hoc-VCG [42]

Nous présentons dans ce qui suit quelques travaux liés au domaine de la coopération (réputaion) dans les réseaux adhoc.

III.4.1. CONFIDANT

CONFIDANT [38] détecte les nœuds malveillants grâce à un mécanisme d'observation qui fournit un rapport sur différent type d'attaque : les nœuds malveillants sont isolés et ne peuvent plus être sollicités pour le routage des paquets. CONFIDANT est proposé

comme une extension du protocole de routage DSR. Chaque nœud dispose d'un mécanisme d'observation pour construire une base de données relative à la réputation des autres nœuds du réseau : les observations locales et les observations indirectes rapportés par d'autres nœuds sont prises en compte pour évaluer le comportement des nœuds par rapport a l'exécution de la fonction de routage. CONFIDANT dispose d'un mécanisme d'alarme pour propager les informations relatives à la réputation : toutefois cette information est gérée par un module de contrôle qui détermine le degré de fiabilité de l'information en fonction de la réputation de la source de l'information. CONFIDANT, tout comme autres mécanismes basés sur la réputation, souffre d'un problème lié à la persistance de l'identité d'un nœud : il est suffisant pour un nœud malveillant de changer d'identité réseau pour se débarrasser d'une mauvaise réputation.

III.4. 2. CORE

Utilisé pour imposer la coopération entre les nœuds. CORE se base sur une technique de surveillance distribuée. Ce mécanisme de coopération n'empêche pas un nœud de nier la coopération ou de dévier d'un comportement légitime mais s'assure que les entités se conduisant mal soient punies en leur refusant graduellement les services de communication. CORE est suggéré comme mécanisme générique qui peut être intégré avec n'importe quelle fonction de réseau comme l'expédition de paquets, découverte de routes, gestion de réseau, et gestion de la localisation. Dans CORE, chaque entité réseau encourage la collaboration d'autres entités en utilisant une métrique de coopération appelée réputation. La métrique de réputation est calculée sur la base des données recueillies localement par chaque nœud et peut se baser optionnellement sur l'information fournie par d'autres nœuds du réseau impliqués dans des échanges de messages avec les nœuds surveillés. Basé sur la réputation, un mécanisme de punition est adopté comme système de dissuasion pour empêcher un comportement égoïste en refusant graduellement les services de communication aux entités qui se conduisent mal. La conséquence immédiate d'un réseau MANET qui adopte CORE est que les nœuds l´légitimes (nœuds qui coopèrent à l'opération de réseau) arrivent à économiser de l'énergie car ils ne servent pas ceux qui ont été détectés comme égoïstes.

III.4. 3. TOKEN BASED

Dans la technique de **TOKEN BASED** [41], chaque nœud doit présenter un jeton afin d'accéder au réseau ad hoc, pendant que les nœuds voisins contrôlent son comportement pour détecter une action malveillante. Une fois que la période de validité du jeton est expirée, le nœud doit présenter une requête de renouvellement auprès de ses voisins : la période de validité dépend de la durée de la participation active du nœud au réseau. Un nœud qui se comporte bien va devoir renouveler son jeton de moins en moins fréquemment. Les informations sur le comportement d'un nœud sont aussi utilisées par une version légèrement modifiée du protocole de routage AODV : les informations sur le routage qui parviennent des voisins sont comparées entre elles et chaque incohérence détectée par cette comparaison est reportée. Cette redondance permet de détecter des nœuds qui fournissent des informations erronées et de décrémenter la période de validité de leur jeton d'accès.

III.5. Les protocoles de routage sécurisés

Des efforts fournis pour la conception des protocoles de routage sécurisés sont principalement orientés aux protocoles réactifs (à la demande) tels que DSR et AODV, dans lesquels, un nœud essaye de découvrir une route à une certaine destination seulement quand il a un paquet à envoyer à cette dernière. Les protocoles de routage à la demande ont été proposés pour mieux s'exécuter avec des changements rapides de la topologie, et pour réduire au maximum les frais de routage (messages de contrôle). Pour sécuriser les protocoles de routage de base déjà proposés, l'effort principal est mis dans le but de trouver des mesures contre des attaques menées par les nœuds malicieux, qui visent à perturber intentionnellement l'exécution correcte du protocole de routage. En d'autres termes, l'objectif de toutes les solutions disponibles est un environnement contrôlé: dans un tel scénario, les nœuds voulant communiquer peuvent échanger des paramètres de sécurisation à l'avance, par exemple, des clés de session peuvent être distribuées.

Beaucoup de protocoles de routage sécurisés qui se basent sur la modification des protocoles existants ont été proposé dans la littérature tel que : Le protocole SAR, SRP [8], ARAN [23], ARIADNE [27]. Cependant ces protocoles exigent une autorité de certification ou un mécanisme de distribution de clés ce qui est contradictoire avec le principe d'un réseau ad hoc.

Conclusion

Apres avoir analysé les protocoles de routages existants dans les réseaux ad hoc ainsi que les problèmes de performance dus essentiellement au nombre de messages de contrôle utilisé, nous avons présenté une étude de la sécurité du routage dans ce type de réseaux. Ce genre de réseaux offre certes des avantages très intéressants, mais souffre des problèmes de performances et d'une sécurité très fragile.

Plusieurs attaques ont été découvertes contre la plupart des protocoles de routage existants essentiellement les mauvais comportements car la plupart des protocoles ad hoc font l'hypothèse que les nœuds coopèrent en routant les paquets des autres nœuds. Dans la réalité, cette hypothèse n'est pas toujours réaliste. Dans un réseau public, il est peu probable qu'un nœud, qui a des ressources limitées, coopère sans attendre en retour une récompense ou un gain.

Le développement de solutions de performance et de sécurité pour ces réseaux reste encore un challenge intéressant, vu les limites imposées. La résolution des problèmes de performance et de sécurité restent un point clé pour le succès des réseaux ad hoc dans divers domaines.

Le travail effectué durant la première partie de ce mémoire a permis l'étude de protocoles de routages existants, leurs caractéristiques, leurs catégories ainsi que les méthodes de gestion du réseau adoptées par ces protocoles (calcul de routes, gestion de ruptures de liens, etc...). Ce travail a permis de mettre en avant un certain nombre de problèmes supportés par ce type de protocoles (problèmes de performances, problèmes de sécurité). Je propose dans la deuxième partie de ce mémoire un nouveau protocole de routage pour réseaux adhoc permettant de résoudre en partie ces problèmes

Partie II :

Contributions : Conception et réalisation d'un protocole de routage MultiChemins basé sur la réputation des nœuds - Le protocole AntTrust -

Chapitre IV

Les fonctionnalités du protocole AntTrust

IV.1. Idées de base du protocole

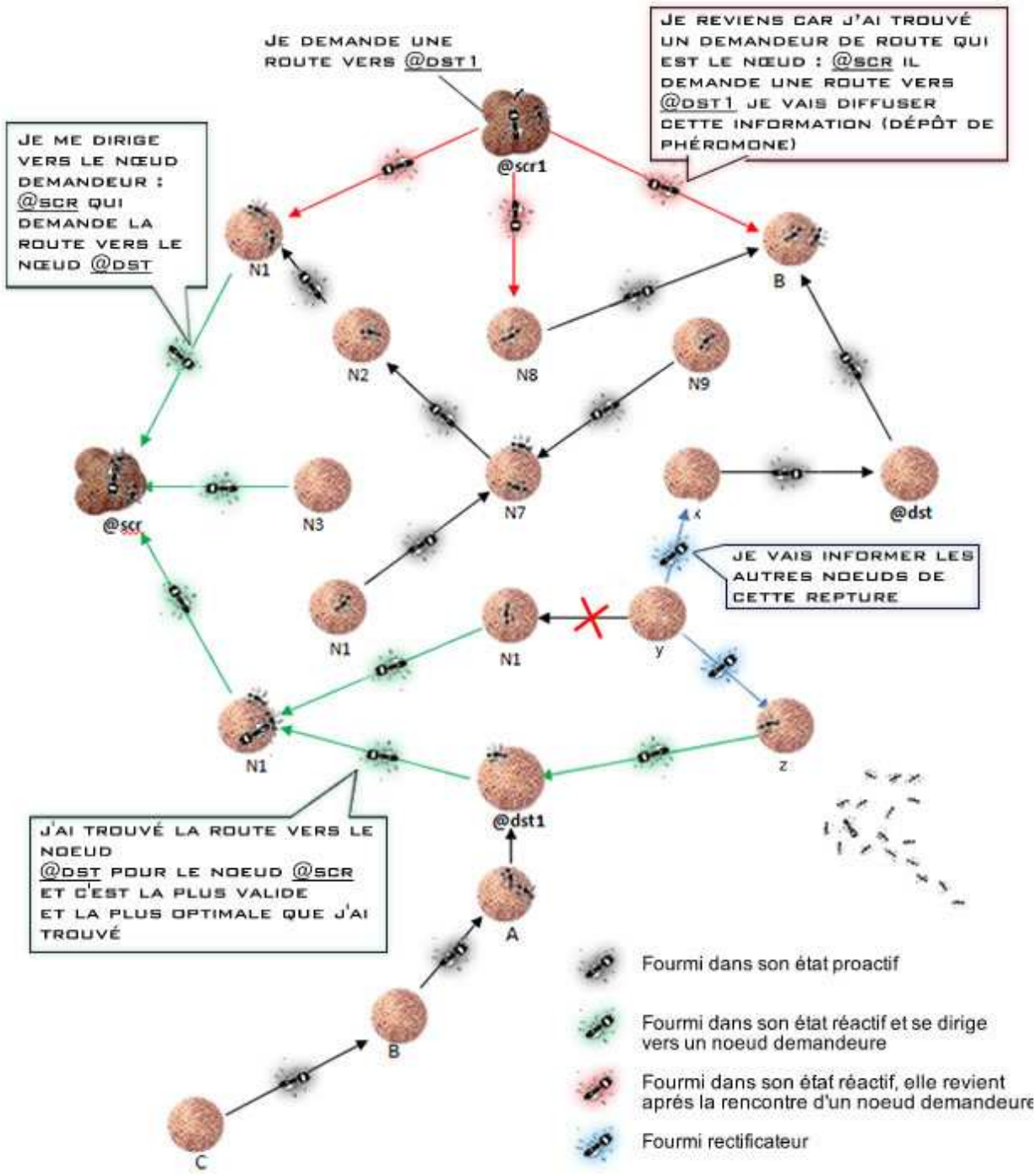


Figure.IV.1. l'idée de base du protocole

L'idée de base est de proposer un protocole de routage sécurisé pour les réseaux ad-Hoc basé sur le fonctionnement d'une colonie de fourmis dans le calcul des routes en proposant une nouvelle approche qui diffère des autres protocoles de même famille, il permet à chaque nœud du réseau d'établir une ou plusieurs routes vers d'autres nœuds, soit de manière proactive ou, réactive.

L'idée principale de notre protocole est de construire un système multi-agents, où chaque nœud fournit plusieurs types d'agents. Pour la partie routage, deux grands types d'agents sont utilisés. Le premier agent mobile est appelé Ant. Cet agent est chargé d'établir des routes de deux manières différentes comme on va voir par la suite. Le second agent mobile est appelé AntRectifier, cet agent est émis par un nœud à chaque fois qu'une modification de la topologie du réseau ou un problème de sécurité est détecté.

L'utilisation des systèmes multi-agents représente l'un des avantages de notre protocole, où un agent travaille indépendamment des autres. Cela correspond très bien au caractère spontané des réseaux sans fil tels que les réseaux ad hoc, en raison de la très grande mobilité et l'auto organisation de ce type de réseaux. Notre protocole hérite des avantages de ce type de modèle: travail autonome, intelligence distribuée robustesse. En effet, dans ce dernier cas, étant donné que chaque agent (Ant) est périodiquement créé par un nœud, le nombre d'agents dans le réseau peut être contrôlé à tout moment et est approximativement égal au nombre de nœuds du réseau. L'utilisation d'agents mobiles permet d'étendre facilement les fonctionnalités d'un protocole en ajoutant simplement d'autres agents ou par l'attribution d'autres fonctionnalités aux agents.

IV.1.1. Le caractère proactif du protocole Antrust

Le caractère proactif du protocole diffère par rapport aux méthodes classiques de même famille du fait que ces protocoles utilisent la diffusion des informations de routage tandis que le protocole AntTrust permet à chaque nœud d'envoyer un seul agent (Ant) dans une direction en fonction de certains critères (qui seront développés plus loin), ce qui minimise largement les messages de contrôle qui consomment la bande passante. Pour calculer et maintenir les chemins entre les nœuds, le protocole utilise des agents Ant qui sont créés périodiquement par chaque nœud du réseau. Un agent Ant appartient à un seul nœud appelé nœud origine. Un agent se déplace sur le réseau d'un nœud à un autre et quand il atteint un nœud, l'agent établit un chemin entre ce nœud et son origine (Figure IV.2). Pour éviter les

boucles de routage, on attribue un identifiant unique <node ID, ID Ant> à l'agent qui est incrémenté à chaque création d'un nouvel agent Ant. Si un nœud reçoit plusieurs fois le même agent, il accepte les informations fournies par les premiers et ignore les autres.

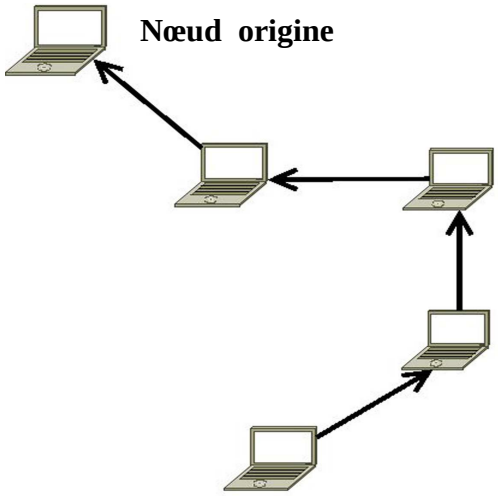


Figure IV.2. Phase Aller

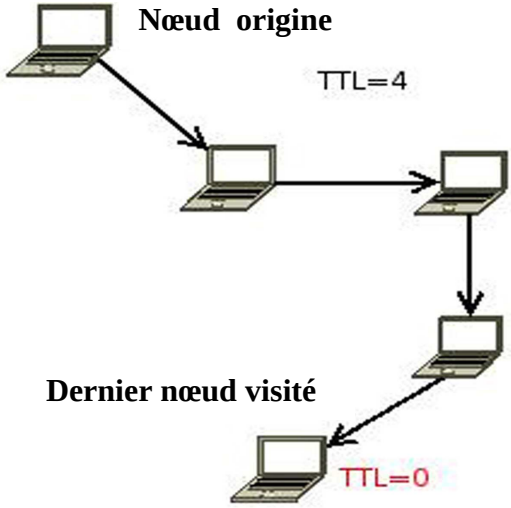


Figure IV.3. Phase Retour

Afin de suivre les activités des agents Ant, un nœud assigne une durée de vie TTL exprimé en nombre de nœud qu'il doit traverser (la valeur du TTL est proportionnelle à la dimension du réseau). Pendant leurs déplacements, les agents Ant établissent des routes vers leurs nœuds d'origine durant leur phase aller et vers le nœud où s'est terminée la phase aller (lorsque le TTL = 0) Durant leur phase retour (Figure IV.3).

IV.1.2. Le caractère réactif du protocole - une nouvelle approche -

Toutes les approches qui se basent sur le principe des fourmis utilisent une diffusion de fourmis (agents) par les nœuds sources pour la découverte des routes. Le rôle d'une fourmi est le dépôt d'une quantité de phéromones pour marquer un chemin entre une source et une destination. Contrairement à ces méthodes, le protocole AntTrust n'a pas besoin d'utiliser une technique de diffusion (broadcast) qui augmentera de façon exponentielle le nombre de messages de contrôle et par conséquent le coût important en bande passante, mais nous avons introduis une nouvelle idée qui consiste a déposer une demande locale de route chaque fois qu'un nœud veut envoyer un paquet de données vers un autre nœud. Ce sont les agents

circulant dans le réseau durant la phase proactive qui seront sollicités pour satisfaire cette demande de route : Les agents Ant ont alors un comportement hybride : réactifs et proactifs.

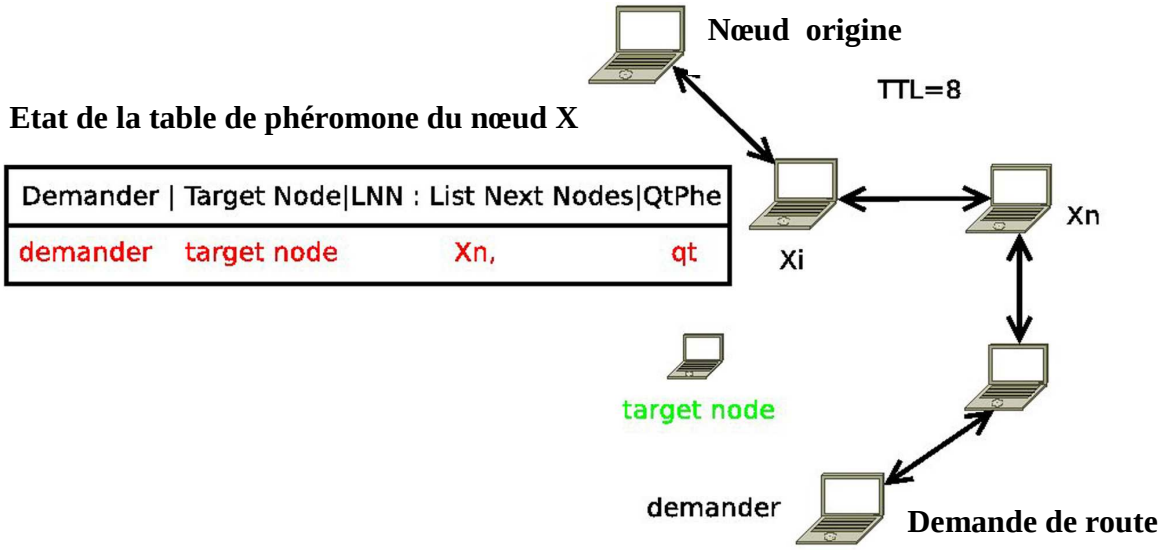


Figure IV.4. Caractère réactif de AntTrust

Quand un nœud veut envoyer des donnees a un autre nœud, mais qu'il n'a pas encore une route vers lui, il fait une demande de route pour cette destination (la demande est disposée localement). Nous noterons que la seule situation qui permet à un agent de lancer sa phase retour avant l'expiration du TTL c'est la visite d'un nœud qui fait une demande route. Au cours de cette phase (phase retour), l'agent dépose une quantité de phéromone sur chaque nœud du chemin inverse vers le nœud d'origine (le nœud demandeur), afin d'informer les autres agents sur cette demande. Ceci aura comme conséquence d'attirer plus d'agents Ant vers le demandeur (voir Figure IV.4).

En effet, lorsqu'un agent qui est dans son état proactif arrive à un nœud, il va choisir le prochain nœud à visiter, proportionnellement à la quantité de phéromones; ce processus augmente les chances de choisir un chemin vers un demandeur sans pénaliser les autres chemins. L'expression qui donne la probabilité de choisir un prochain nœud n_i parmi les nœuds n_k est:

$$p\ (n_i) = Q_i \,/\, \sum_{nj\ \epsilon\ N} Q_j$$

N est l'ensemble des voisins du nœud courant,Qi est la quantitée de pheromone associée au nœud n_i.

IV.1.3. La diffusion de la route optimale

Notre objectif est également d'assurer la diffusion de la route optimale. En fait, chaque fois qu'un agent (tout type d'agents mobiles) arrive à un nœud, il met à jour les informations de routage locale et rafraîchit ses propres informations de routage en s'appuyant sur des informations de la table de routage du nœud courant, par exemple si cet agent se dirige vers un nœud demandeur (agent avec caractère réactif), il va rechercher la route optimale vers la destination demandé pour le compte du nœud demandeur.

IV.1.4. La maintenance des routes – Les Agents rectificateurs -

Dans le but de prendre en compte le changement fréquent de la topologie du réseau, le protocole utilise un autre type d'agent appelé AntRectifier; cet agent est créé par un nœud lors de la détection d'une rupture de lien avec un voisin (en raison d'un déplacement de ce voisin) présent dans la table de routage du nœud courant comme étant un prochain saut vers une destination quelconque. Une fois créé, les agents AntRectifier sont envoyés à tous les voisins concernés par ces destinations inaccessibles (l'agent AntRectifier est envoyé à chaque nœud qui appartient à au champ **Nlist** des entrées de la table de routage pour cette destination inaccessible) dans le but d'informer les voisins de ce changement de topologie et donc prendre en Compte cette nouvelle information. Ensuite ce processus est répété par chaque voisin concerné, contrairement au protocole AODV qui diffuse un message d'erreur à tous les nœuds, même ceux qui ne sont pas concernés par cette rupture de lien. La figure suivante montre un exemple où un nœud @1 détecte une défaillance du lien avec un nœud @2.

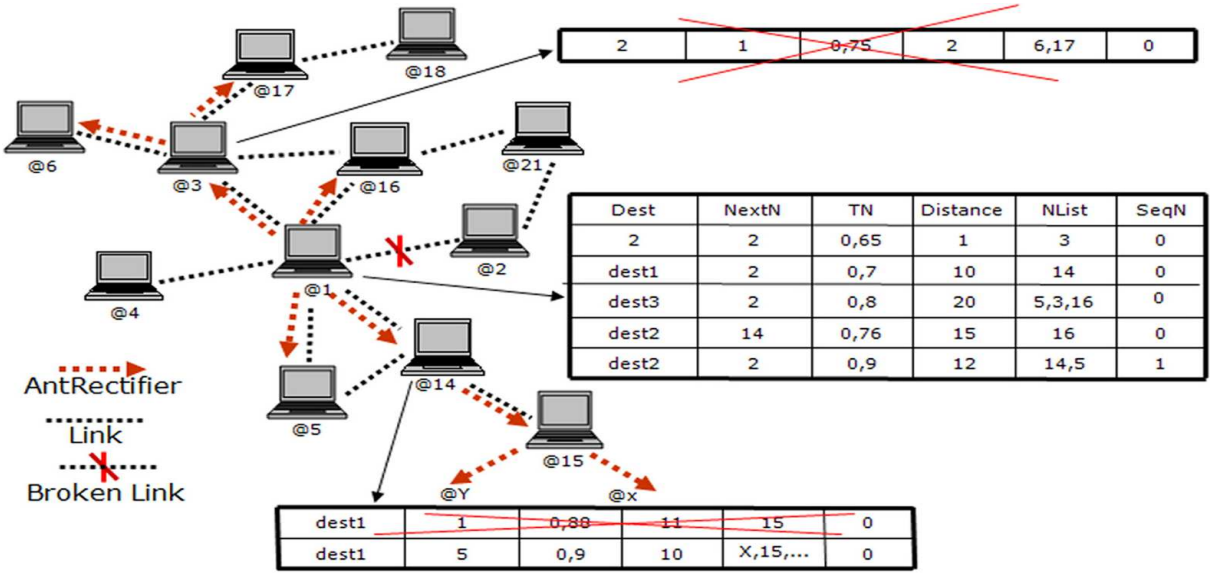


Figure IV.5. Rôle de l'agent rectificateur AntRectifier

IV.2. Description détaillée du protocole

Dans cette partie, je présente en détail les fonctionnalités du protocole; pour cela, je commence par donner quelques définitions nécessaires à la description de AntTrust, puis je passe à la description des différentes tables de routage utilisées par les différents nœuds du réseau, le principe de fonctionnement de chaque type d'agent.

IV.2.1. Définition

Une route entre deux nœuds x, y du réseau, noté Route(x, y), est une séquence de nœuds telle que x est le nœud initial et y le nœud terminal. Plus formellement, nous notons :

Route(x, y) = (x, i1, i2, ..., y) tq : x<>y, avec :

Init (Route(x, y))=x

Terminal (Route(x, y))=y

suiv (i_k) = i_{k+1}, *est le nœud suivant de* i_k *dans Route(x, y) tq :* $i_k <> i_{k+1}$.

Dans la suite de cette partie, je vais exposer le protocole, en commençant par présenter les différentes tables utilisées par chaque nœud, pour ensuite décrire le comportement de chaque type d'agents.

IV.2.2. Structures des différentes tables utilisées par un nœud

Les nœuds sauvegardent certaines informations nécessaires aux déplacements des agents, aux calculs des routes et aux routages des messages de données. A cet effet, il existe trois tables :

- Table de routage.
- Table de phéromones.
- Table des voisins.

Dans ce qui suit, nous allons détailler chacune des trois tables.

IV.2.2.1. Table de routage

Cette table est utilisée dans le routage des messages de données, noté TABROUT, sa structure est présentée par la figure VI.6. Elle contient les entrées suivantes :

- L'adresse de destination.
- L'adresse du nœud suivant pour atteindre cette destination.
- La distance en termes de nombre de sauts restant avant d'atteindre la destination.
- Liste des voisins concernés par cette route.
- La date d'expiration de l'entrée (définie selon la vitesse de déplacement des nœuds).
- L'état de l'entrée (UP, DOWN ou IN_REPARATION).

Remarque.IV.1. La table de routage sauvegarde les routes vers des destinations que les agents lui apportent. Elle est mise à jour à chaque passage d'un agent. La clé de cette table est le couple <**Destination, Nœud-suivant**>.

Destination dest	Nœud suivant n-suiv	Note de Réputation NR	Distance d	Liste-voisins Listv	Numéro de séquence seqn	Date Expiration E	Etat Stat

Figure.IV.6. : Table de routage

Le protocole est multi-chemins. De ce fait, afin de router un message de données vers une destination, un nœud devra choisir entre plusieurs routes et choisir la plus optimale. Une route optimale se définie comme suite :

Définition IV.1 Soit deux routes Routei(x, y), Routej(x, y) :

Routei(x, y) != Routej(x, y) <=> suiv(init(Routei(x, y))) != suiv(init(Routej(x, y)))

Définition IV.2 Si un nœud s veut router vers une destination d, il choisit une route de telle façon qu'elle soit optimale. De manière plus formelle :

Soit Routes(s, d) ⊂ TABROUTs L'ensemble des routes disponibles pour la destination d.

RouteOptimale(s, d) = Routei(s, d) Є Routes(s, d) tq:

TABROUTs[d, suiv(init(Routei(s, d)))].d <= TABROUTs[d, suiv(init(Routej (x, y)))].d pour toute route Routej (x, y) Є Routes(s, d)-{Routei(x, y)}

Les agents Ant peuvent calculer le temps mis pour aller d'un nœud à un autre. Ce qui conduit à avoir une autre formule de calcul de la route optimale entre deux nœuds :

RouteOptimale(s, d) = Routei(s, d) Є Routes(s, d) tq:

TABROUTs[d, suiv(init(Routei(s, d)))].t <= TABROUTs[d, suiv(init(Routej (x, y)))].t pour toute route Routej (x, y) Є Routes(s, d)-{Routei(x, y)}

Remarque 4.2. : Nous pouvons utiliser soit la distance ou bien le temps dans le choix de la route optimale.

IV.2.2.2. Table de voisinage

Un hello-message est diffusé périodiquement vers tous les nœuds à un saut (TTL=1), et chaque message contient une liste d'adresses de nœuds voisins pour lesquels il a reçu un hello-message. Le but est de constituer l'ensemble des voisins à un saut avec lesquels il a un lien bidirectionnel. A la réception d'un nouveau hello-message par un nœud, ce dernier vérifie si son adresse est contenue dans la liste de voisins contenue dans le message, si c'est le cas, le lien avec l'émetteur du message devient bidirectionnel.

La figure qui suit montre la structure de la table de voisins.

Adresse du voisin : AddrV	**Etat du lien**	**Date expiration de l'entrée ExpireE**	**Note de réputation NR**	**Pénaliser P**	**Durée de pénalisation DP**

Figure.IV.7. Table de voisinage

On note que P représente une variable booléenne qui détermine si le voisin est pénalisé ou pas, NR la note de réputation du voisin et DP la durée de pénalisation du voisin. Cette partie sera détaillée plus loin (partie sécurité).

IV.2.2.3. Table de phéromones

La table de phéromone présentée dans la figure **V.8** joue un rôle très important dans le déplacement des agents Ant**.** En effet, un agent Ant choisit le prochain voisin à visiter parmi les premiers nœuds des listes de voisins (listVoisin) de la table de phéromone, de manière stochastique et proportionnellement à la quantité de phéromones.

demandeur	destDem:destination demandée	listVoisin	quantité phéromone: qt-Ph

Fig. IV.8. Table de phéromone : TABPH

La table de phéromones d'un nœud i est mise à jour par chaque agent léger qui traverse ce nœud lors de sa phase retour. , comme indiqué par L'algorithme suivant :

Algorithme1 Mise à jour de la table de phéromone

si $TABPH_i$[*ant.NoeudEtabRout, ant.destDem*]existe
{
TABPHi[*ant.NoeudEtabRout, ant.destDem*].*qt-Ph* + ψ*qt* ;

si *ant.NoeudPreced* $\notin$ *TABPHi*[*ant.NoeudEtabRout, ant.destDem*].*listVoisin*
TABPHi[*ant.NoeudEtabRout, ant.destDem*].*listVoisin* + *ant.NoeudPreced*
}
Sinon
{
//créer un nouvelle entrée dans la table de phéromone avec :
demandeur :=ant.NoeudEtabRout ;
destDem :=ant.dstDem ;
listv+ant.NoeudPreced ;
qt-Ph :=Δ *qt* ;
}

Remarque 4.3

- ✓ La quantité de phéromone initiale ainsi que celle ajoutée (ψ*qt*) sont calculées selon la densité du réseau, de telle sorte qu'elles soient suffisantes pour la découverte de la route recherchée.
- ✓ Un processus d'évaporation des quantités de phéromone est mis en place : périodiquement, la quantité de phéromone de toutes les entrées de la table de phéromone est décrémentée. Si la quantité de phéromone d'une entrée atteint une valeur zéro, l'entrée est supprimée de la table.

IV.2.3. Description du Fonctionnement des agents

Le protocole utilise trois types d'agents mobiles : les agents *Ant* qui ont, comme principale mission, la découverte des routes, les agents rectificateurs *AntRectifier* dont le rôle est la rectification des erreurs de routage dues aux changements de topologie du réseau, et enfin les agents *Reputation* dont le rôle est d'observer l'activité des voisins puis le calcul de la réputation de ces derniers. Cette partie sera décrite en détail plus loin.

Chaque nœud du réseau a le droit d'avoir un seul agent de chaque type, pendant une période d'attente initialisée par chaque nœud localement. Durant cette période, le nœud attend le retour de l'agent *Ant* précédemment crée. Les agents *Ant* ont deux comportements différents ; L'un lors de la phase aller et l'autre durant la phase retour. Un nouvel agent *Ant* est crée par un nœud du réseau :

- Dès son insertion dans le réseau.
- Lors de l'écoulement du temps d'attente.
- Et enfin lors du retour du précédent agent et à condition que le temps d'attente qu'il lui est alloué n'a pas expiré.

Remarque 4.4. Les agents utilisent la table de phéromone et la table des voisins pour se déplacer d'un nœud à un autre.

Remarque 4.5. La valeur du TTL est choisie par rapport à la dimension du réseau, dans le but de mieux contrôler les déplacements des agents et ainsi de mieux explorer les différentes zones autour des nœuds et garder le nombre d'agents dans le réseau est peu près constant (2m par exemple), m étant le nombre de nœuds dans le réseau à l'instant t .

La Figure.4.9. montre le déplacement des agents dans le cas d'un TTL =4.

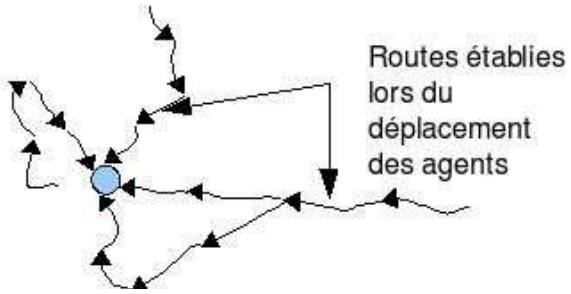


Fig. IV.9. Déplacement des agents dans le cas d'un TTL=4

Notons que les agents *Ant* et *AgentRectifier* sont des agents mobiles (qui se déplacent de nœuds en nœuds) alors que les agents *Réputation* sont locaux à leurs nouds respectifs.

Dans la suite de cette section nous allons présenter de manière exhaustive le fonctionnement de chacun des trois agents.

IV.2.3.1. Agent Ant

L'agent Ant est porteur d'un certain nombre d'informations :

- L'adresse du nœud créateur de l'agent appelé *NœudOrigine*.
- La direction de l'agent *Direction* (aller ou retour).
- Un numéro de séquence noté *seqNumber* incrémenté à chaque déplacement du nœud créateur.
- L'adresse du nœud précédemment visité noté *NœudPreced* initialisée à l'adresse du Nœud Origine.

- Un Champ TTL décrémenté à chaque saut.
- La liste des nœuds déjà visités.
- *NœudCible* : sauvegarde l'adresse d'un nœud faisant la demande de découverte de route vers une destination que l'agent sauvegarde dans une variable appelée *DestDemand*.
- *NœudEtabRout* contient l'adresse du nœud vers lequel l'agent est entrain de créer une route : initialisé à *NœudOrigine*.
- Une information booléenne sur la validité de la route du NoeudPreced vers NoeudEtabRout, initialisée à vrai.
- Le nombre de sauts noté NS pour atteindre *NœudEatbRout* qui est initialisé à zéro.
- *NœudSuiv* qui est le prochain nœud à visiter par l'agent.
- Le temps mis pour aller de *NœudEtabRout* vers le nœud courant noter t.
- L'identifiant de l'agent noté *ID* qui est incrémenté à chaque émission d'un nouvel agent.
- Des champs réservés *reserved* pour une utilisation ultérieure.

Remarque 4.6.

L'affectation d'un identifiant à un agent, combiné à son adresse permet de prévenir les éventuelles boucles pouvant se former. En effet, si un nœud reçoit deux fois de suite le même agent, il ne tient pas compte de l'information véhiculée par l'agent dans la mise à jour de la table de routage. Cette possibilité peut être remplacée par un test sur l'appartenance du nœud suivant à la liste des nœuds déjà visités.

Une fois crée, l'agent Ant est initialement dans sa phase aller et voyage d'un nœud à un autre. Dans ce qui suit, nous allons présenter, en détail les deux phases du cycle de vie d'un agent Ant .

IV.2.3.1.1. Phase aller d'un agent Ant

Afin de décrire les différentes taches effectuées par un agent au cours de son voyage, nous allons prendre l'exemple d'un agent qui atteint un nœud i et venant d'un nœud i-1.

Une fois arrivé au nœud i, il effectue plusieurs opérations :

(a)Mise à jour de la table de routage au niveau du nœud courant

Lors de cette étape, l'agent essaye d'établir une route valide et optimale à travers le nœud précédemment visité vers un nœud désigné par la variable *NœudEtabRout* qui peut être, soit le nœud origine ou bien, une destination demandée par un nœud.

Lorsqu'un agent voyage, il est porteur de *la meilleure route* disponible (en nombre de sauts) vers le *NœudEtabRout*. Si vraiment la route dont il est porteur est plus récente ou bien meilleure que celle existant déjà dans la table de routage alors il modifie l'entrée correspondante avec la nouvelle route et informe les voisins concernés de cette route en envoyant des agents rectificateurs. Par contre, si la route dont l'agent est porteur n'existe pas encore dans la table de routage, alors il crée une nouvelle entrée pour cette nouvelle route.

Remarque IV.7.

La table de routage ne contient que des routes valides, de ce fait, la mise à jour de la table de routage n'est effectuée que si la route ramenée par l'agent est valide.

L'algorithme 4 décrit cette opération de mise à jour.

Algorithm 4 Mise à jour de la table de routage

```
/* Description de la procédure suivie par un agent Ant, venant du NoeudPreced i-1 pour mettre à
jour la table de routage  du NoeudCourant i*/
si (ant.valide=vrai et ant.NoeudPreced Є TABV_noeud-courant)
{
        ant.t :=ant.t+temps estimé par l'agent pour aller du NoeudPreced vers Noeudcourant  ;
        ant.NS++ ;
        Si TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced]n'existe pas
        {
                si(TABV_i[ant.NoeudP reced] existe)
                {
                        // créer une entrée dans la table de routage avec :
                        dest :=ant.NoeudEtabRoute ;
                        n-suiv :=ant.NoeudPreced ;
                        d :=ant.NS ; // ou bien t :=ant.t ;
                        seqNumber :=ant.seqNumber ;
                        listv :=NULL ;
                }
        }
        sinon
        si[ant.seqNumber>TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced].seqNumber
        ou (ant.seqNumber=TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced].seqNumber et
        ant.NS/ant.t<TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced].(d/t))]
        {
                TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced].(d/t) :=ant.NS/ant.t ;
                TABROUT_i[ant.NoeudEtabRout, ant.NoeudP reced].seqNumber :=ant.seqNumber ;
                //Envoyer un agent rectificateur vers tous noeud x Є TABROUT_i[ant.NoeudEtabRout,
                ant.NoeudP reced].listv
        }
}
```

(b) Mise à jour de ses propres informations

Lors de cette étape, l'agent actualise ses propres informations en décrémentant par exemple son champ TTL.

L'algorithme 5 présente les principales mises à jour effectuées par un agent dans ses informations personnelles

Algorithm 5 Mise à jour des informations d'un agent léger

```
/*description des mises à jour effectuées par un agent Ant lors de la visite d'un nœud i et venant
d'un nœud i-1*/
ant.NoeudPreced :=@i //Adresse du nœud courant ;
ant.TTL- - ;
si (ant est dans sa phase aller ) ant.listNoeudVisit+@i ;
```

(c) Choisir le prochain nœud à visiter

Le choix du prochain nœud à visiter se fait toujours de manière stochastique, en donnant, bien sûr, la priorité aux voisins ayant la plus grande quantité de phéromone. Mais dans le cas où l'agent se dirige vers un nœud demandeur de route, on choisit à tour de rôle les nœuds se trouvant dans la liste de voisins de l'entrée de la table de phéromone correspondant au demandeur. L'algorithme 7 décrit de manière détaillée le choix du prochain nœud.

Le choix de la route la plus récente et la plus optimale vers le *NœudEtabRout* à diffuser, s'effectue en sélectionnant parmi toutes les routes dont la destination est le *NœudEtabRout*, celle qui a le plus grand *sequencenumber* ou dans le cas où elles ont toutes le même *sequencenumber*, choisir celle dont la distance ou le temps est le plus petit.
L'algorithme 6 décrit cette opération.

Algorithm 6 Choisir une route Optimale

/*Choisir la route optimale dans la table de routage du NoeudCourant i vers NoeudEtabRout à donner au NoeudSuiv */ soit :

ξ= {$Route_k$(*i*,*NoeudEtabRoute*) $\in$ *Routes*(*i*,*NoeudEtabRout*) tq :
TABROUTi[*NoeudEtabRout, suiv*(*init*($Route_k$(*i*,*NoeudEtabRoute*)))].*seqNumber est le maximum*}

si(RouteOptimale(i,NoeudEtabRoute)$\in \xi$ existe et $\forall$ $Route_k$(*i*,*NoeudEtabRoute*) $\in \xi$,*suiv*(*init*($Route_k$(*i*,*NoeudEtabRoute*))) <> *NoeudSuiv*)
{
ant.NS/ant.t :=*TABROUTi*[*NoeudEtabRoute, suiv*(*init*(*RouteOptimale*(*i*,*NoeudEtabRoute*)))].*d*/t ;
ant.seqNumber := *TABROUTi*[*NoeudEtabRoute,*
suiv(*init*(*RouteOptimale*(*i*,*NoeudEtabRoute*)))].*seqNumber* ;
ant.valide :=vrai ;
} sinon ant.valide :=faux ;

Algorithm 7 Choisir la prochaine destination

```
//Choisir le nœud suivant à visiter par un agent léger Ant parmi les voisins du nœud courant
Si ant.NoeudCible !=NULL
{
        Si (ant.NoeudCible = Noeud Courant) //Déclencher la phase retour
        Sinon
        {
                ant.NoeudSuiv :=TABPHE_Noeud-courant[ant.NoeudCible, ant.NoeudEtabRout].listVoisins.First ;
                 // NoeudSuiv Є TABV
                 si(NoeudCourant=NoeudEtabRout)
                 {
                         NS=0 ;
                         seqNumber=NoeudCourant.seqNumber ;
                         valide=true ;
                  }
                 Sinon
                 {
                         RouteOptimale := choisirRouteOptimale() ;
                         si(ant.valide=vrai)
                                 TABROUT_Noeud courant[ant.NoeudEtabRout, suiv(init(RouteOptimale(Noeud-
                         courant,NoeudEtabRoute)))].listv+ant.NoeudSuiv ;
                 }
        }
}
Sinon si NoeudCourant fait une demande de découverte de la route et NoeudCourant différent NoeudOrigine
//Déclencher la phase retour
Sinon si ant.TTL=0 //Déclencher phase retour
sinon // L'agent est toujours dans sa phase aller
{
        //Calculer la probabilité de choisir un voisin comme prochain nœud à visiter
        Soit V = {x Є TABV / x ∉ ant.listNoeudVisit }
        Pour tout i Є V
                P(i)= Qi / ∑j∈V Qj      Tq :
        si  i Є TABPHE Qi = qt - phe
        sinon Qi = cste
         //Choisir NoeudSuiv aléatoirement dans V;
        si ant.NoeudSuiv Є TABPHE
        {
                 soit demandeur et destDem les valeurs des champs demandeur et destDem correspondant à l'entrée choisie
                 alors :
                 ant.NoeudCible := demandeur ;
                 ant.NoeudEtabRout := destDem;
                 ant.TTL :=1;
                 si(NoeudCourant=destDem)
                 {
                         NS=0 ;
                         seqNumber=NoeudCourant.seqNumber ;
                         valide=true ;
                  }
                 Sinon
                 {
                         RouteOptimale := choisirRouteOptimale() ;
                         si(ant.valide=vrai)
                                 TABROUTNoeud courant[ant.NoeudEtabRout, suiv(init(RouteOptimale(Noeud-
                         courant,NoeudEtabRoute)))].listv+
                         ant.NoeudSuiv ;
                 }
        }
        Sinon si (NoeudCourant !=NoeudOrigine)
        {
                 RouteOptimale := choisirRouteOptimale() ;
                 si(ant.valide=vrai)
                         TABROUTNoeud courant[ant.NoeudEtabRout, suiv(init(RouteOptimale(Noeud-
                 courant,NoeudEtabRoute)))].listv+ant.NoeudSuiv ;
        }
}
```

IV.2.3.1.2. La Phase retour d'un agent Ant

Comme nous venons de le constater à travers les différentes tâches exposées ci-dessus, l'agent Ant a une autre phase durant son cycle de vie dans le réseau, qui est la phase retour vers le nœud créateur. A travers les algorithmes 8 et 9 qui suivent, nous allons décrire le comportement d'un agent Ant lors de cette deuxième phase.

Algorithm 8 Lancement de la phase retour

```
//Description du lancement de la phase retour d'un agent Ant
ant.NoeudEtabRout :=Noeud Courant ;
ant.NoeudPreced :=Noeud Courant ;
ant.NS :=0 ;
ant.valide :=vrai ;
ant.seqNumber=NoeudCourant.seqNumber ;
ant.NoeudSuiv :=ant.listNoeudVisit.suiv() ; //NoeudSuiv Є TABV
ant.destDem :=destination demandé par le nœud courant si elle existe ;
```

Algorithm 9 Phase retour d'un agent

```
//Description des taches effectuées par un agent Ant lorsqu'il atteint un nœud durant sa phase
retour
Mettre à jour la table de routage ; mettre à jour ses propres informations ;
si destDem !=NULL
        Mettre à jour la table de phéromone ;
        si(Noeud-courant=NoeudOrigine) ***fin du cycle de vie de l'agent***
sinon
{
        ant.NoeudSuiv :=ant.listNoeudVisit.suivant() ; // NoeudSuiv Є TABV
        RouteOptimale := choisirRouteOptimale() ;
        si(ant.valide=vrai)
                TABROUTNoeud courant[ant.NoeudEtabRout, suiv(init(RouteOptimale(Noeud -
        courant,NoeudEtabRoute)))].listv +ant.NoeudSuiv ;
}
```

IV.2.3.2. Agents Rectificateurs

Ce type d'agents est utilisé afin de remédier à un changement dans le voisinage d'un nœud. On fait appel aux agents rectificateurs dans tous les cas suivants :

- Rupture du lien avec un voisin

La détection des liens rompus est effectuée de différentes manières. Le premier mécanisme est l'utilisation des hello-message vus dans le cas de la construction de la table de

voisinage, si un hello-message n'est pas reçu de la part d'un voisin pendant une période d'attente , le lien avec ce voisin est déclaré rompu et l'entrée correspondante dans la table des voisins est supprimée. La deuxième méthode est la non réception des acquittements lors de l'envoi d'un message de données ; dans ce cas, il faut avoir une couche de liaison de données utilisant justement ce type d'acquittements.

Une fois la rupture décelée, toutes les destinations utilisant le lien rompu sont supprimées de la table de routage. Mais avant cela, un agent rectificateur est envoyé à chacun des nœuds de la liste des voisins. Après initialisation de cet agent, ce dernier effectue les mêmes tâches qu'un agent Ant fait dans le cadre de la mise à jour de la table de routage.

- Modification de la valeur d'une entrée de la table de routage.

Le rôle de ces agents est de rectifier la route vers toutes les destinations utilisant ce lien en informant tous les nœuds de la liste de voisins qui sont concernés par cette route.

A chaque réception d'un agent rectificateur par un nœud, ce dernier met à jour sa table de routage conformément aux informations ramenées par l'agent et crée un nouvel agent rectificateur pour informer à son tour ses voisins concernés par cette nouvelle route.

Les algorithmes 10 et 11 décrivent respectivement l'initialisation d'un agent rectificateur lors de la détection d'un lien rompu avec un voisin et les opérations qu'ils effectuent à son arrivée au nœud vers lequel il est envoyé.

Algorithm 10 Initialisation d'un agent rectificateur

Soit un nœud i qui détecte qu'un lien avec un nœud voisins j est rompu Supprimer $TABV_i[j]$
//supprimer l'entrée correspondante dans la table de voisinage ;
pour toute destination $dest \in TABROUT_i$ tq $suiv(init(Route(i, dest))) = j$ faire
Supprimer cette entrée ;
pour tous nœud $x \in TABROUT_i[dest, j].listv$ et $x \in TABV_i$faire
Lui envoyer un agent rectificateur contenant les informations suivantes :
{
ant.NoeudSuiv :=x ;
ant.NoeudEtabRout :=dest ;
ant.NoeudPreced :=Noeud Courant //@i ;
RouteOptimale := choisirRouteOptimale() ;
si(ant.valide=vrai)
TABROUTNoeud courant[*ant.NoeudEtabRout, suiv*(*init*(*RouteOptimale*(*Noeud - courant,NoeudEtabRoute*)))].*listv +ant.NoeudSuiv* ;
}

Algorithm 11 Taches effectuées par un agent rectificateur lorsqu'il arrive dans un nœud

/* Description de la procédure suivie par un agent rectificateur venant du nœud i-1 pour mettre à jour la table de routage du noeudCourant i*/
si $TABROUT_i$[*ant.NoeudEtabRout, ant.NoeudP reced*] existe
{
si (ant.valide=vrai)
{
ant.t :=ant.t+temps estimé par l'agent pour aller du NoeudPreced vers Noeud-courant;
ant.NS++ ;
$TABROUT_i$[*ant.NoeudEtabRout, ant.NoeudP reced*].(*d/t*) :=ant.NS/ant.t ;
$TABROUT_i$[*ant.NoeudEtabRout, ant.NoeudP reced*].*seqNumber* :=ant.seqNumber ;
Envoyer un agent rectificateur vers tous *nœud x Є TABROUT*$_i$[*ant.NoeudEtabRout, ant.NoeudPreced*].*listv* ;
supprimer x dans $TABROUT_i$[*ant.NoeudEtabRout, ant.NoeudPreced*].*listv* ;
}
sinon
{
supprimer $TABROUT_i$[*ant.NoeudEtabRout, ant.NoeudPreced*] ;
Envoyer un agent rectificateur vers tous *noeud x Є TABROUT*$_i$[*ant.NoeudEtabRout, ant.NoeudP reced*].*listv* ;
}
}

IV.3. Le modèle de réputation (coopération) dans AntTrust

IV.3.1. Calcul de la réputation

La plupart des protocoles ad hoc font l'hypothèse que les nœuds coopèrent en routant les paquets des autres nœuds. Dans la réalité, cette hypothèse n'est pas toujours réaliste. Dans un réseau public, il est peu probable qu'un nœud, qui a des ressources limitées, coopère sans attendre en retour une récompense ou un gain.

Dans les réseaux ad hoc, il existe un risque de déni de routage de la part de certains hôtes mobiles. Un hôte peut tout simplement rejeter les paquets en transit qu'il reçoit afin d'économiser son énergie. En d'autres termes, un hôte peut profiter du service de routage fourni par les autres nœuds lorsqu'il est en communication avec un hôte destinataire, cependant rien ne l'oblige à fournir le même service en assistant au routage des paquets émis par les autres hôtes. Plusieurs nœuds sont susceptibles de se comporter ainsi, ce qui peut rendre le concept de routage ad hoc très limité.

La méthode que nous proposons incite à la coopération entre les nœuds d'un réseau ad hoc grâce à une technique d'observation collectif et un mécanisme de réputation. Chaque nœud du réseau observe le comportement des ses voisins par rapport à une fonction spécifique, par exemple le routage ou l'acheminement des paquets (packet forwarding), et collecte des informations sur l'exécution de cette fonction. Si le résultat attendu coïncide avec le résultat observé, l'observation va avoir une valeur expriment une bonne réputation, autrement elle va avoir une valeur expriment une mauvaise réputation. En se basant sur les observations collectées au fur et à mesure du temps qui passe, chaque nœud calcule une valeur de réputation attribuée à chacun de ses voisins en utilisant un mécanisme d'évaluation sophistiqué.

Dans le cas du protocole AntTrust, chaque nœud du réseau va disposer d'un agent local (ReputationAgent) dont le rôle est de surveiller les activités de ses nœuds voisins en termes d'entrée et sortie de paquets (en d'autres termes, il va surveiller les paquets reçus par un nœud voisin et les paquets transmis par ce même nœud). La note de réputation est calculée comme suite :

$$NR = (NB_{A->x} / NBF_{A->x})^{alpha} * (NBother_{s->x} / NBFother_{s->x})^{beta}$$

$NB_{A->x}$: Le nombre de paquets envoyés par le nœud A au nœud x.

$NBF_{A->x}$: Le nombre de paquets que x a réellement forwadé.

$NBother_{s->x}$: Le nombre de paquets reçus par x des autres nœuds autre que A.

$NBFother_{s->x}$: Le nombre de paquets que x a réellement forwardés (réexpédiés).

Alpha et beta sont deux réels à fixer par la suite afin de donner plus ou moins d'importance aux paquets forwardés de A ou des autres voisins du nœud.

IV.3.2. Agent Réputation

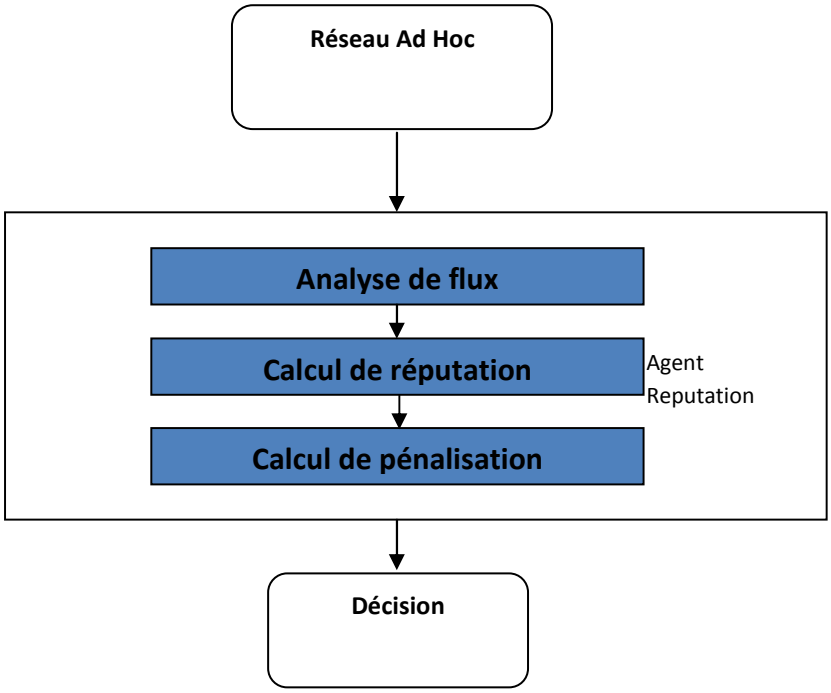


Figure.IV.1. Rôle de l'agent Réputation

Le rôle de l'agent Ant est donc de surveiller les activités de ses voisins (Analyse de flux voir figure IV.1). A partir de cet ensemble de flux, l'agent Réputation à chaque période de temps calcul la réputation du nœud voisin surveillé, pour définir après le degré de pénalisation de ce nœud. On définit le nombre Px = (1 – NR) le degré de pénalisation du nœud x. la décision de pénaliser ou pas un nœud égoïste est une fonction stochastique (probabilité). Si la décision prise est oui alors Durant un temps t proportionnel à P et en

fonction de l'historique de ce nœud, le nœud A va empêcher les agents passant par lui d'envoyer ces agents vers le nœud x. L'algorithme suivant montre le fonctionnement d'un agent Réputation en détail :

Algorithm 12 Algorithme général de fonctionnement de l'agent

```
/* Description de la procédure suivie par l'agent Reputation du nœud a */
Pour chaque période du temps T
Pour chaque nœud Voisin V
{
        Calcul de NB_{a->v} ;
        Calcul de NBF_{a->v} ;
        Calcul de NBother_{s->v}; //tel que Rep(other) >0.5
        Calcul de NBFother_{s->v};
        Rep = (NB_{a->v}/ NBF_{a->v}) ^alpha *(NBother_{s->v}/ NBFother_{s->v}) ^beta;
        P = (1 – Rep) ;
        Décision = alea(p);
        Si (décision==1)
        {
                Si (TABV_a.DP>=D) // si la durée de pénalisation est supérieur a une durée D
                {
                        //supprimer le nœud V définitivement ;
                        Marquer l'information dans  TABV_a ;
                        TABV_a .P=vrai ;
                        Pour  toute destination dst
                        {
                                Si (TABROUTE_a (v, dst)) existe Supprimer TABROUTE_a (v, dst);
                                //envoyé un agent rectificateur ;
                        }

                } sinon
                {
                        //Marquer la pénalisation du nœud V dans TABV_a et déterminer la durée
                        //de pénalisation selon la valeur précédente;
                        TABV_a .P=vrai ;
                        Pour  toute destination dst
                        {
                                Si (TABROUTE_a(v, dst)) existe Supprimer TABROUTE_a (v, dst);
                                //envoyé des agents rectificateurs ;
                        }
                }
        } Sinon Si ((TABV_a .P==vrai) et (TABV_a.DP<=now))
        //si le nœud est pénalisé et que la durée de pénalisation est expirée
        {
                //dépénaliser le  nœud V de TABV_a ;
                TABV_a .P=faux ;
                //envoyé des agents et établir des routes à travers ce nœud ;
        }

}
```

Si la décision prise par un nœud A est oui, alors durant un temps t proportionnel à Px, le nœud A va empêcher les nœuds Ant passant par lui d'être envoyés vers le nœud x. Cette

opération va donc provoquer le fait que la table de routage de x sera moins fréquemment mise à jour ; le nœud x peut même se retrouver dans l'incapacité d'accéder au réseau si ses autres voisins appliquent la même politique de pénalisation envers lui (en effet sa table de routage ne sera jamais mise à jour, et par conséquent il ne pourra pas envoyer ses paquets à lui vers d'autres nœuds du réseau). De plus ce processus est complètement décentralisé et s'adapte bien au protocole AntTrust.

Pour prendre en charge les erreurs lors de la surveillance des voisins, on attribue un bas niveau de réputation seulement aux nœuds qui montrent les traits caractéristiques d'égoïsme d'une façon persistante. Pour cela, après une certaine période de pénalisation, le nœud est remis parmi les voisins du nœud courant, mais s'il va montrer encore une fois un comportement égoïste il va être marqué comme nœud pénalisé dans la table de voisinage avec une durée de pénalisation plus que la précédente. Si la situation persiste et que le nœud pénalisé atteint une certaine durée, il sera écarté définitivement.

La conséquence immédiate est que les nœuds légitimes (nœuds qui coopèrent à l'opération de réseau) arrivent à économiser de l'énergie car ils ne servent pas ceux qui ont été détectés comme égoïstes.

Remarque. IV.1

Avec ce nouveau mécanisme de sécurité, le protocole AntTrust va au delà du simple calcul des routes les plus courtes, en calculant les routes les plus sûres. Puisque le protocole intègre un mécanisme permettant à chaque nœud d'avoir à tout moment une note de réputation de tous ses voisins, ce qui permet à chaque nœud de décider de router ou non des messages vers ou à travers un voisin donné.

Pour l'agent Ant par exemple, en plus de la diffusion de la route optimale en terme de nombre de sauts, il transporte aussi la route la plus sure et la plus valide trouvée. Une route valide se définit comme suit :

Une route entre deux nœuds est dite valide, si chaque nœud faisant partie de la route fait confiance à son successeur :

$$Route(i, j) = valide \Longleftrightarrow qlqs\ x \in Route(i, j), NR(x, suiv(x)) >= 0.5$$

Où NR(x, y) est la note de réputation que le nœud x possède au sujet de y.

Remarque. IV.2

La route optimale dans ce cas se définie comme suit :

Soit Routes(s, d) ⊂ $TABROUT_s$ L'ensemble des routes disponibles pour la destination d.

RouteOptimale(s, d) = $Route_i$(s, d) ∈ Routes(s, d) tq:

TABROUTs[d, suiv(init($Route_i$(s, d)))].nr >= TABROUTs[d, suiv(init($Route_j$ (x, y)))].nr pour toute route $Route_j$ (x, y) ∈ Routes(s, d)-{$Route_i$(x, y)}

ET

TABROUTs[d, suiv(init($Route_i$(s, d)))].d/t <= TABROUTs[d, suiv(init($Route_j$ (x, y)))].d/t pour toute route $Route_j$ (x, y) ∈ $Route_s$(s, d)-{$Route_i$(x, y)}

Remarque. IV.3

On note que l'envoie des agents rectificateurs peut se faire aussi dans le cas ou la note de réputation qu'un nœud possède au sujet d'un voisin devient inférieure au seuil (0,5).

Chapitre V Mise en œuvre du protocole AntTrust à l'aide du simulateur NS2

V.1. Techniques d'implémentations choisies

Les simulations ont été programmées à l'aide de l'environnement NS2[45]. Pour cette partie, le protocole ANTTRUST a été développé entièrement avec le langage C++ et intégré dans NS2 en respectant le même processus de développement que les autres protocoles.

En faisant un état de l'art sur les différents protocoles de routage dans la littérature, nous avons implémenté quelques techniques permettant le bon fonctionnement du protocole et qui prend en compte les problèmes rencontrés avec les protocoles de routage étudiés.

V.1.1. Implémentation d'une file d'attente

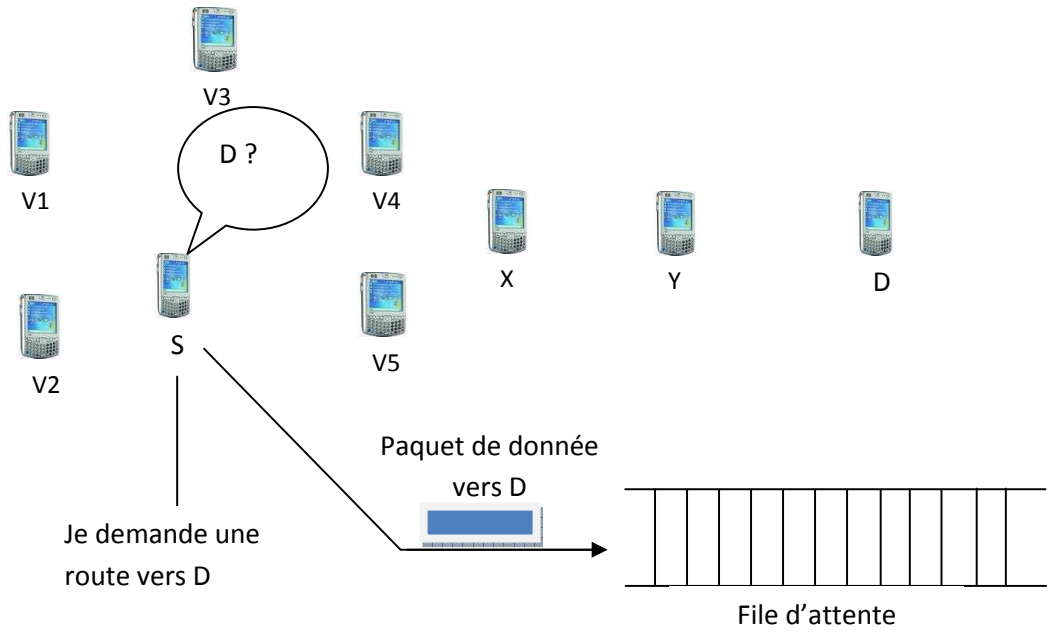


Fig.V.1. Implémentation d'une file d'attente

Pour la mise en cache des données dont la route n'est pas encore établie (en cas ou implémenté une file d'attente qui bufférise tous les paquets afin de les envoyer après la détection de la route par un agent Ant au caractère réactif. Une entrée dans la file d'attente est caractérisée par une durée de vie et la file d'attente par une taille maximale.

A chaque détection d'une nouvelle route, tous les paquets qui se trouvent dans la file d'attente et qui sont concernés par ce chemin seront envoyé.

V.1.2. Temps d'expiration d'une entrée de la table de routage

Vu que la topologie des réseaux ad hoc varie, une entrée dans la table de routage possède une date d'expiration (*rt_expire*) qui est mise a jour a chaque relais d'une donnée, cette valeur est définie selon le *type d'application* et *la vitesse de déplacement* des nœuds.

Apres l'expiration d'une entrée dans la table de routage, s'il ya des paquets de données a envoyé et qui sont concerné par cette entrée le nœud doit déposer une demande de route et empiler ces paquets de données dans la file d'attente.

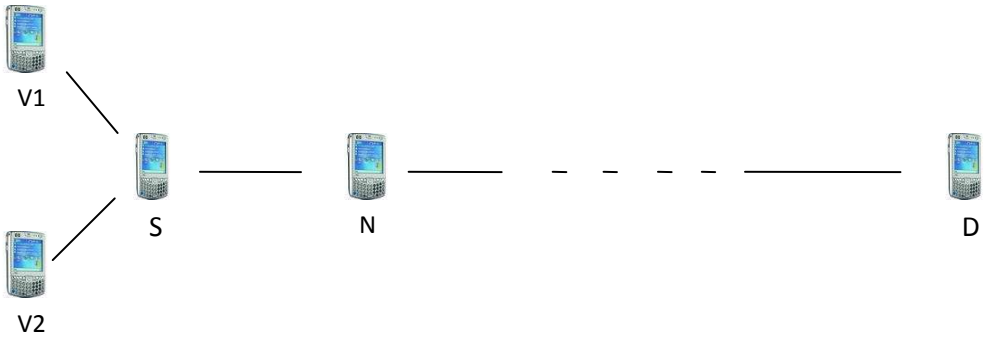


Fig.V.2. Temps d'expiration d'une entrée de la table de routage

Table de routage du nœud S

Destination : **dest**	Nœud **suivant : n-suiv**	Note de **réputation : NR**	Distance : **d**	Liste-voisins : **Listv**	Numéro de séquence : **seqn**	Date expiration : ***rt_expire***
D	N	0.8	8	V1, v2	1024	date

V.1.3. L'envoi des agents Ant Proactif aux voisins

Afin de récupérer le maximum de nouveaux chemins et dans le cas ou les agents envoyés du nœud courant son proactif (c'est-à-dire qu'ils ne se dirigent pas vers un

demandeur de route), le nœud envoie ces agents à ses voisins à tour de rôle comme illustrer dans la figure suivantes :

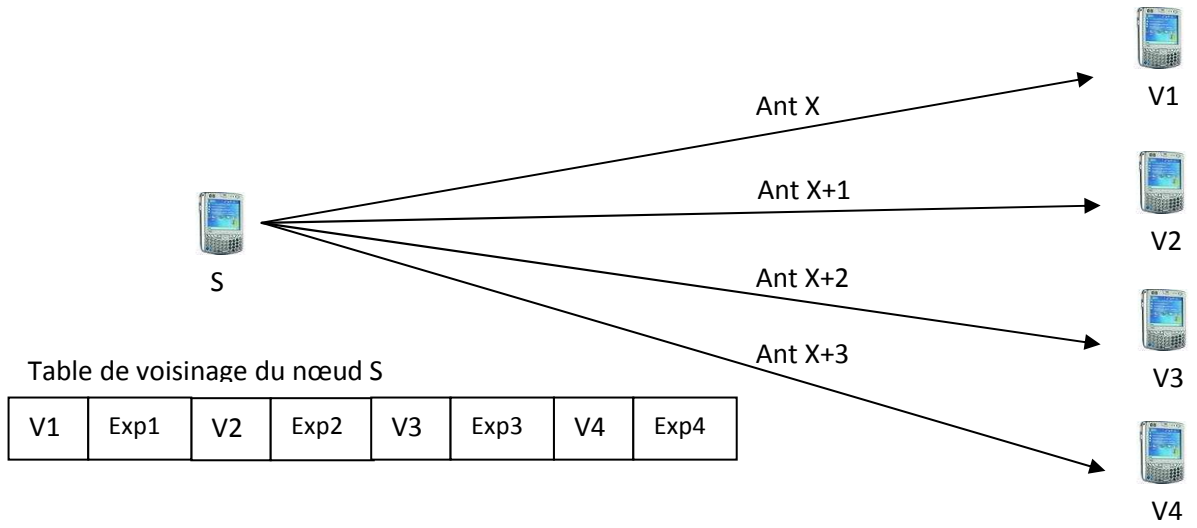


Figure.V.3. L'envoi des agents Ant Proactif aux voisins

V.1.4. Gestion des ruptures au moment de l'envoie

Les agents *AntRectifier* assurent la gestion des ruptures des liens avec les voisins et informe les autres nœuds de cette rupture. Cependant dans certains cas la rupture avec un voisin se détecte juste au moment ou un nœud veut envoyer une donnée même si l'entrée correspondante dans la table de routage existe et est valide, donc la donnée sera perdue, pour cela une gestion de tel type de rupture est nécessaire.

AntTrust utilise un mécanisme qui consiste à introduire la notion de réparation des entrées de la table de routage, c'est-à-dire que, après la détection de cette rupture, l'entrée de la table de routage serait marquée *En_Reparation*, le paquet de donnée sera empilé et une demande de route est déposée localement.

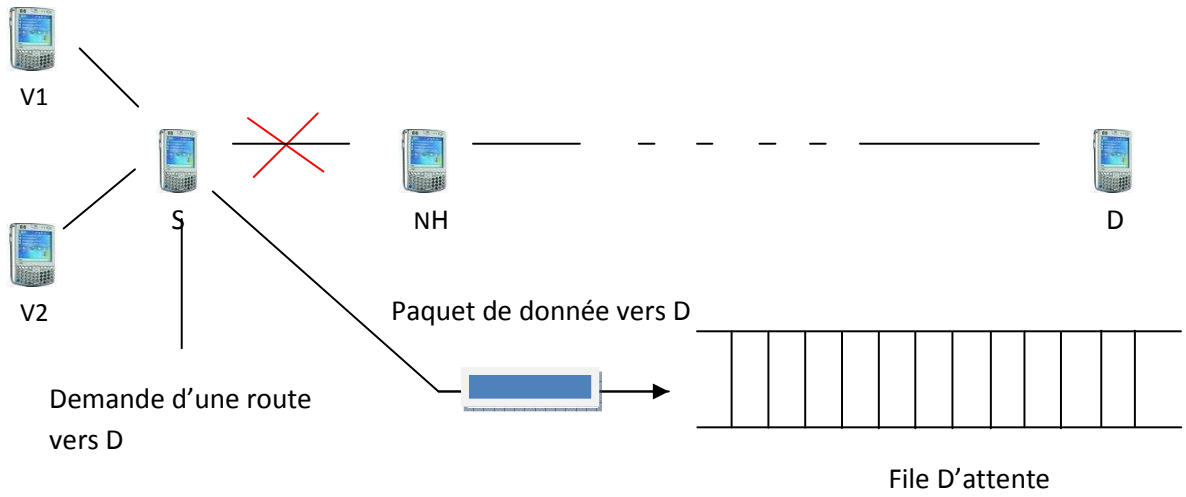


Figure.V.4. Gestion des ruptures au moment de l'envoi

Table de routage du nœud S

Destination : **dest**	Nœud **suivant : n-suiv**	Note de **Réputation : NR**	Distance : **d**	Liste-voisins : **Listv**	Numéro de séquence : **seqn**	Date expiration : ***rt_expire***	Flag
D	N	0.8	8	V1, v2	1024	date	RTF_IN_REPAIR

V.1.5. L'envoi direct des paquets de données aux voisins

Cette technique repose sur le fait que si un nœud veut envoyer un paquet (son paquet ou le paquet d'un autres nœud), et que la destination est un voisin du nœud courant, il envoie directement le paquet à partir de la table de voisinage sans procédé à parcourir la table de routage ni a déposé une demande de route. C'est une technique d'optimisation qui apporte beaucoup d'avantages en termes de performance et qui n'est pas implémentée dans plusieurs protocoles.

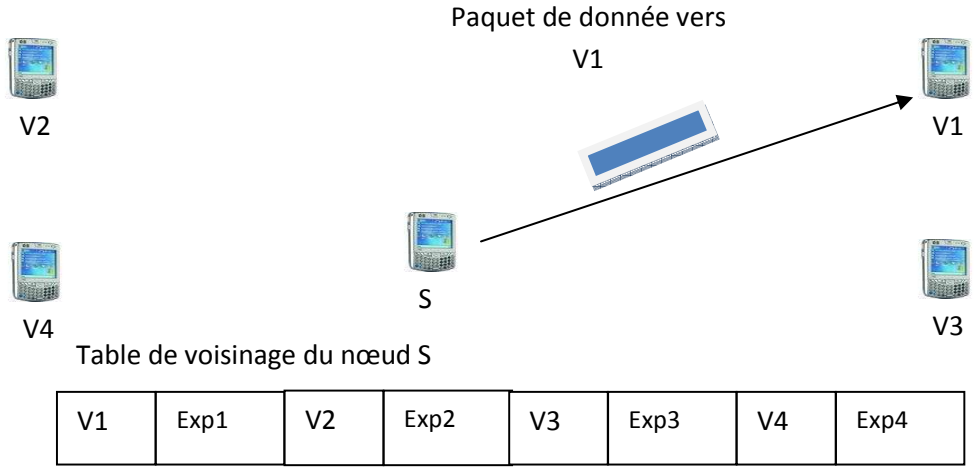


Fig.V.5. L'envoi direct des paquets de données aux voisins

V.2. Les résultats de la simulation sur NS2

Dans chaque simulation les nœuds se déplacent à une vitesse maximale de 5m/s. Le trafic a été généré automatiquement au moyen d'un script Tcl, que nous avons développé et en générant aussi des scénarios avec le programme setdest de NS2. Le trafic généré est aléatoire, dans lequel n communications sont établies en choisissant n couples de nœuds au hasard. Une communication se résume par l'envoie d'un paquet de 512 octets au moyen du protocole UDP.

L'étude suivante montre le comportement des deux protocoles AODV et ANTTRUST, ainsi qu'une comparaison de leurs performances. Pour se faire, nous avons analysé leur scalabilité suivant deux paramètres :

- *Le nombre de nœuds du réseau* : pour cette classe de simulation, le nombre de nœuds varie entre 20 et 100, le nombre de paquets émis est fixé à 700.
- *Le trafic entre les nœuds* : le nombre de nœuds est fixé à 80 et le nombre de paquets émis varie entre 100 et 1000.

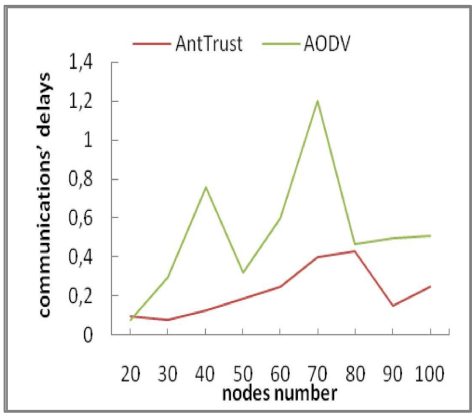
Les métriques étudiées sont les suivants :

- *Les délais de communications* : qui représentent le délai moyen entre le temps d'émission d'un paquet et le temps de sa réception.

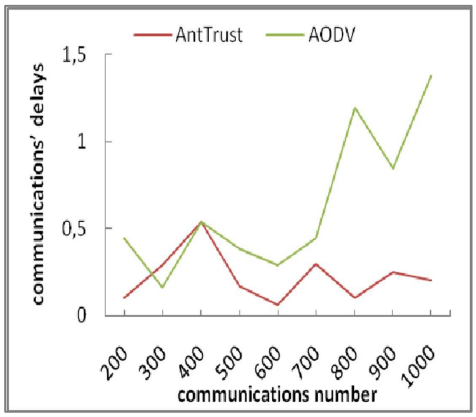
- *Le nombre de messages de contrôle* : qui représente le nombre de messages de contrôle (ou d'agent dans le cas de ANTTRUST) nécessaires aux différentes taches réalisées par le protocole.
- *Le nombre de messages participant à la découverte des routes* : qui représente le nombre de messages (ou d'agents dans le cas de ANTTRUST) utilisés par le protocole pour la découverte des différentes routes.

V.2.1. Délais de communication

Cette métrique permet de mesurer la rapidité du protocole et son temps de réponse. Des délais trop longs rendent l'utilisation du protocole inadaptée pour la plupart des applications.



Figures 1(a):



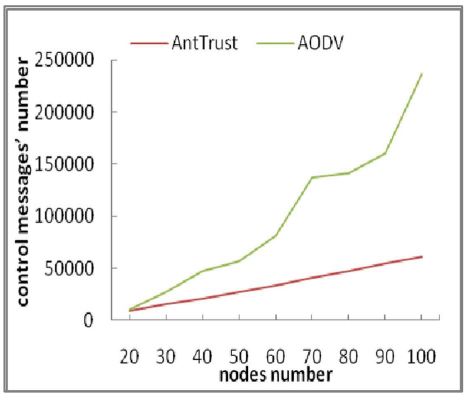
Figures 1(b):

Les figures 1(a) et 1(b) montrent les délais obtenus par les simulations. Le protocole ANTTRUST présente un délai de communication plus petit que celui d'AODV, c.à.d. que la procédure de routage du protocole ANTTRUST est plus rapide que celle du protocole AODV. Ce qui démontre l'apport de l'utilisation :

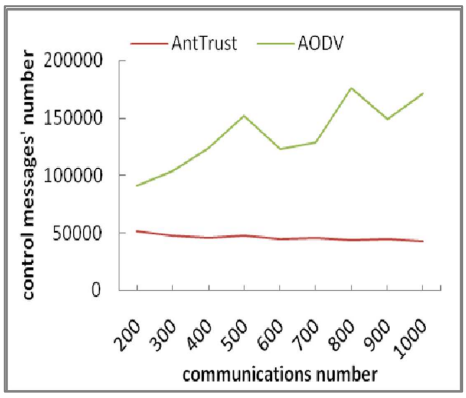
- Du caractère proactif du protocole dans le cas où la route est déjà établie donc il n'y a pas de demande de route.
- De l'envoi direct des données quand il s'agit d'un voisin.
- Du choix du chemin le plus court car le protocole est multipath.

V.2.2. Le nombre de messages de contrôle

Cette métrique permet de mesurer la consommation du protocole en termes de nombre de messages de contrôle utilisés. Ce qui permet d'avoir une idée sur sa consommation en bande passante ainsi que de l'énergie des nœuds.



Figures 2(a):

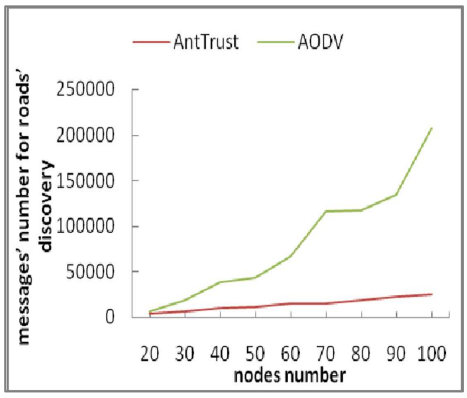


Figures 2(b):

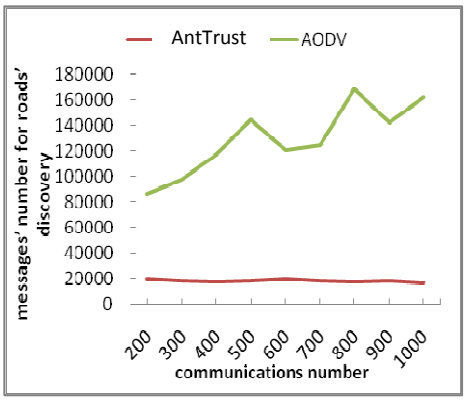
D'après les figures 2(a) et 2(b), on remarque que ANTTRUST utilise un nombre largement plus réduit de messages de contrôles que AODV. Et cela revient au fait que AODV utilise généralement la diffusion des messages pour la réalisation de ses différentes taches (demande de route, détection des ruptures des liens,…) tandis que ANTTRUST utilise des méthodes plus optimales telles que le choix du prochain saut, les agents rectificateurs,…

V.2.3. Le nombre de messages de découverte des routes

Cette métrique permet de mesurer la consommation du protocole en termes de nombre de messages utilisés pour la découverte des routes. Nous avons choisi cette métrique car généralement c'est le type de messages qui consomme le plus de bande passante pour un protocole de routage.



Figures 3(a)



Figures 3(b)

D’après les figures 3(a) et 3(b), on remarque que ANTTRUST utilise un nombre largement plus réduit de messages pour la découverte des routes que AODV, particulièrement pour le graphe qui représente la variation de nombre de paquets, car AODV utilise la diffusion pour la découverte des différentes routes pour chaque paquet de données, tandis que ANTTRUST c’est le caractère proactif des agents qui se transforme en un caractère réactif ce qui permet de diminuer le nombre de messages (ou agent ANTTRUST) participant à la procédure de recherche (exploration), donc la consommation de la bande passante est plus réduite, évitant l’attente des trames au niveau de la couche 2 (couche MAC).

Chapitre VI

Conclusion générale

Les réseaux ad hoc représentent une solution partiellement exploitée dans le domaine des communications, du fait d'un certain nombre de contraintes rendant difficile la tache de conception de protocoles de routage fiables (liberté totale de mouvement, absence d'un référentiel fixe, vulnérabilité . . . etc).

Dans ce travail, on a étudié entre autres la méthode basé sur le fonctionnement des colonies de fourmis dans le calcul des routes optimales et la notion de modèle de réputation pour le coté sécurité pour un protocole de routage, ce travail nous a donnée la possibilité de voir les protocoles de routage d'un autre coté.

L'étude, qui a porté sur l'analyse des protocoles existants, nous a permis de faire ressortir un ensemble de problèmes. Afin de les résoudre, nous avons conçu un nouveau protocole de routage, le protocole AntTrust.

AntTrust se base sur une idée originale qui diffère des autres protocoles de routage de même famille. L'idée principale consiste à déposer une demande de route locale (pour un nœud qui veut envoyer des donnes à une destination) au lieu de diffuser une demande de route dans tout le réseau (qui représente un vrai problème pour ce type de réseau : coûts élevés en terme de bande passante) comme le cas des autres protocoles de la même classe. Dans AntTrust, c'est le rôle des agents Ant circulant dans le réseau de diffuser et ramener au nœud demandeur la route la plus sûre et la plus optimale trouvée vers la destination demandée.

AntTrust vise à construire un système multi agent qui s'adapte à ce type de réseau. L'utilisation des systèmes multi-agents représente l'un des avantages de notre protocole, où un agent travaille indépendamment des autres. Cela correspond très bien au caractère spontané des réseaux sans fil tels que les réseaux ad hoc, en raison de la très grande mobilité et l'auto organisation de ce type de réseaux. Notre protocole hérite donc des avantages de ce type de modèle: travail autonome, intelligence distribuée et de la robustesse.

L'utilisation de ces systèmes permet aussi d'étendre facilement les fonctionnalités d'un protocole en ajoutant simplement d'autres agents ou par l'attribution d'autres fonctionnalités liées à d'autres aspects (par exemple des fonctionnalités liées à d'autres aspects de la sécurité).

Les résultats obtenus par simulation, comparés aux autres protocoles tels que AODV et selon un certain nombre de paramètres, encouragent et montrent la validité de notre conception. AntTrust a montré de meilleurs résultats que le protocole AODV par exemple.

Le modèle de réputation proposé pour AntTrust est original par rapport aux autres travaux existants. Dans les modèles existants, la réaction d'un nœud A à un nœud voisin B qui ne coopère pas (qui ne route pas les paquets de A) est de refuser (avec une certaine probabilité) de router les paquets de B, dans notre cas la pénalisation d'un nœud non coopératif consiste à l'isoler petit à petit (pendant une durée t, pour prendre en compte les erreurs d'analyse de flux) du réseau. L'avantage principal de cette approche est que ce travail se fait de façon totalement distribuée.

Ce travail, ayant apporté des améliorations dans les domaines des réseaux ad hoc, envisage un certain nombre de perspectives tels que la finalisation et l'implémentation de la partie sécurité et en particulier le modèle de réputation avec le simulateur ns 2, pour pouvoir faire des simulations et voir des résultats qui montrent l'apport du protocole AntTrust en terme de sécurité.

En conclusion, notre travail a permis de résoudre un certain nombre de problèmes rencontrés dans les réseaux ad hoc en élaborant ce protocole, qui présentent des performances plus optimales par rapport aux protocoles étudies, permettant une plus large utilisation de ce type de réseau.

Bibliographie

[1] T. Clausen and P. Jacquet. Optimized link state routing protocol. http ietf.org/internet-drafts/draft- ietf-manet-olsr- 11 .txt, July 2003.

[2] Y.C. Hu, D.B. Johnson, and A.Perrig. Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Systems and applications - WMCSA,* June 2002.

[3] Y. Huang and W. Lee. A cooperative intrusion detection *system* for ad hoc networks. In *Proceedings of 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, Fairfax, VA, USA,* October 2003.

[4] Ph. Jacquet, P. Muhlethaler, and A. Qayyum. Optimized link state routing protocol, draft-ietf-manet-olsr-00.txt. Internet Draft, IETF MANET Working Group, November 1998.

[5] D. B Johnson and D. A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, Mobile Computing, volume 353. Kluwer Academic Publishers, 1996.

[6] Charles. E. Perkins, Elizabeth M.Royer. “Ad hoc on demand distance vector (AODV) algorithm”. In Systems and Applications (WMCSA’99).1999, pages 90-100.

[7] Ch. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In Proceedings of the Conference on Communications Architectures, Protocols and Applications-A CM/SIGCOMM’94, pages 234 244, 1994.

[8] P.Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In Proceedings of the Communication Networks and Distributed Systems Modeleing and Simulation - CNDS, January 2002.

[9] L. Zhou and Z. Haas. Securing ad hoc networks. IEEE Network Magazine, 13(6), November-December 1999

[10] K. Mase, Y. Wada, N. Mori, K. Nakano, M. Sengoku, and S. Shinoda. Flooding schemes for a universal ad-hoc network. In Industrial Electronics Society, volume 2, pages 1129 1134, 2000.

[11] W. Peng and X.-C. Lu. On the reduction of broadcast redundancy in mobile ad-hoc networks. In Mobile and Ad-Hoc Networking and Computing, pages 129 130, 2000.

[12] S. Basagni. Mobile Ad Hoc Networking. IEEE Press, 2004.

[13] A. Tanenbaum. Computer Networks. Prentice Hall, 4th edition, 2003.

[14] D. Johnson. Routing in ad hoc networks of mobile hosts. 1994.

[15] D. Djenouri, L. Khelladi, and A. N. Badache. A survey of security issues in mobile ad hoc and sensor networks. Communications Surveys & Tutorials, IEEE, 7(4) :2– 28, 2005.

[16] H. Yang, H. LUO, F. YE, S. LU, and L. ZHANG. Security in mobile ad hoc networks : Challenges and solutions. IEEE Wireless Communications, pages 38–47, 2004.

[17] Dr. Nadjib BADACHE, Djamel DJENOURI, Abdelouahid DERHAB, Tayeb LEMLOUMA. "Les protocoles de routage dans les réseaux mobiles Ad hoc", Revue RIST. Vol 12 N° 2, 2002, pages 77-112.

[18] A. Menezes, P. Van Oorschot, S. Vanstone. "Handbook of Applied Cryptography". CRC Press, 1996.

[19] Pietro Michiardi, Refik Molva. "Ad hoc networks security". ST Journal of System Research, Volume 4, N°1, Mars 2003.

[20] Adrian Perrig, Ran Canetti, J. D. Tygar, Dawn Song."Efficient Authentication and Signing of Multicast Streams over Lossy Channels". IEEE Symposium on Security and Privacy, S&P, Oakland, California, USA.2000, pages 56-73. http://www.citeseer.ist.psu.edu/perrig00efficient.html

[21] Emmanuel Bresson, Jacques Stern, Michael Szydlo."Threshold Ring Signatures and Applications to Ad-hoc Groups".Advances in cryptology – Proceedings of CRYPTO'02, Santa Barbara, California.Août 2002, pages 465-480.

[22] Douglas Stinson."Cryptographie - Théorie et Pratique". Traduction de la version originale : "Cryptography – Theory and Practice". Edition Vuibert, Paris, 2001.

[23] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, E. Belding-Royer. "A secure routing protocol for ad hoc networks". In the 10th IEEE International Conference on Network Protocols (ICNP 02). Novembre 2002.

[24] Ioanna Stamouli, Patroklos G. Argyroudis, Hitesh Tewari."Real-time Intrusion Detection for Ad hoc Networks". Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05), 2005.

[25] Levente Buttyan, Istvan Vajda. "Towards Provable Security for Ad Hoc Routing Protocols".The second ACM workshop on Security of Ad hoc and Sensor Networks (SASN'04), Washingtion DC, Octobre 2004.

[26] Le routage à vecteur de distance. http://cisco.goffinet.org/s2/vecteur_distance

[27] Yih-Chun Hu, Adrian Perrig, David B. Johnson. "Ariadne: A secure On-Demand Routing Protocol for Ad Hoc Networks". In proceedings of MOBICOM 2002.

[28] : Yih-Chun Hu, Adrian Perrig, David B. Johnson. "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks". Proceedings of the Twenty-Second

Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), San Francisco, CA. Avril 2003, vol. 3, pages 1976-1986.

[29]Laura Marie Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T99-07, 1, 1999.

[30] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Communications, 1999.

[31] Gianni Di Caro, Frederick Ducatelle, Luca Maria. An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks. Technical Report No. IDSIA-27-04-2004, Septembre 2004.

[32] Mesut Gunes, Udo Sorges, Imed Bouazizi. ARA-The Ant-Colony Based Routing Algorithm for MANET. International Workshop on Ad Hoc Networking (IWAHN 2002), Van-couver, British Columbia, Canada, August 18-21, 2002.

[33] G.F. Marias, P. Georgiadis, D. Flitzanis et Mandalas. Cooperation enforcement schemes for MANETs : A survey. 2006 John wiley & Son.

[34] Buchegger S, Le Boudec JY. Performance analysis of the CONFIDANT protocol. In Procee- dings of 3rd ACM International Symposium, on Mobile Ad Hoc Networking and Comput- ing, Juin 2002.

[35] Michiardi P, Molva R. CORE : a collaborative reputation mechanism to enforce node coope- ration in mobile ad hoc networks. In Proceedings of 6th IFIP Communication and Multimedia Security Conference, Septembre 2002.

[36] Bansal S, Baker M. Observation-based cooperation enforcement in ad-hoc networks. Techical Report, Stanford University, 2003.

[37] Zhong S, Chen J, Yang R. Sprite : a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In Proceedings of IEEE INFOCOM2003, April 2003.

[38] Buchergger, S., Boudec, J.Y.L. : Performance Analysis of the CONFIDANT Protocol : Cooperation Of Nodes, Fairness in distributed ad hoc networks, Proceedings of IEEE MobiHoc2002

[39] Michiardi, P., Molva, R. : Core: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks, in proceeding of IFIP CMS 2002 Conference.

[40] Michiardi, P., Molva, R. : A Game Theoretical Approach to Evaluate Cooperation Enforcement Mechanisms in Mobile Ad hoc Networks, in proceedings of ACM/IEEE WiOpt 2003 Workshop.

[41] Yang, H., Meng, X., Lu, S. : Self-Organized Network-Layer Security in Mobile Ad Hoc Networks, ACM Workshop WiSe 2002.

[42] Anderegg L, Eidenbenz S. Ad-hoc-VCG: a truthful and costef .cient routing protocol for mobile ad-hoc networks with sel.sh agents. In Proceedings of 9th Annual International Conference on Mobile Computing and Networking, September 2003.

[43] http://www.ietf.org/

[44] J. Moy : OSPF Version 2 RFC2328 April 1998.

[45] P. Anelli & E. Horlait : NS-2: Principes de conception et d'utilisation, Version 1.3

[46] Sécuriser les protocoles de routage des MANETs, PFE: Outhmane Mahmoudi et Nadjib Bouamama, USTHB.2005-2006.