République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme MASTER
en Informatique

Option :

Systèmes Complexes et Technologies de l'Information et du Contrôle

Thème

Le problème d'égoïsme dans les réseaux Mobiles Ad Hoc

Présenté par :

M^r AIT ARAB Sofiane
M^lle KHIREDDINE Anfel
M^r ZAOUCHE Lotfi

Soutenu le : 30/06/2012

Devant le jury composé de :

Président :	*M^lle KHOULALEN* Nadjet	M.A.B université A. MIRA de Bejaia
Rapporteur :	*D^r OMAR* Mawloud	M.C.B université A. MIRA de Bejaia
Examinateurs :	*D^r AMAD* Mourad	M.C.B université A. MIRA de Bejaia
	M^me BEN SAID Samia	M.A.A université A. MIRA de Bejaia

Année universitaire 2011/2012

Remerciements

Nous remercions dieu de nous avoir donné la force et la volonté pour mener à bien notre travail.
Nous remercions grandement Monsieur OMAR Mawloud, sans lequel notre travail et l'aboutissement de ce mémoire, n'auraient pas vu le jour. Merci pour votre confiance, votre disponibilité et vos encouragements.

Sans oublier de remercier les enseignants qui ont appuyé notre cursus universitaire ;

Merci à tous ceux qui nous ont apporté leur aide, chacun à sa manière, tout au long de ce memoire ;

Nos remerciements vont également aux membres du jury. Merci d'avoir accepté de juger notre travail.

Nous conclurons, en remerciant vivement nos familles respectives qui nous ont toujours supporté moralement et financièrement pendant toutes nos années d'études.

Dédicaces

On dédie ce modeste travail :

***A** Mes nos chers parents pour leurs sacrifices et leur affection,*

***A** nos familles,*

***A** tous nos amis(es),*

***A** tous les étudiants de la promotion Informatique de l'université de Béjaïa.*

Table des matières

Liste des figures

Liste des Acronymes

AES	Advanced Encryption Standard.
AODV	Ad-hoc On demand Distance Vector routing.
APNSD	Additional Penalisation for Not Sending Data.
CBRP	Cluster Based Routing Protocol.
DARPA	Defense Advanced Research Projects Agency.
DES	Data Encryption Standard.
DoS	Denial of Service.
DSR	Dynamic Source Routing.
FSR	Fisheye State Routing.
IEEE	Institute of Electrical and Electronics Engineers.
IETF	International Engineering Task Force.
LPR	Low-cost Packet Radio.
MANET	Mobile Adhoc NETwork.
MPS	Most Probable Selfish.
MD5	Message Digest5.
OLSR	Optimized Link State Radio.
PNSD	Penalisation for Not Sending Data.
PPM	Packet Purse Model.
PTM	Packet Trade Model.
RREQ	Route Request.
PRNet	Pakets Radio Network.
RSA	Rivest-Shamir-Adleman.
RSD	Reward for Sending Data.
RSM	Reward for Signaling misbehavior.
RST	Reward for Sending 2hopack.
SHA	Secure hach Algorithme.
SURAN	Survivable Radio Networks.
TBRPF	Topology dissemination Based on Reverse Path Forwarding.
ZRP	Zone Routing Protocol.

Introduction générale

Les communications sans fil ont un rôle crucial à jouer au sein des réseaux informatiques. Elles offrent des solutions ouvertes pour fournir de la mobilité ainsi que des services essentiels là où l'installation d'infrastructures n'est pas possible. Ces réseaux sont en plein développement du fait de la flexibilité de leur interface, qui offre à un utilisateur la mobilité. Cette mobilité est le nouveau mode de communication utilisé. Elle engendre de nouvelles caractéristiques propres à l'environnement mobile tel qu'une fréquente déconnexion, un débit de communication modeste, et des sources d'énergie limitées. Les réseaux mobiles sans fil sont classés en deux catégories : les réseaux avec infrastructure qui utilisent généralement le modèle de la communication cellulaire, et les réseaux sans infrastructure appelés réseaux ad hoc. Un réseau mobile ad-hoc, consiste donc en un grand nombre d'unités mobiles se déplaçant dans un environnement quelconque en utilisant, comme moyen de communication, des interfaces sans fils. Ces réseaux sont auto-organisés et ne reposent sur aucune infrastructure fixe. Les éléments les composant sont en général reliés par radio, potentiellement mobiles, et peuvent être amenés à entrer ou sortir du réseau à tout moment. Ils sont caractérisés par la capacité de chaque participant d'agir à la fois comme client et comme routeur du réseau. Si un émetteur n'est pas à portée directe de la machine destination, les informations devront être transmises de proche en proche, le long d'un chemin établi et maintenu par le réseau. Avec l'émergence de ces dispositifs, de multiples recherches ont été faites dans des Communications mobiles. Ces recherches se concentrent en particulier sur la façon de router les informations entre les nœuds. Compte tenu de la mobilité imprévisible des nœuds, la topologie d'un réseau ad-hoc est instable et est susceptible à de fréquents changements. Router dans de telles conditions devient une tâche complexe, d'autant plus que les ressources énergétiques sont limitées ce qui incite les nœuds légitimes à devenir égoïstes et refuser de router les paquets à fin de préserver leurs énergies.

Il est vrai que les protocoles de routages proposés dans la littérature sont très efficaces quant à la transmission des données, néanmoins leur plus grande faille réside dans le fait que la majorité d'entre eux ne prennent pas en considération les nœuds égoïstes et n'implémentent pas vraiment de mécanisme pour les détecter et les isoler du réseau.

Objectif du travail : Dans ce travail, on s'intéresse à l'étude de quelques approches proposées pour le problème d'égoïsme dans les réseaux mobiles ad hoc. Notre objectif est d'exposer leurs spécificités, caractéristiques, ainsi que leurs modes de fonctionnement, afin de pouvoir aboutir à l'élaboration d'une nouvelle solution qui répondra au besoin de détection et réaction aux nœuds égoïstes.

Plan et contenu du mémoire : Afin de présenter notre projet, nous organisons ce manuscrit en cinq chapitres. Dans le premier on introduit les réseaux mobiles ad hoc et leurs caractéristiques. Le second contiendra une brève présentation de la cryptographie, le troisième un état de l'art sur les solutions proposées pour la problématique de l'égoïsme. Le quatrième quant à lui représente notre approche. Enfin, le Cinquième consiste à faire une simulation de la solution que nous avons proposée.

Chapitre 1

Les réseaux mobiles ad hoc

1.1 Introduction

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calcul portables poussent aujourd'hui les chercheurs à faire de plus en plus d'efforts à fin de pouvoir offrir l'accès à l'information n'importe où et n'importe quand. Au début, le développement des réseaux Ad-Hoc a été le résultat de la demande du milieu militaire pour le déploiement rapide d'infrastructures de télécommunication pouvant survivre aux pannes et aux attaques. Un réseau centralisé autour de stations de base n'est pas une bonne option dans ce milieu car elles doivent être déployées en premier lieu (presque impossible dans un terrain hostile) et le réseau est vulnérable dans le cas où une ou plusieurs de ces stations de base sont détruites. Face à ces limites, en 1972 le département de la défense américaine, en particulier DARPA [1], a sponsorisé le programme de recherche PRNet [2]. Ce projet traitait en particulier la problématique de routage et l'accès au média dans un réseau de communication multi-sauts par onde radio. En 1983, ce projet a évolué vers le programme SURAN dont les objectifs étaient d'augmenter le nombre de noeuds supportés par PRNet dans une zone géographique étendue et réduire la consommation d'énergie en développant de nouveaux algorithmes de routage. Grâce à toutes ces recherches LPR [3] a vus le jour en 1987, il offrait la commutation de paquets, et a aussi amélioré la sécurité et la gestion de la consommation de l'énergie des nœuds. La technologie PRNet est devenue accessible au grand public depuis que les ordinateurs portables ont été équipés de carte sans fils et de ports infrarouges en 1990. L'IEEE adoptait alors le terme "réseaux ad hoc" pour le standard IEEE 802.11 des réseaux locaux sans fil. Beaucoup de standards ont suivi le développement des réseaux Ad hoc, ainsi le groupe de travail MANET a été fondé au sein de l'IETF, il avait pour but d'essayer de standardiser les protocoles de routage dans les réseaux Ad hoc [4].

1.2 Les environnements mobiles

1.2.1 Le modèle des environnements mobiles

Un environnement mobile est un système de sites mobiles qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques. Les réseaux mobiles ou sans fil, peuvent être classés en deux catégories : Les réseaux avec infrastructure et les réseaux sans infrastructure. Comme nous le montre la figure suivante (Figure. 1.1).

Les réseaux avec infrastructure utilisent généralement le modèle de la communication

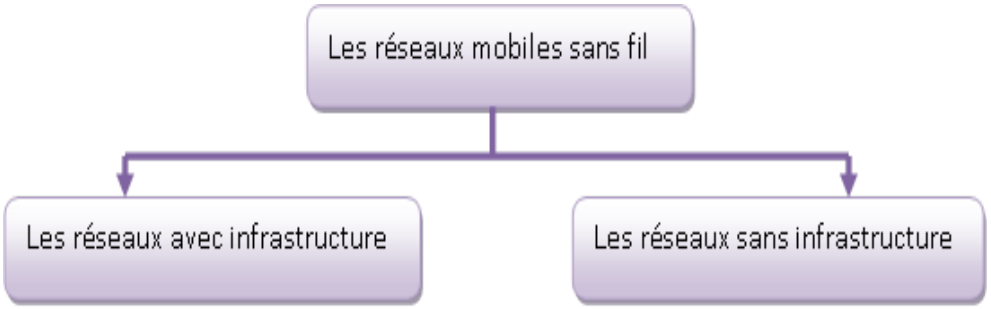

FIGURE 1.1 – Les catégories des réseaux mobiles

cellulaire comme illustré dans la Figure. 1.2. Ce modèle est composé de deux ensembles d'entités distinctes : les sites fixes d'un réseau de communication filaire (wired network), et les sites mobiles (wireless network). Certains sites fixes, appelés stations de base ou point d'accès sont munis d'une interface de communication sans fil pour permettre la communication directe avec les sites (unités) mobiles, localisés dans une zone géographique limitée, appelée cellule. A chaque station de base correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé.

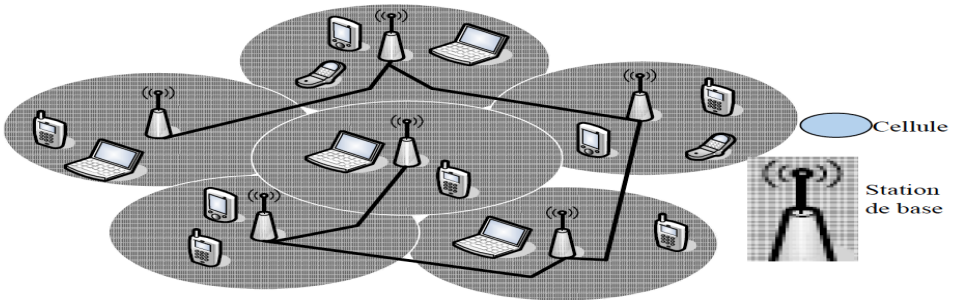

FIGURE 1.2 – Modèle de réseaux mobiles avec infrastructure

La deuxième classe à la quelle nous nous intéressons est la classe des réseaux sans infrastructure préexistantes ou les réseaux mobiles Ad hoc. Ce modèle de réseau ne comporte pas

d'entités fixes. Tous les sites du réseau sont mobiles et communiquent d'une maniére directe en utilisant leurs interfaces de communication sans fil comme le montre la figure suivante.

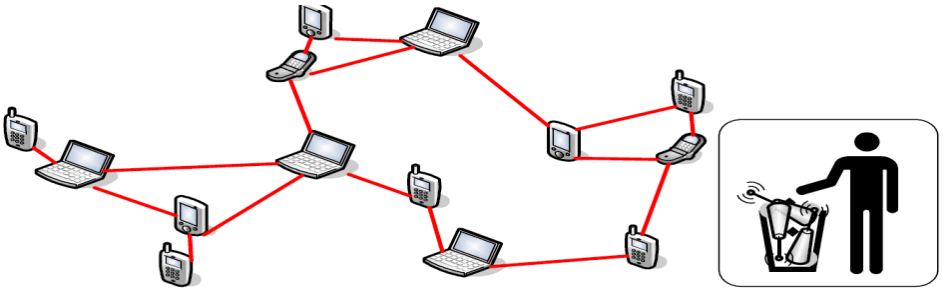

FIGURE 1.3 – Modèle de réseaux mobiles sans infrastructure

1.2.2 Les réseaux mobiles Ad hoc

1.2.2.1 Définition

Un réseau mobile Ad hoc généralement appelé MANET (Mobile Ad hoc NETworks) consiste en un grand nombre d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou une administration centralisée. Ces réseaux sont aussi dynamique dans l'espace et dans le temps et offrent une grande flexibilité.

1.2.3 Avantages des réseaux Ad hoc

Le mode de fonctionnement des réseaux ad hoc présente de nombreux avantages :

- La flexibilité de la topologie.
- La rapidité de déploiement d'un réseau ad hoc : en effet, Il est possible d'utiliser un réseau dans des situations d'urgence, par exemple, pour organiser les secours lors d'une catastrophe naturelle.
- La possibilité de création d'un réseau local mobile : en effet, Un ensemble de personnes se déplaçant peuvent créer un réseau local mobile. On peut, par exemple, imaginer l'utilisation d'un réseau ad hoc pour relier entre eux au cours d'un trajet, les camions d'un convoi routier.
- Facilité d'installations temporaires : telles que les stands de foire, les expositions ou salles de conférences (recours moindre au personnel informatique).

- Installation plus économique du réseau dans les endroits difficiles à câbler.
- Adapté aux environnements dynamiques nécessitant des transformations fréquentes grâce au coût minime du câblage.
- Libération de l'utilisateur de sa dépendance à l'égard des accès câblés au backbone en lui offrant un accès permanent et omniprésent.

1.2.4 Routage dans les réseaux Ad hoc

Le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Il peut arriver qu'une unité mobile veuille communiquer avec une autre qui n'est pas dans sa portée de communication directe. C'est grâce au mécanisme de routage que les unités mobiles formant le réseau vont pouvoir communiquer, même si elles ne sont pas à la portée directe de la communication. Les messages vont donc devoir être transmis de proche en proche jusqu'à la destination. Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être classés en plusieurs catégories : les protocoles proactifs, réactifs ou hybrides. Les protocoles proactifs établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage, alors que les protocoles réactifs cherchent les routes à la demande.
Le problème de routage consiste à déterminer un acheminement optimal des paquets à travers le réseau au sens d'un certain critère de performance (bande passante, délai). Le problème consiste à trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa survabilité en cas de n'importe quelle panne d'arc ou de noeud [5].

1.2.4.1 Protocoles proactifs

Le principe de base des protocoles proactifs est de maintenir à jour les tables de routage, de sorte que lorsqu'une application désire envoyer un paquet à un autre mobile, une route est immédiatement connue. Dans le contexte des réseaux Ad Hoc, les nœuds peuvent apparaître ou disparaître d'une manière aléatoire et rapide, aussi la topologie même du réseau peut changer ; cela signifie qu'il va falloir un échange continuel d'informations pour que chaque nœud ait une image à jour du réseau. Les tables sont donc maintenues grâce à des paquets de contrôle, et il est possible d'y trouver directement et à tout moment un chemin vers les destinations connues en fonction de divers critères. On peut par exemple privilégier les routes comportant un nombre réduit de sauts, celles qui offrent la meilleure bande passante, ou encore celles où le délai de transmission est le plus faible. L'avantage premier de ce type de

protocole est d'avoir les routes immédiatement disponibles quand les nœuds en ont besoin, mais cela se fait au coût d'échanges réguliers de messages (consommation de bande passante) qui ne sont certainement pas tous nécessaires (seules certaines routes seront utilisées par les nœuds en général).
Les protocoles de cette famille se différencient par la manière dont cette information de mise à jour est transmise à travers le réseau ainsi que par le nombre de tables de routage utilisées. Parmi les principaux protocoles proactifs, nous citons :

- OLSR (Optimized Link State Routing) [7].
- FSR (Fisheye State Routing) [8].
- TBRPF (Topology Dissemination Based on Reverse-path Forwarding) [9].

1.2.4.2 Protocoles réactifs

Les protocoles de routage réactifs (dits aussi protocoles de routage à la demande) représentent les protocoles les plus récents proposés dans le but d'assurer le service du routage dans les réseaux sans fil. Les protocoles de routage appartenant à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information spécifiée, inconnue au préalable. Plusieurs approches peuvent être appliquées dans la découverte des routes. La méthode classique de recherche de route consiste à inonder le réseau avec une requête de demande de route " *RREQ* " (Route REQuest). Les nœuds voulant communiquer à travers le réseau lancent des requêtes à la recherche de routes permettant l'acheminement des paquets d'information. Parmi les principaux protocoles réactifs, nous citons :

- AODV (Ad-hoc On demand Distance Vector routing) [10].
- DSR (Dynamic Source Routing) [11].

1.2.4.3 Protocoles hybrides

Les protocoles hybrides combinent les deux approches proactives et réactives. Ils utilisent un protocole proactif, pour apprendre le prochain voisin (par exemple voisinage à deux ou trois sauts) ; ainsi ils 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 routes 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 diriger la requête vers les autres zones sans déranger le reste de sa zone. Parmi les principaux protocoles hybrides, nous citons :

- ZRP (Zone Routing Protocol) [11].
- CBRP (Cluster Based Routing Protocol) [12].

1.2.5 La consommation et optimisation de l'énergie dans les réseaux ad hoc :

Le développement d'un réseau sans fil est freiné par plusieurs problèmes et défis. Ces problèmes débutent de la méthode d'accès et le partage des ressources jusqu'à l'introduction de la QoS et passant par l'adaptation aux conditions du réseau. Le principe fondamental relatif à ces réseaux se manifestant dans l'indépendance de toute infrastructure totalement fixe affronte le problème de l'énergie.

1.2.5.1 Le besoin en énergie

D'une façon générale, l'énergie est un facteur majeur pour les réseaux sans fil et dans le contexte de mobilité du fait que les stations du réseau sont en activité permanente (radio : écoute du canal, réception, transmission au niveau supérieur : routage, traitement de l'information, ...). Cette activité ne pourrait être présente et continue qu'à travers les batteries des stations présentant l'inconvénient d'être de charge limitée.
Néanmoins, cette problématique est plus complexe en mode Ad Hoc que pour les réseaux en mode centralisé. En effet, les nœuds devront être à double fonctionnalités : routeur et nœud de terminaison et donc la consommation d'énergie se manifeste encore plus. En fait, les informations échangées pour n'importe quel protocole de routage proactif ou réactif augmente la charge des nœuds et celle du réseau. Ceci induit une perte additionnelle d'énergie entre la fonction de gestion et de maintenance des routes et celle de transmission. Dans ce cas, plusieurs nœuds par manque de charge ne feront plus partie du réseau formé et seront la cause de sa fragmentation évidente. Il est à signaler que pour les protocoles de routage, l'énergie présente au niveau des nœuds peut être un critère de choix du chemin de la source vers la destination. Ainsi, il est crucial de pouvoir économiser et plus loin minimiser la consommation d'énergie pour satisfaire la présence de tels réseaux le plus longtemps possible.[20]

1.2.6 Vulnérabilité des réseaux Ad hoc

Gayraud et al [14] ont mené une étude d'analyse de risque dans les réseaux ad hoc. Cette étude est une corrélation entre l'analyse des besoins et exigences des réseaux ad hoc et les risques issus de leurs vulnérabilités.
Une liste des attaques les plus probables est ainsi dressée dans cette section.

- **Les attaques de dénis de service (DoS)** : Les attaques DoS apparaissent comme les plus faciles à réaliser par un attaquant, mettant en péril la disponibilité des membres du groupe et plus particulièrement des entités qui jouent le rôle de serveurs ou de contrôleurs au sein du réseau ad hoc. Les attaques DoS les plus connues dans un MANET sont les suivantes :

  - Brouillage du canal radio pour empêcher toute communication.
  - Tentative de débordement des tables de routages des nœuds servant de relais.
  - Non coopération d'un nœud au bon fonctionnement du réseau, dans le but de préserver son énergie. Cette attaque est connue sous le nom de "*Selfishness*" et peut être détectée grâce à des mécanismes de réputation et de détection de comportements égoïstes.
  - Dispersion et suppression du trafic en attaquant les mécanismes de routage. Un attaquant peut alors par exemple injecter des fausses informations ou des paquets redondants dans le réseau. Pour les réseaux Ad Hoc, une attaque du type déni de service peut viser la bande passante, le trafic, les ressources énergétiques des nœuds ou la qualité de transmission (par interférences).
  - Attaque des mécanismes de sécurité eux mêmes.

- **Les attaques passives d'écoute et analyse de trafic** : Ces attaques, appelées aussi "*Sniffing*", consiste à écouter le réseau dans lequel transitent des paquets de données et à récupérer à la volée et illégalement des données qui peuvent être confidentielles. Les attaques d'écoute et d'analyse de trafic sont d'autant plus dangereuses dans les réseaux sans fils tels que les MANETs. En effet, les ondes radio-électriques ont intrinsèquement une grande capacité à se propager dans toutes les directions avec une portée relativement grande, facilitant ainsi à une personne non autorisée, d'écouter le réseau et d'analyser le trafic. Ces attaques constituent une menace certaine pour la confidentialité des données, ainsi que pour l'anonymat des utilisateurs.

- **Spoofing** : Les attaques *Spoofing* sont aussi appelées attaques " *Impersonation* ". Dans ce type d'attaques, l'attaquant essaie de prendre l'identité d'un autre nœud du réseau pour recevoir les messages destinés à ce nœud [15]. Selon le niveau d'accès du nœud impersoné (chef de groupe par exemple), l'attaquant peut reconfigurer le réseau soit pour permettre à d'autres attaquants de rejoindre le réseau ou d'enlever les mesures de

sécurité. L'attaque *Spoofing* peut permettre aussi à l'attaquant d'avoir accès aux clés de cryptage et d'authentification. Selon la couche où l'identité est impersonée, il peut être difficile d'empêcher cette attaque. En exploitant la faiblesse des protocoles de la couche MAC des réseaux Ad Hoc, l'attaquant peut se localiser entre deux nœuds communicants et se prendre pour l'un de ces nœuds (attaque appelée aussi *Man in the Middle*). En général, l'attaque *Spoofing* peut se réaliser en s'appuyant sur les mécanismes de découverte de voisins et de gestion de topologie dans les réseaux Ad Hoc.

- **Attaque Sinkhole** : En réalisant l'attaque *Sinkhole*, l'attaquant vise à attirer les données de tous ses voisins. Puisque ceci donnera accès pour l'attaquant à toutes les données émises ou reçues par ses voisins, l'attaque *Sinkhole* est la base de plusieurs autres attaques (*Eavesdropping* ou *Data Alteration*). L'attaque *Sinkhole* utilise les loopholes (recherche de meilleur chemin) dans les algorithmes de routage dans les réseaux Ad Hoc et l'attaquant se présente à ses voisins comme le meilleur saut d'un chemin de routage multi sauts [15]. L'effet de l'attaque *Sinkhole* est plus grave pour les réseaux multicouches (organisés en groupes) si l'attaquant arrive à impersoner un chef de groupe. [16]

- **Attaque Wormhole** : L'attaque *Wormhole* est étroitement liée à l'attaque *Sinkhole*. Dans l'attaque *Wormhole*, l'attaquant utilise un chemin en dehors du réseau (en utilisant une fréquence hors bande) pour router des messages à un autre nœud compromis [15]. Cette attaque est difficile à détecter puisque le chemin utilisé ne fait pas partie de la bande de fréquences du réseau. Normalement l'attaque *Wormhole* seule n'est pas grave puisqu'elle ne permet que de router des messages de manière plus rapide. Cependant cette attaque a des effets très dangereux sur le réseau Ad Hoc. En effet elle cause des confusions pour les mécanismes de routage et de gestion de topologie du réseau Ad Hoc sans avoir besoin pour l'attaquant de connaître ces mécanismes. [16]

- **Attaque Blackhole / Grayhole** :Les techniques *blackhole* et *grayhole* ont pour but de ne retransmettre aucun ou une partie seulement des paquets reçus. Pour le cas des *grayholes*, le choix des paquets retransmis n'est généralement pas lié aux hasards, le but étant par exemple de favoriser une partie du trafic. Toutefois, il est à noter que cette attaque peut être également confondue avec le fait qu'un nœud est soit surchargé, soit incapable (nœud à faible capacité) de jouer le rôle d'un routeur, ce qui peut compliquer la détection de ce genre d'attaques.

- **Attaque "privatisation de sommeil"** :Ce type d'attaque propre aux réseaux mobiles Ad Hoc repose sur l'idée de demander un service que le nœud visé offre de manière répétitive afin de lui gaspiller sa puissance et ses ressources système et par suite de l'empêcher de "se reposer" [15]. Ceci est très grave pour un réseau Ad Hoc où les nœuds sont caractérisés en général par une batterie à durée de vie limitée. Cette attaque peut

aussi empêcher d'autres nœuds du réseau de demander des données, des services ou des informations de l'entité attaquée. L'attaquant peut alors interagir continuellement avec le nœud afin de lui faire consommer de la batterie. La durée de vie de la batterie est un paramètre critique pour beaucoup d'éléments portables, et tout est fait pour la rendre maximale et l'utiliser le moins possible. [16]

- **Attaque Sybil** : Les nœuds malicieux dans un réseau Ad Hoc peuvent non seulement "*impersoner*" un nœud, ils peuvent aussi assumer l'identité de plusieurs nœuds en déclenchant l'attaque appelée attaque Sybil [15]. Puisque les réseaux Ad Hoc reposent pour fonctionner correctement, sur les communications entre ses différents nœuds, plusieurs systèmes appliquent des algorithmes redondant pour s'assurer que la communication est entre les noeuds source et destination. Par conséquent, les attaquants auront des difficultés pour détruire l'intégrité des informations échangées. Si le même paquet est envoyé sur plusieurs chemins différents (protocoles de routage multi chemins comme *SMR*), le changement de l'un de ces paquets peut être facilement détecté et par suite il sera facile d'isoler l'attaquant. Cependant, si un nœud arrive à représenter l'identité de plusieurs autres nœuds en utilisant cette attaque, l'efficacité de ces mesures est significativement dégradée. L'attaquant peut alors accéder à toutes les informations et à tous les paquets transmis de sorte que le nœud récepteur ne pourra pas détecter le changement du contenu du paquet. Pour le routage à base de confiance, se représenter sur différentes identités peut mener à des faux niveaux de confiance aux nœuds attaquants ce qui leur permettra d'accéder facilement au trafic en cours. [16]

- **Attaque Rushing** :Ce type d'attaque est spécifique aux protocoles de routage à la demande comme le protocole *DSR* (Dynamic Source Routing) [15]. Selon cette attaque, un nœud malicieux essaye de toucher à des messages de demande de chemins (messages *ROUTE REQUEST*) en modifiant le champ de la liste des nœuds et de transmettre ce message au nœud suivant. Puisque pour ces protocoles de routage un seul message *ROUTE REQUEST* dans la découverte de chemin est rediffusé, le nœud malicieux peut s'inclure dans le chemin de la route si son message arrive le premier au nœud suivant. Les attaques *Rushing* peuvent être détectées en évaluant le mécanisme de découverte de chemins. [16]

- **Les attaques physiques d'un élément valide du réseau** : Ces attaques compromettent des nœuds valides du réseau (destruction, altération ou changement physique d'un composant) et s'avèrent être des attaques particulièrement dangereuses dans les réseaux ad hoc.

1.3 Conclusion

Dans ce chapitre, nous avons présenté les réseaux mobiles ad hoc qui sont un type particulier de réseaux sans fil, ne nécessitant aucune infrastructure fixe pour se créer et s'organiser. Malgré les progrès réalisés dans ce domaine, pour atteindre les objectifs de ces réseaux, beaucoup de travail reste à faire. Les caractéristiques de ces réseaux constituent de réels défis. Le caractère fortement dynamique des réseaux ad hoc, nécessite l'implémentation de protocoles plus complexes que ceux des réseaux fixes ou des réseaux avec points d'accès. A cause de leurs vulnérabilités, les réseaux ad hoc sont sujets à de très nombreuses menaces, notamment le problème d'égoïsme qui a fait l'objet de plusieurs recherches. Dans le but d'essayer d'y remédier les chercheurs ont essayé d'élaborer plusieurs approches. Ces approches seront abordées plus en détail dans le prochain chapitre.

Chapitre 2

Etat de l'art sur les solutions existantes

2.1 Introduction

Les environnements mobiles présentent une hétérogénéité importante et une grande variabilité aussi bien au niveau de la capacité de traitement que de l'énergie entre les différents nœuds. Dans de tels environnements, des déséquilibres de charges peuvent apparaître. En effet, un nœud plus puissant en terme de capacité de traitement peut devenir oisif, parce qu'il a vite accompli sa tâche, tandis que les autres moins puissants sont occupés la majorité du temps et consomment plus d'énergie. La capacité des nœuds puissants peut-être exploitée par des nœuds surchargés c'est ce qui constitue le problème d'égoïsme dans les réseaux mobiles ad hoc.
Dans le cas spécifique offert par les réseaux MANET ouverts, il est très simple de manipuler un protocole d'acheminement afin d'économiser l'énergie dépensée par un nœud d'une façon égoïste. Le besoin de coopération entre les nœuds pour assurer le fonctionnement du réseau est en conflit avec l'intérêt individuel de chaque nœud visant à ne dépenser de l'énergie que pour les flux de trafic qui leur sont destinés ou pour les quels ils sont originés.
On peut identifier deux types de nœuds non-coopératifs : des nœuds défectueux/malicieux (faulty/malicious) et des nœuds égoïstes (selfish). Les nœuds défectueux/malicieux appartiennent a la classe des nœuds qui sont soit défectueux et ne peuvent donc pas suivre un protocole bien déterminé ou intentionnellement malicieux et qui essayent d'attaquer le système.[18]
Vu qu'un nœud égoïste est un nœud économiquement rationnel dont l'objectif est de maximiser son propre bien-être, qui est défini comme le bénéfice de ses actions moins leurs coûts, et que la transmission d'un message inflige un coût (de l'énergie et autres ressources) à un nœud ; le nœud égoïste aurai besoin d'incitation (récompense) en vue de transmettre les

messages des autres [19].
Il existe deux catégories pour inciter les nœuds égoïstes dans les réseaux mobiles ad hoc à coopérer : les mécanismes de réputations (reputation mechanism) et les systèmes à base de crédits.
Dans ce qui suit nous allons nous intéresser au problème des nœuds égoïstes dans les réseaux mobile ad hoc, nous présenterons les principales approches déjà proposées dans le but de résoudre ce problème.

2.2 Les différentes catégories des solutions existantes

Les chercheurs ont proposé plusieurs solutions pour lutter contre le problème d'égoïsme dans les réseaux ad hoc qui devient de plus en plus pénalisant pour la robustesse du système et la fiabilité des services qu'il fournit. Ces solutions sont généralement répertoriées en deux grandes catégories : les systèmes à base de réputation et ceux basés sur le crédit.

2.2.1 Systèmes de réputation

Lorsque l'observation seule ne permet pas une mesure directe et objective de la malveillance des nœuds, il est nécessaire que chaque nœud maintienne un degré de confiance en chacun des autres nœuds qu'il a observé. La valeur de ce degré de confiance est influencée par les observations de telle façon qu'elle puisse fournir une estimation la plus juste possible de la malveillance du nœud observé. La littérature fournit un large champ d'application sur les systèmes de réputation. Néanmoins en ce qui concerne le routage pour les réseaux ad hoc, on distingue deux courants principaux, l'un ayant pour but de décourager les comportements malveillants et l'autre de les éliminer. Bien que la nuance semble subtile, dans le premier cas il ne s'agit que de mesures d'encouragement aux comportements bienveillants qui ne peuvent pas empêcher l'action des nœuds malveillants. L'idée est que si un nœud veut lui-même pouvoir transmettre ses propres paquets à travers le réseau, il a tout intérêt à afficher ses bonnes intentions en coopérant dans le réseau. Cependant, il est clair qu'un nœud malveillant n'ayant pas spécialement besoin ou envie de transmettre son propre trafic peut très bien avoir un comportement malveillant. On peut donc voir ces systèmes de réputation comme une variante de système d'encouragement.
Dans le second cas, où le système de réputation est mis en place afin d'éliminer les nœuds dont l'action malveillante a été observée, un nœud agissant de façon nuisible sera exclu du

réseau ou au moins évité du point de vue de la retransmission des paquets. Il s'agit donc dans ce cas là de rendre un réseau ad hoc résistant à l'action des nœuds malveillants. [20]

2.2.2 Systèmes de crédit

L'idée de base des systèmes à base de crédit est, de fournir des incitations aux nœuds pour assurer les fonctionnalités du réseau. A fin d'atteindre cet objectif virtuel (électronique), un système de paiement peut être mis en place. Les nœuds sont payés pour fournir des services à d'autres nœuds et relayer leurs paquets. Ce genre de système peut être implémenté en utilisant deux modèles : le modèle de sac en paquet *PPM*, et le modèle de commerce de paquet *PTM*. [21]

2.3 Approches étudiées

La littérature est riche en propositions de solutions au problème de la résistance des réseaux aux nœuds malveillants (égoïstes). Dans cette section nous allons proposer quelques approches que nous avons étudiées parmi les solutions existantes :

2.3.1 Watchdog et Pathrater

Afin d'éviter d'avoir à effectuer une recherche explicite du nœud fautif, Marti et al. Ont proposé une méthode de détection de fautes basée sur l'écoute des interfaces en mode "*espion*" comme illustrée dans la figure ci-dessous. Il s'agit pour chaque nœud de traiter toutes les trames reçues sur le médium, même si elles ne lui sont pas destinées. En supposant que tous les nœuds ne possèdent qu'une seule interface réseau sans fil, que toutes les interfaces transmettent sur le même canal et avec la même puissance et que toutes les antennes sont omnidirectionnelles, un nœud est capable de vérifier le contenu de ce que chacun de ses voisins transmet. En particulier, lorsqu'un nœud transmet à son voisin un paquet de données destiné à être retransmis par ce dernier, celui-ci peut effectivement vérifier si son voisin a bien retransmis le bon paquet ou pas.
Supposons par exemple qu'il existe un chemin du nœud S vers le nœud D par l'intermédiaire des nœuds B et C, et que A transmet un paquet à C en passant par B. En écoutant le trafic de B, A pourra dire si ce dernier a retransmis le paquet ou pas.

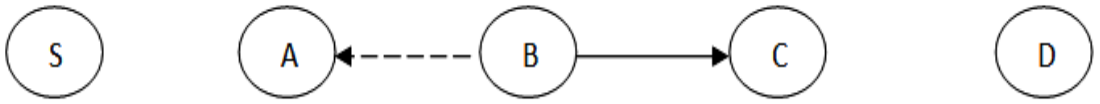

FIGURE 2.1 – L'écoute du trafic par les voisins

Ainsi, on requiert de chaque nœud de vérifier que ses voisins retransmettent correctement les paquets de données qu'il leur envoie. Cette phase est baptisée **watchdog** et consiste à ce que chaque nœud garde en mémoire une copie de chaque paquet retransmis et non destiné à un voisin. À chaque fois qu'une transmission est entendue sur l'interface, le paquet correspondant en mémoire, s'il existe, est retiré. À chaque paquet en mémoire est aussi associée une date de sa retransmission, afin de pouvoir déterminer à quel moment trop de temps s'est écoulé et le paquet doit être considéré comme non-retransmis par le voisin. Lorsque le nombre de non-retransmissions dépasse un certain seuil, une notification est envoyée à la source pour la prévenir de la faute. Les nœuds source peuvent donc maintenir un historique des fautes, sous forme de table de poids par exemple.
L'exploitation des informations recueillies grâce au watchdog est effectuée par un autre élément appelé **pathrater** qui maintient des poids pour les différents nœuds qui ont pour vocation de représenter leur fiabilité constatée, il est utilisé par chaque nœud pour choisir le chemin le plus fiable pour la transmission de leurs paquets. La solution proposée est donc utilisée par-dessus un protocole de routage réactif classique (comme *DSR* par exemple) et permet de disposer d'informations permettant de choisir des chemins fiables. Cependant, il est clair que cette approche ne fournit pas une résistance aux fautes byzantines.
Il existe deux types de fautes byzantines :

- Naturelles : ce sont des erreurs physiques sur une transmission de message ou en mémoire, ou alors des erreurs logicielles qui mènent à une non vérification des spécifications.
- Malicieuses : ce sont des comportements visant à faire échouer le système (sabotage, virus ...).

De l'aveu même des auteurs, la technique du watchdog est faillible en plusieurs points :

- un nœud n'est pas forcément en mesure de capter les transmissions de ses voisins :
  - si les nœuds n'utilisent pas tous le même canal (en utilisant plusieurs interfaces par exemple) ;
  - si les antennes ne sont pas omnidirectionnelles ;
  - si les nœuds utilisent des puissances d'émission différentes, une transmission d'un voisin vers un autre nœud peut être trop faible ;
  - si une collision arrive au niveau du nœud ;

- un nœud malicieux peut essayer de partitionner le réseau en prétendant que les nœuds qui le suivent sur le chemin se comportent mal.
- la possibilité d'échec de transmission des paquets à cause des collisions :
  - A n'entend pas la transmission du paquet " 1 " de B à C, car la transmission de B est en collision avec le paquet " 2 " au niveau de A.

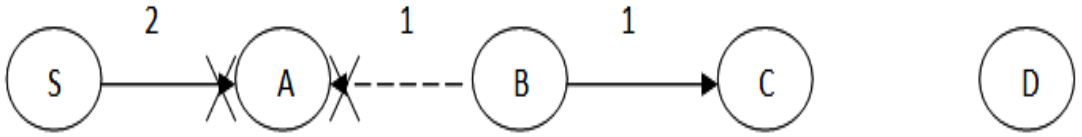

FIGURE 2.2 – 1er cas

  - A croit que B a transmis le paquet " 1 " à C, alors que C n'a jamais reçu le paquet à cause de la collision qui a eu lieu à son niveau avec le paquet 2.

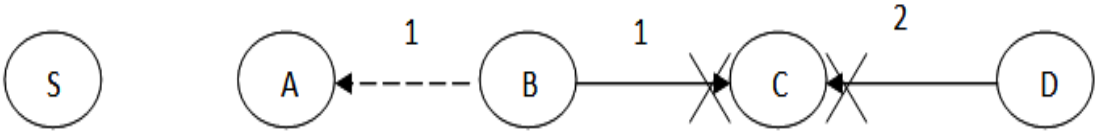

FIGURE 2.3 – 2iéme cas

- le watchdog ne fait pas la différence entre une erreur de transmission et la mauvaise conduite donc elle peut accuser à tord certains nœuds innocents. [21]

2.3.2 Modèle TWOACK

Dans le but de détecter les nœuds égoïstes, les auteurs ont proposé d'utiliser le modèle *TWOACK* [34], qui peut s'ajouter dans n'importe quel protocole de routage. Le principe de ce modèle est simple, un nœud qui transmet/retransmet un message vérifie que le nœud qui le suit a bien accompli sa tâche en retransmettant à son tour le message, et ceci se fait en recevant de la part du nœud à deux sauts à qui le nœud intermédiaire devait transmettre le message, un acquittement spécial appelé *TWOACK packet*. C'est donc le nœud qui reçoit un message qui accuse la réception du message en envoyant un *TWOACK packet* au nœud à deux sauts en arrière sur le chemin du message. Soit trois nœuds N_i, N_{i+1}, N_{i+2} ; N_i envoi un message à N_{i+1} qui devra retransmettre le message à N_{i+2}, ce dernier en recevant le message de N_{i+1}, envoi un *TWOACK packet* à N_i. Cette procédure se répète pour tous les triplets de

nœuds consécutifs qui constituent le chemin qu'empreinte le message .
Afin de détecter le mauvais comportement d'un nœud, l'émetteur maintient à son niveau une liste d'ID des messages dont il n'a pas encore reçu de paquet *TWOACK* d'un nœud à deux sauts, et chaque nœud maintien une liste unique pour chaque lien de transmission qu'il utilise. Chaque élément dans cette liste contient les champs informations suivantes :

- Les récepteur du message dans les deux prochains sauts, notés N_2 et N_3.
- Un compteur de nombre de détection de mal comportement noté C_{mis}.
- ***LIST*** : Une liste d'ID des paquets de données qui ne sont toujours pas acquittée d'un paquet *TWOACK*.

Supposant qu'un nœud N_1 veuille envoyer un paquet en empruntant le chemin N_1 -> N_2 -> N_3, il ajoute l'ID du paquet à *LIST* dans la liste correspondante au lien N_2 -> N_3. Quand N_1 reçoit un paquet *TWOACK*, il retire à *LIST* l'ID du paquet correspondant. Si après un temps dit *TimeOut*, N_1 ne reçoit pas de paquet *TWOACK* pour un certain paquet, alors le lien N_2 -> N_3 sera suspecté, et le compteur C_{mis} de ce lien sera incrémenté. Quand le C_{mis} dépasse le seuil noté *tresh*, le nœud déclare le lien N_2 -> N_3 comme égoïste, et envoie un paquet *RERR*, pour informer la source. Tout nœud recevant ce message mettra le lien N_2 -> N_3 comme égoïste. Ainsi chaque nœud sera en possession des informations concernant les liens égoïstes et pourra ainsi les éviter dans la sélection du chemin de routage de paquets.
Un autre paquet est utilisé quand le nœud N_2 ne peut plus joindre N_3 , soit parce qu'il n'est plus à sa portée ou pour d'autres raisons particulières, dans ce cas le nœud N1 ne va pas incrémenter le C_{mis}.

Le modèle *S-TWOACK* :

Ce modèle n'est qu'une amélioration du modèle *TWOACK*, qui vise à réduire la congestion du réseau causé par le nombre important de paquets *TWOACK* émis. S'inspirant du principe de la fenêtre glissante, *S-TWOACK* n'accuse pas la réception d'un seul message, mais d'un certain nombre de messages bien reçus.
Ce modèle possède deux nouveaux paramètres par rapport au précédent : *timeout-Last-Sent* (le timeout du dernier paquet envoyé) et *maximum-IDs-Carried* (le nombre maximum de paquet envoyés sans être acquités). Quand le nombre de messages reçus par un nœud N_3 atteint la valeur *maximum-IDs-Carried* ou que le *timeout-Last-Sent* expire, le nœud N_3 envoi à N_1 un paquet *TWOACK*. [34]

Le modèle *2ACK* :

Le modèle *2ACK* [22] diffère du modèle *TWOACK*, dans le faite qu'il n'acquitte pas tous les messages reçus, mais seulement une partie. De plus *2ACK*, intègre un mécanisme de certification pour la sécurité des ses paquets.
Il n'est pas non plus semblable au *S-TWOACK*. En effet, il y a une différence subtile, qui est le faite que le *S-TWOACK* accuse la réception d'un certain nombre de paquets, alors que le *2ACK* n'accuse que la réception d'un seul paquet. Selon les auteurs, cela permet d'offrir un

meilleur compromis entre les performances du réseau et le coût.
Ce modèle apporte un plus dans la détection de comportement égoïste dans les réseaux ad hoc. En effet, en comparent au modèle de Mirati et al. Où l'émetteur reste à l'écoute de son voisin pour voir s'il a émis le message (watchdog), où on a le risque de mal évaluer le comportement du nœud à cause des collisions, *TWOACK* évite cela grâce aux acquittements. Aussi, il permet la réintroduction des nœuds exclus à cause de leurs comportements après un certain temps.
Néanmoins, ce modèle possède quelques inconvénients, par exemple le fait qu'on ne sache pas quel est le nœud qui se comporte de façon égoïste, tout ce qu'on peut dire c'est que le lien entre deux nœuds n'est pas fiable. En effet la non réception du paquet *TWOACK* du nœud N_{i+2}, peut-être dû soit au fait que N_{i+1} n'a pas accomplis sa tâche en envoyant le paquet à N_{i+2}, ou bien c'est N_{i+2} qui a ignoré le message de N_{i+1}, et donc il n'enverra pas de *TWOACK* à N_i.
Ce modèle ne permet pas de punir les comportements égoïstes puisqu'il ne permet pas de savoir quel est le nœud qui a un tel comportement, ce qui est un avantage pour ce dernier qui verra ses paquets servis alors que lui conservera son énergie au détriment du réseau.
Un nœud égoïste peut facilement utiliser les paquets *RERR*, pour tromper le nœud émetteur. Dans le meilleur cas, il va juste dire qu'il ne peut plus joindre le nœud suivant. Mais dans le pire cas, il pourrait accuser le lien suivant comme étant égoïste, sous prétexte qu'il ne reçoit pas de packet *TWOACK* à chaque fois qu'il émet le message. Ceci est un bon stratagème qui pourra lui éviter de gaspiller de l'énergie et d'être soupçonné d'être égoïste.
L'ajout des paquets *TWOACK* est une charge supplémentaire pour le réseau. En effet pour chaque triplet des nœuds consécutifs sur le chemin d'un message, on aura au moins un packet *TWOACK*. Mais pour ce problème les auteurs ont quand même proposé une amélioration du modèle pour y remédier en proposant le *S-TWOACK* (Selective-TWOACK), qui consiste à acquitter par un seul paquet *TWOACK* plusieurs messages, c'est-à-dire que le nœud ne va envoyer le paquet *TWOACK* qu'après avoir reçu un certain nombre de messages, réduisant ainsi la congestion du réseau engendré par le flux de paquet *TWOACK*.

2.3.3 L'utilisation d'un système de cache pour détecter les nœuds égoïstes dans les réseaux mobiles ad hoc

Dans le schéma de détection assistée par le matériel, le matériel est responsable de détecter la mauvaise conduite du logiciel, et de faire un rapport pour tous les autres nœuds du réseau. Ce système contient donc, une unité de mémoire cache ainsi que quelques compteurs sachant que le cache stocke les informations sur les identités des paquets reçus récemment, et utilise ces informations pour différencier les paquets dupliqués des paquets originaux.

Un nœud mobile peut recevoir la même demande d'itinéraire plusieurs fois, cela est dû au fait de la diffusion lors du processus de découvertes des routes comme le montre la figure suivante :

Quand un nœud A reçoit une demande de route, il diffuse cette demande à tous ses voisins

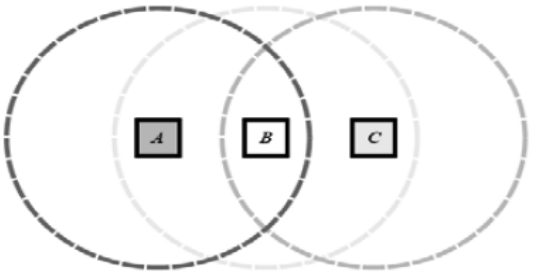

FIGURE 2.4 – " Diffusion de la demande de route reçue par A "

qui à leurs tours la diffuse encore une fois, par conséquent le nœud A recevra à nouveau cette demande. C'est la ou le cache intervient pour reconnaitre les paquets originaux des paquets dupliqués.

Il y'a 4 compteurs utilisés dans ce système : TC (Total Counter), DC (Drop counter), TDC (Total Data Counter), DDC (Data Drop Counter). Les deux premiers compteurs sont utilisés pour détecter l'abandon simple, contrairement aux deux derniers qui sont utilisés pour détecter l'abandon sélectif.

TC : est utilisé pour enregistrer le nombre total de paquets uniques reçus (les demandes de routes) ;

DC : est utilisé pour enregistrer le nombre total de paquets uniques abandonnés par un nœud ;

TDC : est utilisé pour enregistrer combien de paquets de données ont été reçus par un nœud ;

DDC : est utilisé pour enregistrer combien de paquets de données qui ont été abandonnées par un nœud.

Chaque élément du cache dispose de 4 champs : *un drapeau de validité*, *adresse source*, *adresse destination* et *le numéro de la demande*. Lorsque le matériel reçoit une demande d'itinéraire, il interroge la table de cache si la même demande a déjà été reçue en utilisant les adresses sources et destination comme indices. S'il y'a une correspondance dans la table de cache, le numéro de demande de ce paquet est comparé avec le numéro de la demande stockée dans la table. Si le numéro de la demande entrante est supérieur à celui de la demande stockée alors la requête de route est originale, dans le cas contraire elle est en double.

Si la requête est d'origine la table est mise à jour en ajoutant cette demande, et les compteurs *TC* et *DC* sont incrémenté d'une unité. Lorsque la table est pleine et qu'il y'a un nouvel élément qu'on doit ajouter, un algorithme simple et efficace est alors utilisé : la stratégie du premier entré-premier sorti.

Si le paquet entrant est un paquet de donné, il n'est pas nécessaire d'interroger la table de cache et les compteurs TDC et DDC sont augmentés d'une unité. Pour chaque paquet sortant, les compteurs sont traités selon le type de paquet reçu. Par exemple pour un paquet de demande de route le compteur DC est décrémenté de 1, par contre pour un paquet de donnée DDC diminue de 1.
A un certain moment, si le rapport " $DC/\ TC$ " ou " $DDC/\ TDC$ " dépasse un certain seuil prédéfini, le matériel de détection envoi un message d'avertissement aux nœuds du réseau. Pour diminuer le " *faux positif* " (l'accusation d'un nœud à tord), DC et DDC doivent être supérieur à la valeur fixée à la base avant que le nœud puisse être inculpé d'égoïste. Les nœuds recevant ce message d'avertissement peuvent réduire la note (évaluation) du nœud inculpé ou même l'isoler.
Le schéma proposé dans cette solution est un bon moyen pour la détection des nœuds égoïste, néanmoins il nécessite certaines améliorations dans le but de réduire le taux des nœuds égoïste dans un réseau. L'une des modifications apportées à ce schéma et d'ajouter une minuterie qui sera utilisée pour donner une pénalité supplémentaire à un nœud qui ne retransmet pas une demande d'établissement de route. La pénalité (Penalty Timer : PeT) débute quand une demande d'itinéraire initiale est reçue. Si le nœud ne retransmet pas une requête de route au cours de la période de PeT ; une pénalité supplémentaire est ajoutée à DC. Cependant, une demande d'itinéraire de route en double (duplicata) ne déclenchera pas l'initialisation de PeT.
L'efficacité de détection des nœuds égoïstes : elle détermine l'efficacité du système proposé dans cette solution à découvrir les nœuds malveillants. Elle se calcule comme suit :

Efficacité_détection = (nœuds_détectés / nombre total_nœuds_malveillants) *100.

Sachant que " nœuds_détectés " représente les nœuds malveillants qui ont été inculpés ; Certains nœuds légitimes peuvent être accusés à tord, c'est pour cela que les auteurs ont été dans l'obligation de définir une métrique : " *faux-positif* " qui est calculé de la sorte :

Faux-positif = (nœuds_accusés_à tord / nœuds_détectés)*100.

La détection parfaite aurait un résultat de 0% dans le calcul de Faux-positif et 100% dans Efficacité_détection mais cela reste difficile à obtenir, voire même impossible. [23]

2.3.4 Le système CONFIDANT

Le système CONFIDANT, proposé par Buchegger et Le Boudec [24] a pour but de détecter et exclure les nœuds malveillants dans un réseau ad hoc. Il est composé de plusieurs modules :

- **Le contrôleur** : son rôle est de collecter des informations de première main sur le comportement des nœuds dans le réseau. En écoutant le canal radio, ce module sert à vérifier autant que possible si les voisins se comportent bien en termes de participation au protocole de routage et de retransmission des paquets. Les observations servent à classifier directement un nœud comme bienveillant ou malveillant.
- **Le système de réputation** : il se charge de combiner les informations de seconde main avec les informations de première main en une réputation locale sur chacun des nœuds qui sert à son tour à décider si un nœud doit être considéré comme malveillant.
- **Le gestionnaire de confiance** : c'est le module qui décide à quel moment un message d'alarme doit être envoyé aux autres nœuds de confiance afin de les avertir du comportement malveillant d'un nœud. C'est aussi lui qui décide si le contenu d'un message d'alarme doit être considéré ou ignoré.
- **Le gestionnaire de chemins** : il manipule la topologie vue par le nœud en fonction des degrés de confiance des autres nœuds afin d'exclure les nœuds malveillants du réseau.
- **Le protocole de routage** : il exploite l'information de la topologie pour trouver des chemins valides et fiables. Accessoirement, il peut se baser sur le système de réputation pour refuser de router des paquets en provenance des nœuds malveillants.

Dans leur version préliminaire du système, les auteurs ne précisent pas quels sont les critères de décision pour qu'un nœud ai une bonne réputation. Il est juste évoqué que les degrés locaux de malveillance détectés doivent croître avec les observations de fautes et décroître avec le temps afin de permettre aux nœuds de se racheter ou bien aux fautes rares d'avoir peu d'incidence. Par ailleurs, le protocole de routage envisagé est *DSR* et ses mécanismes réactifs sont exploités pour la recherche de chemins qui évitent une liste de nœuds suspects désignés par la source. Dans un rapport de recherche ultérieur [25], les auteurs donnent une description plus détaillée du système en précisant la méthode utilisée, basée sur un modèle bayésien de représentation [26, 27], pour la maintenance des réputations. Les nœuds intègrent les observations directes des comportements malveillants et les réputations des voisins (observations indirectes) dans leur degré de réputation des autres nœuds. Une observation indirecte est intégrée à la condition que le degré de confiance du nœud émetteur soit suffisant. Ce dernier est maintenu en fonction de la compatibilité des observations indirectes annoncées par le nœud émetteur et les observations directes du nœud courant. Ainsi, les annonces des nœuds qui diffèrent trop d'observations directes locales sont finalement ignorées, afin de se prémunir contre la diffamation.

2.3.5 Topologie véridique de contrôle de nœuds égoïstes dans les réseaux mobiles ad hoc

Vu que le comportement égoïste dans les réseaux ad hoc est inévitable à cause de l'absence de pouvoir de réglementions du système et la rareté des ressources, l'article [28] étudie le problème de contrôle de topologie de MANET dans un environnement non coopératif. En effet, en raison de la réserve d'énergie limitée d'un nœud du réseau, les économies d'énergie sont essentielles pour maintenir la convivialité d'un MANET. Les algorithmes de contrôle de topologie permettent aux nœuds de réduire leur puissance d'émission tout en gardant la même connectivité du réseau. Le mécanisme de contrôle de topologie présenté s'intitule "*TRUECON*", c'est un mécanisme de révélation directe pour attaquer l'intention égoïste dans lequel chaque nœud a une fonction d'utilité quasi-linéaire pour minimiser sa puissance d'émission nominale.
TRUECON a besoin de connaitre la direction à partir de la quelle le message est reçu. Cette information peut être obtenue en utilisant des antennes directionnelles. Un nœud U diffuse périodiquement des messages *HELLO* dans le quel il met la valeur courante de sa puissance d'envoi P_s. A la réception du message *HELLO*, un nœud V_i mesure la puissance du signal reçu P_{ri}, et calcule la puissance minimum P_{u,v_i} nécessaire à U pour parvenir à lui. V_i informe U en lui répondant alors avec un *ACK-message* contenant une valeur P'_{u,v_i} basée sur P_{u,v_i} et qui ne lui est pas égal. A la réception des *ACK-messages*, le nœud U ajoute chaque voisin qu'il lui a envoyé un acquittement à l'ensemble N_u. Le nœud U réduit sa puissance de transmission de tel sorte à rester en contacte avec un ensemble de ses voisins restreint mais suffisant pour garder la connectivité du réseau et réduire le brouillage causés par l'envoi des messages *HELLO*, Mais cette puissance change entre deux cycles d'envoi de messages *HELLO*, vu que l'ensemble N_u change à cause de la mobilité des réseaux ad hoc. Chaque nœud de cet ensemble est appelé " *nœud de transfert* ". Ce mécanisme peut aussi être intégré avec un protocole de routage comme *DSR* pour trouver le chemin minimisant la puissance de transmission tout en gardant la propriété d'incitation, quoique le paiement tout au long de ce chemin doit être plus élevé que le coût actuel afin de donner une incitation aux nœuds de transfert à router les paquets sachant que les prix des services doivent représenter la quantité d'énergie dépensée par les fournisseurs de ces services. Après avoir intégré DSR dans *TRUECON*, *DSR-TRUECON* implémente un algorithme distribué de *BELLMAN-FORD* pour trouver le plus court chemin dans un graphe pondéré. Il permet de découvrir le chemin le plus économe en énergie et il transfère les informations de payement. La complexité du système *DSR-TRUECON* est de O (n^2) sachant que n est le nombre de nœuds dans le réseau. Il est vrai que *TRUECON* est une méthode proactive intéressante visant à réduire la

consommation d'énergie et les interférences radio, pour stimuler la coopération à fin de découvrir une topologie du réseau efficace en ressource, néanmoins cette technique a sa limite car il est difficile de trouver un remplaçant pour chaque nœud de transfert en cas de déconnexion. Aussi, la mobilité des noeuds n'a pas été prise en compte dans cette solution. [28]

2.3.6 Le modèle SPRITE

Dans [19], les auteurs ont proposé une solution intéressante basée sur système de crédit dite *SPRITE*, elle est énoncée comme suit : Lorsqu'un nœud reçoit un message, il garde un reçu de ce message, ensuite quand il disposera d'une connexion rapide à la *CCS* (Credit Clearance Service ou service de compensation de crédit) il lui rapporte les reçus des messages qu'il a reçus / transmis. La CCS détermine ensuite la charge et le crédit à chaque nœud impliqué dans la transmission du message, selon les reçus rapportés. L'avantage majeur de cette solution réside dans le fait d'être une solution entièrement logicielle, néanmoins, cette solution soulève deux problèmes majeurs :
Premièrement, puisque c'est aux nœuds égoïstes de rapporter leurs reçus à la *CCS*, un nœud égoïste (ou même un groupe de nœuds complotant ensemble) peut très bien essayer de tromper le système dans le but de maximiser leurs gains. Par exemple, un nœud égoïste peut comploter avec d'autres nœuds pour forger de faux reçus.
Deuxièmement, un nœud devrait recevoir suffisamment de crédit en transmettant un message pour un autre nœud, de sorte qu'il puisse transmettre ses propres messages avec le crédit reçu, sauf si la ressource du nœud lui-même est extrêmement faible. La manière dont les charges et les crédits sont déterminés par la CCS est modélisée en utilisant la théorie des jeux.

Pour mieux comprendre, les auteurs ont d'abord commencé par répondre aux questions suivantes :

- Quel nœud paye quel autre nœud ?
- Quel est le principe du schéma de payement ?

La réponse à la première question, est que le nœud source qui paye les autres nœuds à travers la CCS pour transmettre son message.
Pour la deuxième question, Le système ne cible pas un payement équitable i.e. il n'est pas exigé que la charge totale du nœud source soit égale au crédit total reçu par les autres nœuds. Si le réseau est large, la CCS donne périodiquement à chaque nœud mobile une quantité de crédit fixe.
Les actions de triche :
Un nœud peut tricher de 3 manières différentes :

- Après la réception du message, le nœud forge le reçu, mais ne transmet pas le message ;
- Le nœud a reçu un message, mais ne rapporte pas le reçu à la CCS ;
- Le nœud ne reçoit pas de message, mais clame le contraire.

Pour le premier cas, la CCS détermine le dernier nœud qui a reçu le message, ensuite demande au nœud source de payer un prix β à ce nœud, et α à tous ses prédécesseurs, tel que $\alpha < \beta$. De cette manière, les nœuds ont beaucoup à gagner en transmettant les messages beaucoup plus qu'en les supprimant.
Pour le deuxième cas, Le dernier nœud(ou les k derniers nœuds) qui n'a jamais reçu le message peut comploter avec le nœud source (en effet si le dernier nœud ne reporte pas son reçu a la CCS, alors le nœud source gagnera un prix α tandis que le dernier nœud perd un prix β, cependant si le nœud source donne au dernier nœud une compensation de $\beta + \xi$, tel que $\xi > 0$, le dernier nœud sera comblé tandis que le nœud source jouira d'un gain net de α-($\beta + \xi$)), ainsi le groupe complotant obtient un gain net de α - β. Pour y remédier, la CCS charge le nœud source d'un supplément si la destination ne rapporte pas son reçu du message, ce supplément va à la CCS au détriment de tous les nœuds. La charge globale du nœud source (y compris le paiement d'autres nœuds et les suppléments) devrait être kβ moins que la charge du nœud source quand le message arrive à la destination, où k est le nombre de nœuds ne soumettant pas des reçus.

Pour le troisième cas, si la destination ne rapporte pas le reçu du message, on multiplie le crédit payé pour chaque nœud par y tel que $y<1$. De ce fait tout les nœuds ont intérêt à ce que le message arrive à la destination.

Dans cette approche, les auteurs ont effectivement répondu à toutes les questions et ont proposé une approche qui incite les nœuds à coopérer dans le réseau. Néanmoins, elle comporte quelques failles et inconvénients comme dans le cas ou un nœud destination complote avec des nœuds intermédiaires. En effet puisque les nœuds rapportent des reçus au lieu du message lui-même, la destination peut bien rapporter un faux reçu à la CCS alors qu'elle n'a pas reçue la totalité du message. De plus, le besoin d'une autorité centrale (CCS) accessible via Internet pour gérer les crédits est une contrainte très dure dans les réseaux mobiles ad hoc qui n'est pas toujours réalisable.

2.3.7 Protocole d'incitation équitable (FIP : Fair Incentive Protocol)

Pour réaliser la stratégie d'incitation basée sur le crédit, il faut prendre en considération ces hypothèses :

- Chaque nœud doit avoir un ID unique différent de zéro, une paire de clé (privé/publique) certifiée et peut supporter les opérations cryptographiques.
- Chaque nœud mobile doit avoir un compte pour stocker ses crédits, qui seront utilisés pour payer l'aide d'autres nœuds à transmission leurs paquets. Généralement un nœud mobile peut obtenir des crédits soit en l'achetant avec de l'argent réel ou alors en le recevant d'autres nœuds à qui il a transmis des paquets. Comme pour les cartes de crédit dans la vie réelle, un nœud est autorisé à demander un service en premier, puis réaliser l'opération avec le service *TCCS* (Trusted Credit clearance service) plus tard.
- TCCS est fiable et possède une paire de clés publiques et privées qui effectuent les opérations d'autorisation de crédit pour les nœuds mobiles.
- La communication entre les nœuds mobiles est bidirectionnelle c'est-à-dire que deux nœuds à l'intérieur de la portée de transmission sans fil peuvent communiquer directement l'un avec l'autre.
- Un nœud mobile peut rapporter au *TCCS* l'attestation de crédit à travers un canal rapide et sécurisé.

Objectifs en matière d'équité : Dans un MANET égoïste avec une stratégie d'incitation, certains nœuds peuvent rester égoïstes et causer par conséquent des problèmes injustes quant à la transmission des paquets. Ceci peut arriver dans les deux cas suivants :

- Après que les nœuds intermédiaires transmettent les paquets du nœud source jusqu'au nœud destination et que les nœuds sources et destination refusent de payer les nœuds intermédiaires.
- Les nœuds intermédiaires qui ont obtenu du crédit des nœuds sources sont réticents quant à la transmission de leurs paquets.

Dans les deux situations, le résultat sera injuste et peut produire des conséquences fatales sur la coopération du système. C'est pour cela que les auteurs ont défini dans cette approche, un principe d'atomicité qui dit que les nœuds intermédiaires ne recevront pas de crédit jusqu'à ce que le nœud de destination reçoive le paquet. Le schéma proposé dans cette solution montre la relation entre l'incitation et l'équité. Ce schéma contient 4 états :
Etat0 : n'a pas de problème d'incitation ni d'équité, tous les nœuds mobiles sont égoïstes et ne coopèrent pas ainsi l'ensemble du réseau est en dégradation.
Etat1 : est juste et l'ensemble du réseau est dans le statut " *coopération optimale* ". Tous les noeuds se comportent de maniére correcte grâce à la stratégie d'incitation et le principe de

l'équité.

Etat2 : est injuste, même si la stratégie d'incitation a été injectée les nœuds source et destination restent égoïstes. En raison de cette injustice, le réseau n'atteindra jamais le statut "*coopération optimale* ".

Etat3 : est également injuste, le caractère abusif est du à l'égoïsme des nœuds intermédiaires. L'ensemble du réseau est en dégradation.

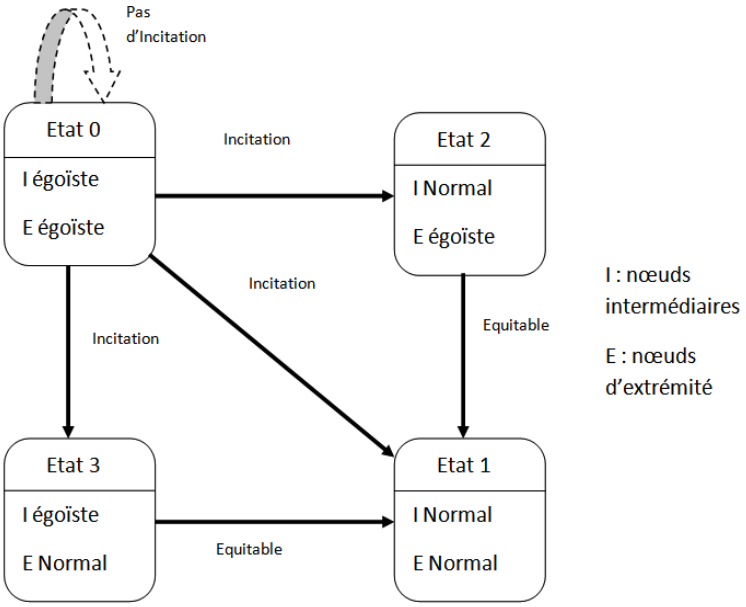

FIGURE 2.5 – les objectifs d'équité

Basée sur la définition de ces statuts, les objectifs d'équité dans cette approche visent à fournir une stratégie efficace pour repousser l'injustice de l'état 2 et l'état 3 à l'état optimal de coopération de l'état 1. [21]

2.3.8 Sessions-based Misbehavior Detection Protocol (SMDP)

Cette solution se compose de deux étapes liées : la phase de surveillance dans la quelle les nœuds surveillent leurs voisins directes lors de la transmission des paquets et la phase de décision dans la quelle les nœuds décident sur le comportement de chaque nœud surveillé en se basant sur le résultat de l'étape précédente : [21]

1. **Phase de surveillance** : dans SMDP, chaque nœud sur la route surveille l'ensemble de ses voisins directs et vérifie s'ils transmettent les paquets correctement ou pas. Une

session est définie ou le trafic est envoyé continuellement du nœud source jusqu'au nœud destination et le protocole de routage doit être conscient du début et la fin de cette session.

2. **Phase de décision** : la méthode de surveillance permet aux voisins de déterminer si chaque nœud surveillé lors de la session a transmis les paquets correctement ou non.

2.3.9 The Packet Purse Model (PPM)

Dans ce modèle, l'initiateur de la charge est partitionné entre les nœuds terminaux d'expédition de la manière suivante : lors de l'envoi du paquet, l'initiateur le charge avec un numéro de grains suffisant pour atteindre la destination. Chaque nœud terminal de transfert acquiert un ou plusieurs grains de ce paquet ce qui accroît le stock de ses grains, le nombre de grains dépend de la connexion directe sur la quelle le paquet est transmis (une longue connexion nécessite plus de grains). Si le paquet n'a pas suffisamment de grains pour être transmis, alors il sera rejeté. La Figure 3.2 illustre le principe de cette solution :

Le problème avec cette solution est qu'il pourrait être difficile d'estimer le nombre de

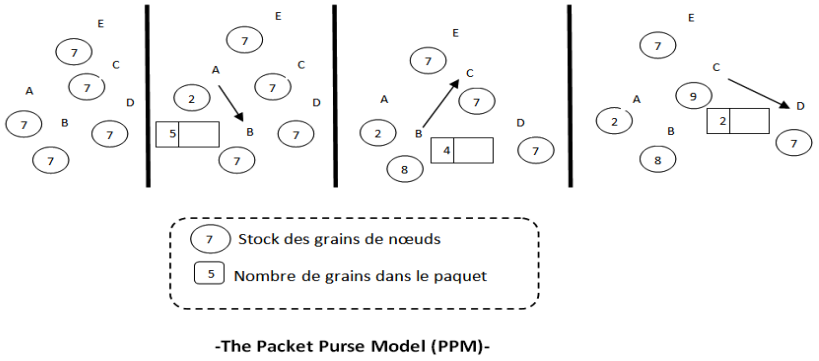

FIGURE 2.6 – - The Packet Purse Model (PPM)-

grains qui sont nécessaires pour atteindre une destination donnée. En effet, si l'initiateur sous-estime le nombre, le paquet risque d'être rejeté, par conséquent l'initiateur va perdre son investissement dans le paquet. Dans le cas contraire s'il sur estime le nombre, le paquet arrivera bien à la destination, néanmoins l'initiateur va perdre le reste des grains. [21]

2.3.10 The Packet Trade Model (PTM)

Dans cette approche, le paquet ne comporte pas de grains, mais il est négocié par des nœuds intermédiaires. De cette façon, chaque intermédiaire qui a fourni un service en transmettant le paquet, augmente le nombre de ses grains et le cout total de la transmission du paquet est couvert par la destination finale.

L'avantage de cette approche est que l'initiateur n'a pas à savoir à l'avance le nombre de

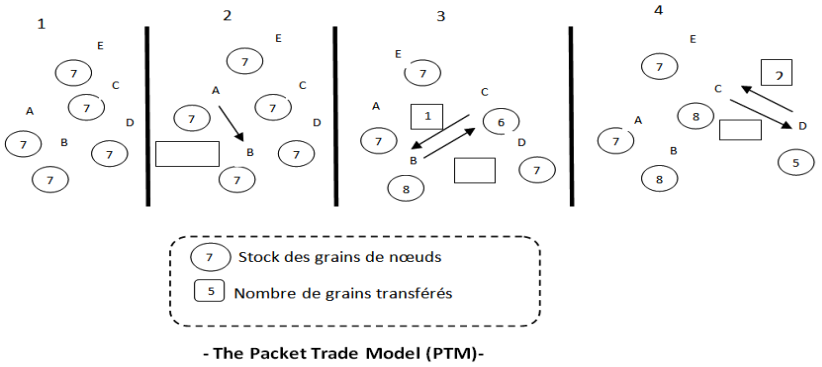

FIGURE 2.7 – - The Packet Trade Model (PTM)-

grains nécessaire pour livrer un paquet. En outre, le fait de laisser le nœud destinataire payer le routage des paquets, rend cette approche applicable en cas de paquet multicast. L'inconvénient de cette approche est qu'elle ne prévient pas directement l'utilisateur de l'inondation du réseau. [21]

2.3.11 Prévenir l'égoïsme dans les réseaux mobiles ad hoc

Dans [29], les auteurs ont proposé le premier modèle qui combine entre le principe de réputation et celui du crédit pour palier au problème d'égoïsme.

Ils ont proposé un modèle où chaque nœud maintient à son niveau une structure de données pour tous les autres nœuds contenant un certain nombre de champs décrivant son impression pour chacun d'eux. Cette structure de données, notée $status_i[j]$, est maintenu par le nœud i pour le nœud j, contient les champs suivants :

- **credits** : un entier signé, qui est incrémenté quand i retransmet un message de j, et décrémenté lorsque c'est j qui retransmet un message de i.
- **maxCredits** : la borne supérieure de crédit.
- **State** : valeur parmi OK, SUSPECTED, OTHERS SUSPECTED, SELFISH.
- **notSelfishTo** : liste des nœuds aux quels le nœud j est disposé à fournir un service

- **selfishTo** : liste des nœuds aux quels le nœud j ne fournit pas de service.
- **dead** : un entier positif incrémenté périodiquement par un compteur, s'il atteint un certain seuil, le nœud i cesse de considèrer le nœud j comme son voisin. Ce compteur est remis à zéro à chaque fois que le nœud i entend le nœud j envoyer un message.

De plus, chaque nœud maintient deux listes *amSelfishTo* et *amNotSelfishTo* qui contiennent respectivement les adresses des nœuds aux quels il est prêt à transmettre des messages et ceux à qui il refuse, plus une liste *pending-messages* qui garde l'ensemble des messages envoyé et n'étant toujours pas acquittés. Aussi, une variable entière *Sc* qui est la somme des crédits des toutes les structures de données.
Périodiquement, chaque nœud fait un broadcast d'un message nommé *selfState* comportant les données suivantes : *notSelfishTo*, *selfishTo* et *knownSelfishNode* qui est une liste de nœud que i considère comme égoïste, mais qu'il n'apparaît pas dans leur liste *selfishTo*.
Quand un nœud i reçoit un message m du nœud j, il vérifie s'il est le destinataire final, sinon il vérifie que j n'appartient pas à *selfishTo* et que le nœud prochain k n'est pas égoïste, et il envoie m à k. le tuple (k, m) est ajouté à *pending-messages*, et i incrémente status_i[j].credits. Dans le cas où j appartient à *selfishTo* ou que status_i[j].credits $>$ maxCredit un message *selfState* sera envoyé et la variable dead sera remis à zéro.
Dès qu'un nœud reçoit un message *selfState*, il remet à zéro la variable dead, et passe à la consultation des données que contient le message. Les listes *notSelfishTo* et *selfishTo* servirons à mettre à jour la structure status_i[j].*notSelfishTo* et status_i[j].*SelfishTo*. La liste *knownSelfish-Node* sert à un nœud d'avertir les autres qu'un nœud k refuse de lui transmettre ses messages, alors que j n'est pas dans la liste *selfishTo* du nœud k. le nœud j recevant ce message mettra l'adresse du nœud i dans status_j[k].*selfishTo* et l'enlèvera de status_j[k].*notSelfishTo*, et marque status_j[k].state = others-suspected.
Cette approche est très intéressante et très sophistiquée pour détecter et traiter les comportements égoïstes. Un nœud est marqué comme égoïste s'il a atteint le crédit maximum qu'on offre, ou qu'il ne transmet pas les messages de ses voisins, ou alors que le nombre de nœud qui le considère comme égoïste à atteint un certain seuil.
L'un des inconvénients de cette approche est qu'un nœud peut passer en mode silencieux, pour ne plus être détecté par ces voisins et ainsi la variable dead aura atteint un seuil, et les messages qui sont en attente d'acquittement de ce nœud seront effacés et aucune action ne sera prise contre lui car ses voisins le considéreront comme hors de portée.

2.3.12 Le système LARS

Dans un rapport de Hu [30], les systèmes CONFIDANT et CORE sont passés en revue de façon critique et de nombreux points faibles y sont dénoncés. Par ailleurs, ce rapport propose

le système *LARS* qui est une forme de système de réputation où seules les observations de première main sont diffusées. Un mécanisme de chiffrement à clés publiques est déployé et pour éviter la diffamation, chaque diffusion d'observation négative doit nécessairement être signée par la clef privée du nœud incriminé. Pour rendre cette signature possible, la cryptographie à seuil est utilisée. Chaque nœud distribue des parts de sa clé privée à un certain nombre de ses voisins qui collectivement deviennent capables d'effectuer l'opération de signature à sa place. Ici aussi un système de watchdog est utilisé pour détecter les non-retransmissions ou d'autres fautes. Le protocole de routage préconisé est *DSR* ou *AODV* et un système d'envoi d'accusés de réception par la destination pour tout paquet de données est nécessaire. Lorsque la source ne reçoit pas un accusé de réception, elle lance une procédure de trace qui consiste en un paquet spécial parcourant le chemin du paquet perdu et demandant à chaque nœud de rediffuser le paquet perdu (qu'ils ont gardé en mémoire un certain temps). Cette procédure a pour but de s'assurer que soit le nœud fautif se dénoncera tout seul (en ne rediffusant pas le paquet en direction de la source), soit il sera reconnu comme fautif par ses voisins. [21]

2.3.13 Un mécanisme basé sur la réputation pour isoler les nœuds égoïstes dans les réseaux ad hoc

Dans [31], les auteurs ont essayé de trouvé un mécanisme pour détecter et isoler les nœuds égoïstes en se basant sur la réputation des nœuds.
Leur mécanisme procure un schéma de réputation distribuée de façon autonome sur chaque nœud du réseau. Chaque nœud maintient une table de réputation où il stocke la valeur de la réputation de ces voisins à un saut, ce qui veut dire qu'il ne s'occupe que de la réputation de ces voisins directs. Chaque nœud tient pour responsable l'aboutissement ou non du paquet émis. Quand un paquet arrive à destination, chaque nœud récompense le nœud qui le succède à un saut en lui augmentant localement sa réputation. De même, le non-acheminement du paquet forcera les nœuds à diminuer la réputation de leurs successeurs à un saut.
Quand un nœud i reçoit un message de j à transmettre à k, il vérifie si ce message n'est pas une retransmission en vérifiant dans une table qui sauvegarde des informations sur les messages envoyés, si c'est le cas, le nœud que i avait chargé de transmettre ce message se verra retirer des points dans sa réputation. Ensuite le nœud i vérifie si la réputation du nœud j n'est pas inférieure à un certain seuil r_{thresh}, si c'est le cas alors le message sera ignoré, sinon il le transmettra à k comme convenu. Quand un acquittement est reçu d'un nœud k, le nœud i vérifie que c'est k qu'il avait chargé de transmettre le message, si c'est le cas alors il mettra à jour la réputation de k en l'augmentant.
Quand un nœud devient voisin d'un autre nœud, ils créent respectivement une case dans leurs tables de réputation avec comme valeur initial r_0. La réputation d'un nœud selon les auteurs

ne peut augmenter que jusqu'un seuil r_{max} prédéfini.
Ce modèle offre l'avantage de ne pas causer un trafic supplémentaire dans le réseau puisqu'il n'y pas d'échange d'information sur la réputation, ce qui fait qu'on a plus à se charger de la véracité des informations de réputation reçus par d'autres nœuds. De plus, il n'y a nul besoin de charger des nœuds spéciaux pour superviser le bon déroulement du protocole. Enfin ce modèle est totalement indépendant du protocole de routage, car il se contente de la réponse du destinataire, un acquittement *TCP* pourrait faire l'affaire.
L'un des inconvénients de cette approche est que les nœuds sont obligés de sauvegarder dans une table les informations des messages qui sont passés par eux pour pouvoir éventuellement savoir si un nœud à bien accomplis la tâche qui lui a été confiée. En effet le nombre de messages échangés et passant par des nœuds intermédiaires étant important, la taille de cette table le devient assez rapidement, ce qui exige que les nœuds doivent avoir une capacité de stockage importante. De plus, tenir pour responsable son voisin de l'échec d'acheminement d'un paquet même si ce dernier a bien transmis le paquet conduit à augmenter le nombre de fausses accusations des nœuds corrects.

2.3.14 Un système basé sur la réputation pour encourager la coopération des nœuds égoistes

Dans [32], les auteurs proposent une solution au problème de l'égoïsme en utilisant une approche par réputation appelée Reputation Management System (*RMS*). Ce modèle peut être utilisé comme extension à n'importe quel protocole de routage, s'exécutant indépendamment sur chaque nœud. Il permet de faire distribuer les valeurs de réputation calculées localement entre voisins.
Le *RMS* se compose des modules suivants :

- **Monitoring Module** : utilisant le mécanisme watchdog pour superviser les nœuds voisins s'ils envoient effectivement les paquets demandés. Ce module garde deux variables pour chacun de ses voisins, SBF(i, j) et ABF(i, j) qui sont respectivement le nombre de messages que le nœud i attend de j de transmettre et le nombre de message réellement transmis pas j selon i.
- **Reputation Manager** : responsable du maintient des valeurs de réputation de ses voisins dans la table de réputation. A Chaque période de temps T le *RM* met à jour la réputation de ces voisins grâce aux données fournies par le *MM* et aux réputations reçues par le Module de Communication . Pour calculer la réputation, les auteurs ont donné la formule suivante : RF(i,j)= β * RFO(j)+ (1-β)* RFR(j). Où RFO(j) est la valeur de la réputation observée par soit même, et RFR(j) est la valeur de réputation

reçue par les voisins, β est le degré de confiance en soit même.
Sachant que RFO(j)= α * RF_{table} (i,j)+ (1-α) (ABF(i,j))/(SBF(i,j)).
RF_{table} (i,j) Est la dernière valeur calculée par i pour j, et α est le degré d'importance qu'on veut donner au comportement passé pour ne pas pénaliser les nœuds qui ont eu des difficultés pendant une courte période de temps.
Sachant que :
RFR(j)= (($\sum_{v=1}^{n}$ (RR(i, k_v))+ RF(k_v, j)) / ($\sum_{v=1}^{n}$ (RR(i, k_v))).

À noté RR(i, k) est la valeur de réputation qu'a le nœud i pour son voisin k qui lui a envoyé sa valeur de réputation pour le nœud j RF(k, j). Cette fonction est définie ainsi pour éviter que des nœuds malicieux propagent de fausses réputations sur un nœud et falsifier ainsi les résultats de réputation. Il est aussi à noter que le système n'accepte pas les valeurs de réputation qui dépasse une certaine marge. En effet quand le nœud i reçoit RF(k, j) il compare cette valeur à RF(i, j) ;
Si |RF(k,j) - RF(i,j)| < O ; alors la valeur sera prise en compte et la réputation RF(i,k) sera incrémentée, sinon elle ne sera pas prise en compte et RF(i,k) sera décrémentée.
Selon la nouvelle valeur de réputation obtenue, l'état du nœud j sera déterminé soit *cooperative*, *suspected*, *inspecting*, *punished*, ou *blacklisted*. Le modèle introduit deux types de seuil, $TH_{selfish}$ et TH_{scoop}. Si la réputation est inférieure à $TH_{selfish}$ alors il sera à l'état *punished*,par contre si elle est supérieure à TH_{scoop} il sera à l'état *cooperative*. En revanche $TH_{selfish}$ < RF(j) < TH_{scoop} alors le nœud sera marqué *suspected*, et s'il reste dans cet état la pendant une période donnée, il sera marqué *punished*. L'état *punished* englobe quatre niveaux de punition, après une certaine durée de temps, si un nœud reste toujours dans cet état, il montera en niveau de punition jusqu'à arrivé au niveau 4, qui sera la dernière étape avant l'état *blacklisted* où le nœud n'aura plus de chance de faire marche arrière.

- **Communication Module** : ce module sert d'interface entre les *RMS* des nœuds voisins. La fonction principale de ce module est d'échanger les réputations entre voisins en utilisant des messages spéciaux.
- **Response Module** : ce module participe dans la phase de recherche et de sélection de route, en aidant à choisir la route avec la meilleur moyenne en réputation et n'ayant pas de nœud égoïste. Ce module traite aussi le cas où le noeud reçoit une requête de type *Route Request* d'un nœud égoïste, il va traiter cette requête selon l'état de ce nœud, s'il est *punished* avec un niveau 1, 2 ou 3 il sera traité respectivement avec 75%, 50% et 25% de réponse. S'il est dans l'état *punished* avec un niveau 4 ou dans l'état *blacklisted* alors aucun message ne lui sera délivré.

Ce modèle offre une possibilité aux nœuds de se racheter après s'être comporter de façon égoïste, en voyant qu'ils se font pénaliser, ils peuvent décider de corriger leurs comportements. Si on arrive à faire participer des nœuds égoïstes alors c'est tout le réseau qui est gagnant. Le modèle permet de ne pas pénaliser directement un nœud qui a eu des difficultés pendant une

certaine durée de temps en donnant plus d'importance au comportement qu'il a eu depuis le début. Aussi la diffamation de réputation est abolie dans ce modèle grâce au principe que prend le modèle en donnant plus d'importance aux observations faites par soit même, et en ne prenant pas en considération les réputations reçues ayant une différence supérieure à une certaine marge, en plus de pénaliser ce genre de diffamation en réduisant la réputation du noeud ayant envoyé cette information.
Ce modèle se base sur le mécanisme watchdog, qui possède plusieurs problèmes que nous avons énumérés précédemment. Certes les auteurs ont précisé que le mécanisme de réputation se fait par échange de messages, mais que ces échanges ne se font qu'au bout d'un certains temps, et seulement pour certains nœuds, néanmoins, sa reste toujours une charge en plus pour le réseau.

2.3.15 Protocole CATCH

Le protocole Catch, proposé par Mahajan et al. [33] a pour but de détecter les nœuds égoïstes dans un réseau multisauts sans fil. Ce protocole est destiné à être appliqué en parallèle du protocole de routage et fonctionne par l'échange de ses propres paquets de contrôle. Les nœuds du réseau mesurent le taux de retransmission de leurs paquets par leurs voisins par l'utilisation d'un watchdog. De plus, un sous-protocole d'échange de paquets anonymes (Anonymous Challenge Message (ACM) protocol), pour lesquels l'adresse MAC source de la trame ainsi que l'adresse IP source du paquet sont générées aléatoirement, est utilisées pour détecter les pertes intentionnelles. Comme les paquets sont anonymes, il est difficile pour celui qui les reçoit de déterminer leurs sources réelles dans l'hypothèse courante que les antennes utilisées sont omnidirectionnelles. L'idée du sous-protocole ACM est que les voisins testeurs d'un nœud testé envoient à ce dernier régulièrement des challenges anonymes qu'il est censé retransmettre. Si un défaut de retransmission par le testé est constaté par un testeur, alors ce dernier considère le lien comme déficient et ne retransmet plus les paquets en provenance du testé. Comme les challenges sont anonymes, le testé ne peut pas discerner leurs sources et a tout intérêt à les retransmettre tout au risque de voir un de ses liens critiques (qui lui permettent de rester connecté au reste du réseau) invalidés. Si le testé ne détruit pas intentionnellement des paquets, alors les résultats du sous-protocole ACM et ceux du watchdog devraient être concordants. Dans le cas où une incohérence entre les deux résultats est constatée par un testeur, alors le testé est considéré par le testeur comme égoïste et ce dernier invalide son lien avec le testé.
Pour pouvoir avertir les autres testeurs du diagnostic posé sur le testé, un testeur utilise un autre sous-protocole : Anonymous Neighbor Verification (ANV). Celui-ci se déroule en deux phases :

– premièrement (ANV Open), les testeurs envoient dans un paquet anonyme destiné au

testé (qu'il est censé retransmettre ensuite) le résultat de l'application d'une fonction de hachage à une valeur aléatoire ;
- deuxièmement (ANV Close), chaque testeur qui considère le testé comme non-égoïste envoie dans un paquet anonyme (à retransmettre également) la valeur initiale à laquelle avait été appliquée la fonction de hachage dans la première phase).

Les valeurs diffusées dans la seconde phase servent à chaque testeur à vérifier si tous les autres testeurs sont satisfaits du testé, par l'application de la fonction de hachage sur cette valeur et l'élimination des paquets reçus en première phase et contenant cette valeur hachée. S'il reste des paquets reçus en première phase et non éliminés en seconde phase, alors au moins un testeur considère le testé comme égoïste et par conséquent tous les testeurs s'accordent pour considérer le testé comme tel.
En plus de ces deux sous-protocoles, les testeurs envoient au testé un paquet nommé Tester Information Exchange (*TIE*) à retransmettre et qui contient l'identité du testeur ainsi qu'un bit de signe indiquant quelle part des paquets (données ou *ACM*) a eu le plus grand taux de retransmission. Les testeurs accumulent les signes des autres testeurs et effectuent le test du signe pour vérifier l'hypothèse selon laquelle les paquets de données et les paquets *ACM* ont la même probabilité d'être perdues. Enfin, chaque testeur effectue un test afin de décider si sa connectivité avec le testé telle que mesurée par le sous-protocole *ACM* correspond bien à celle mesurée par watchdog. La faiblesse de ce protocole quant à l'application aux réseaux ad hoc est qu'il est plutôt destiné aux réseaux à topologie statique. En effet, la décision prise par un testeur d'isoler un testé n'est prise qu'à la suite de plusieurs cycles d'ANV Open et ANV Close (appelés epochs) dont la durée de chacun est de l'ordre de la minute. Il faut donc que la topologie reste statique à cette échelle de temps, ce qui contredit l'hypothèse d'une topologie dynamique telle que comprise dans le cadre des réseaux ad hoc mobiles. [21]

2.3.16 SORI : A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks (Un système d'incitation sécurisé et Objectif basée sur la réputation)

Dans [38], les auteurs ont proposé une approche qui permet de détecter les nœuds égoïstes et de les punir, le schéma de base de cette approche est constitué de trois composants qui sont : la surveillance des voisins (neighbor monitoring), la propagation de réputation (reputation propagation) et la punition (punishment).

- **Surveillance des voisins (neighbor monitoring) :**
  Cette étape consiste en la collecte d'informations, chaque nœud N maintient une liste de

tous ses voisins notée NNL_n (Neighbor Node List) et surveille leurs transmissions (tel que watchdog), et pour chacun de ses voisins noté X, il calcul $RF_n(X)$ qui est le nombre total de message qu'il a transmis a X pour que ce dernier les retransmettent, et $HF_n(X)$ qui est le nombre total de message que X a retransmis. Ces deux variables sont mises a jour comme suit : quand le nœud N envoi un paquet a X (un de ses voisins), $RF_n(X)$ est incrémenté, puis N se met à l'écoute du canal et s'il détecte la retransmission du paquet par X, il incrémente $HF_n(X)$.
Une fois $RF_n(X)$ et $HF_n(X)$ calculés, le nœud N procède à l'évaluation du nœud X en lui créant un record $LER_n(X)$ (Local Evaluation Record) qui est constitué de deux parties :

- $G_n(X)$=$RF_n(X)/HF_n(X)$.
- $C_n(X) = RF_n(X)$ le degré de confiance du nœud N dans ses jugements à propos de la réputation de X.

- **Propagation de réputation (réputation propagation) :**
  En cette seconde étape, les nœuds voisins s'échangent la réputation des autres nœuds, ainsi un nœud égoïste sera puni par tous ses voisins. La propagation de réputation s'énonce comme suit :
  Premièrement, chaque nœud N met à jour périodiquement ses $LER_n(X)$ pour tous ses voisins X et ensuite si jamais la valeur de $G_n(X)$ marque un changement significatif, la nouvelle valeur $LER_n(X)$ est diffusée pour l'ensemble des voisins.
  Deuxièmement, le nœud N utilise $LER_n(X)$ et $LER_i(X)$ (i appartient au voisinage de N -> i appartient à NNL_n) pour calculer le $OER_n(X)$ (overall evaluation record) selon la formule suivante :

  $ORN_n(X)=(\sum_{i \in NNL_n \cup N, i \neq X} \lambda_N(i).C_i(X).G_i(X) \; / \; \sum_{k \in NNL_n \cup N, k \neq X} \lambda_N(k).C_k(X))$.

  Tel que λ (i) = G_N(i) est la crédibilité que le nœud i a eu du nœud N ($\lambda_N(N)=1$).

- **Punition (punishment)** :
  Une fois la réputation calculée, le nœud N procède à la punition de ses voisins comme suit : si $OER_n(X)$ est inferieur a un seuil prédéfini, les paquets de X seront supprimés selon la probabilité :
  p= { q-δ si q>δ ; 0 sinon }.

  Tel que q=1- $OER_n(X)$ et $0<\delta<1$ est la marge de tolérance, car des fois il se peut que le comportement égoïste d'un nœud est dû a une collision et non a l'égoïsme du nœud lui-même.

Cependant les auteurs ont soulevé certains risques que comporte leur approche, en effet un nœud égoïste peut usurper l'identité d'un nœud voisin qui a une bonne réputation afin de transmettre ses propres paquets ou bien diffuser de fausses informations d'observation, dans le but d'accroitre sa réputation calculée par les autres nœuds. Mais les auteurs ont pu y remédier en renforçant leur approche par un mécanisme de sécurité appelé *one-way-hash chain* qui est un mécanisme d'authentification complètement distribué.

Cette approche dénote pas mal d'avantages concernant la détection des nœuds *selfish* car la réputation d'un nœud n'est transmise qu'aux voisins ce qui permet de réduire la charge de communication du réseau, de plus même si un nœud est reconnu égoïste, il existe toujours une marge de tolérance puisque ses paquet ne sont supprimés que suivant une probabilité, ce qui permet de réduire le nombre de fautes commises par l'accusation d'un nœud d'égoïsme a tord.

2.3.17 OCEAN : Observation-based Cooperation Enforcement in Ad hoc Networks (Observation basée sur l'exécution de coopération dans les réseaux ad hoc)

Dans [39], les auteurs considèrent deux types de comportement malsains pour le routage dans les réseaux mobiles ad hoc.

- Le premier, appelé " *tromper* " ou " *misleading* ", c'est le fait qu'un nœud réponde positivement à une requête de route sans réellement transmettre le paquet, trompant ainsi les autres nœuds en envoyant leur trafic à travers lui.
- Le deuxième appelé " *égoïste* ", c'est le fait qu'un nœud ne répond même pas aux requêtes de route, alors qu'il continue à utiliser le réseau en envoyant ses propres paquets.

Pour remédier au premier comportement malsain, OCEAN propose cinq composants :

1. **NeighborWatch :**
   Repose un peu sur le principe du WatchDog, chaque nœud surveille ses voisins directes, quand un nœud envoi un paquet, le NeighborWatch garde un *checksum* du paquet envoyé et il se met en écoute du canal, s'il ne détecte pas la retransmission du paquet par son voisin avant l'expiration d'un *timeOut*, il enregistre contre ce dernier un événement négatif et il supprime le *checksum* du paquet de son buffer, sinon (dans le cas ou il détecte une retransmission d'un paquet par son voisin) il compare le paquet avec le

checksum qu'il a gardé dans son buffer, et si le *checksum* correspond au paquet alors il enregistre un événement positif et il supprime le *checksum* de son buffer sinon il traite le paquet comme un paquet non retransmis, et ces événements seront communiqués au *RouteRanker* qui maintient une notation pour les nœuds voisins.

2. **RouteRanker :**
   Chaque noeud maintient une notation pour chacun de ses voisins, la notation est initialisée a *Neutral* (0) comme montré dans le tableau ci-dessous, et elle est incrémentée ou décrémentée selon les événements positifs ou négatifs reçu du NeighborWatch. Si jamais la notation d'un nœud tombe en dessous d'un certain seuil noté *Faulty_ Threshold*, le nœud est ajouté à la *faulty_ list*. Cette liste contient tous les nœuds ayant un comportement malsain, ainsi une route est qualifiée de bonne si elle ne contient pas de nœuds appartenant à la *faulty_ list*, sinon elle sera qualifiée de mauvaise.

Paramétres	**Valeurs**
Neutral Rating	0
Positive Step	+1
Negative Step	-2
Faulty_Threshold	-40

TABLE 2.1 – Les paramètres par défault d'OCEAN

3. **Rank-Based Routing :**
   Ce module applique les informations du NeighborWatch dans le choix des routes, pour éviter de construire des routes qui contiennent des nœuds dans la *faulty_ list*, un champ est ajouté a la DSR *Route-Request* (RREQ) nommé *avoid-list*, qui est la liste des nœuds que l'émetteur de *RREQ* souhaite éviter dans ses futures chemins, en rediffusant une *RREQ*, un nœud ajoute sa *faulty_ list* à la *avoid_ list* de la *RREQ*, chaque nœud recevant une *RREQ* vérifie sa *avoid_ list* et selon cette liste et la *RREQ_ route*, un nœud décide s'il supprime la *RREQ* ou l'honore en la rediffusant ou en envoyant la DSR *Route-Reply* (RREP). Si l'intersection de la *avoid-list* et la DSR route dans la *RREQ* est non vide alors la *RREQ* est supprimée, de la même manière un nœud envoi une *RREP* seulement si le chemin de la *RREP* ne contient aucun noeud appartenant a sa *faulty_ list*.
   De ce fait chaque nœud prend sa propre décision sur les nœuds en qui il aura confiance et il n'a de contrôle que sur les routes dans lesquels il participe.

4. **Malicious Traffic Rejection :**
   Ce module s'occupe de la stratégie du rejet du traffic des nœuds misleading, en effet tous les nœuds misleading voient leurs trafics rejetés jusqu'à ce qu'ils transmettent des paquets pour d'autres nœuds (qu'ils redeviennent légitimes).

5. **Second Chance Mechanism :**
   L'objectif du mécanisme de seconde chance est de permettre a un nœud considéré comme *misleading* de redevenir utile à nouveau, puisqu'une fois qu'un nœud est ajouté a la *faulty_ list*, toutes les routes établies dans le future l'éviterons, ainsi il ne pourra jamais transmettre des paquets pour d'autres nœuds afin de regagner leurs confiances. Une approche basée sur un *timeOut* est utilisée pour remédier à ce problème, un nœud est retiré de la *faulty_ list* après une période déterminée d'inactivité observée, sa notation n'est pas pour autant remise a Neutral, ce qui fait qu'il peut rapidement être ajouté de nouveau a la faulty_list dans le cas ou il continue à mal se conduire. La variable du *timeOut* est nommée *Faulty_ Timeout*.

Pour ce qui est du deuxième comportement malsain, à savoir l'égoïsme, l'idée générale est de punir les nœuds égoïstes en rejetant leurs trafics.
OCEAN au lieu de s'appuyer sur les observations directes des interactions avec les voisins afin de mesurer leurs performances, chaque nœud maintient un compteur nommé *chipcounts* pour chacun de ses voisins. Un nœud gagne des *chips* en transmettant un paquet pour un autre nœud, et il en perd s'il demande à un autre nœud de lui transmettre un paquet.
Il existe deux schémas basés sur le commerce qui ont été proposés : un schéma optimiste et un autre pessimiste.
Dans l'approche optimiste un nœud A incrémente le *chipcounts* d'un nœud B chaque fois que ce dernier accepte un paquet de A indépendamment du fait que B retransmette le paquet ou pas, alors que dans l'approche pessimiste le *chipcounts* de B n'est incrémenté que si A observe la retransmission de son paquet par B.

2.3.18 COMMIT : A Sender-Centric Truthful and Energy-Efficient Routing Protocol for Ad Hoc Networks with Selfish Nodes (Un protocole de routage énergiquement efficace pour les réseaux ad hoc avec les nœuds égoïstes)

Dans [17], les auteurs ont carrément proposé un protocole de routage qui a pour but d'offrir un meilleur routage avec un minimum d'énergie et d'obliger tous les nœuds du réseau à coopérer entre eux. Pour ce qui est de notre cas nous allons nous intéresser à la manière dont le protocole traite les nœuds égoïstes.

Les auteurs ont eu recours à la théorie des jeux pour modéliser leur protocole, un nœud S voulant établir une connexion possède une information privée qui est son consentement de

payer pour l'établissement de la connexion. En supposant que m est le prix maximum par paquet que S est prêt à payer, le gain du joueur S si la connexion est établie est modélisé comme suit : u_s = m-c_s(D) tel que c_s(D) représente le montant réel que S payera par paquet. Dans le cas ou la connexion ne peut pas être établie, u_s=0. Pour ce qui est d'un nœud intermédiaire V, son information privée est représentée par son niveau de puissance *l(v)*. Le cout de la transmission c_v d'un paquet envoyé par S est déterminé par *l(v)* et d'autres facteurs comme l'énergie restante du nœud, bande passante du nœud, ...etc. pour des raisons de simplicité les auteurs ont supposé c_v=*l(v)*. Le gain du nœud V s'il prend part a la transmission est u_v = *pay(v)*-l(v) tel que *pay(v)* est le montant que V a reçu en retransmettant un paquet pour S.

Le protocole assure trois tâches essentielles :

- **Détermination du gagnant (Winner determination)** : détermine le chemin gagnant (s'il existe), en d'autre terme c'est le chemin qui coute le moins d'énergie.
- **Calcul de payement (payment computation)** : dans le cas ou le chemin gagnant existe, on détermine le montant que S devra payer pour la transmission/réception des paquets, ainsi que le payement pour les nœuds du chemin gagnant.
- **Facturation (billing)** : si la communication est établie, une facture est alors envoyée au nœud S et un payement des nœuds du chemin gagnant selon le montant déterminé précédemment est effectué.

Le *winner determination* et le *payment computation* sont effectués par le nœud destination D, le *billing* se fait quand la transmission de données commence.

Dans ce protocole, les auteurs ont supposé plusieurs hypothèses plus ou moins fortes comme l'absence des nœuds malicieux dans le réseau ou le fait que les nœuds ne peuvent pas comploter ensemble pour maximiser leurs gains. De plus les auteurs n'ont pas spécifié la manière dont le payement doit s'effectuer sachant que c'est un problème majeur dans les approches basées sur le crédit.

2.4 Conclusion

Dans ce chapitre, nous avons passé en revue les différentes solutions proposées pour palier au problème de l'égoïsme des nœuds dans les réseaux mobiles ad hoc et cela dans le but de renforcer les performances du réseau et gérer au mieux les ressources énergétiques. Pour ce

faire, nous avons présenté deux types d'approches : les approches basées sur la réputation, et les approches à base de crédit. Chacune des deux approches souffrent de quelques handicaps que Miranda et Rodrigues on essayé de résoudre en proposant une solution hybride combinant les deux catégories existantes.
Dans le chapitre suivant, nous allons introduire notre contribution qui consiste en une solution basée sur la réputation.

Chapitre 3

Approche de détection et d'isoation des nœuds égoïstes dans les MANETs

3.1 Introduction

Soit un réseau Ad hoc ayant parmi ses nœuds, des nœuds aux caractères égoïstes. De tels nœuds essayent de tirer usage des ressources du réseau en profitant de la coopération d'autres nœuds, sans pour autant faire de même pour eux. La non-coopération d'un nœud signifie que les paquets qui passent par lui seront perdus. Pour le bien du réseau, on doit trouver une solution pour palier à ce problème, ce qui permettrait de ne plus avoir de paquets abandonnés par ces nœuds égoïstes en les obligeant à coopérer dans le réseau au lieu de directement les exclure et cela permet d'augmenter considérablement les performances du réseau.
Pour la solution que nous allons proposer, nous avons opté pour la notion de réputation des nœuds, pour les forcer à participer avec les autres nœuds pour avoir une réputation et ainsi être servis par les autres. La proposition que nous allons présenter s'inspire des modèles à acquittement multi-saut tel que *TWOACK* [34], mettant en évidence leurs points fort et essayant de combler leurs manques et faiblesses.
Dans le présent chapitre nous allons commencer par poser la problématique puis nous passerons à la partie des hypothèses prise en considération pour un meilleur fonctionnement de notre système, ensuite nous passerons à la présentation de notre approche, et enfin nous donnerons une conclusion.

3.2 Problématique

Définition de l'égoïsme :
le problème de l'égoïsme consiste dans le fait qu'il y'ai des nœuds parmis les nœuds du réseau qui essayent de maximiser leur gains au détriment du réseau ; c'est à dire qu'un nœud égoïste va tenter tans bien que mal de récolter le maximum des gains pour son bien être et servir que son propore flux et cela en ne prenant guerre compte de l'intêret des autres nœuds du réseau.

Bien que le problème d'égoïsme soit une forme d'attaque passive, il est quand même l'origine de néfastes répercussions sur les performances du réseau. De nombreuses études et simulations [35, 36, 37] ont été faites pour évaluer l'impacte de la présence des nœuds égoïstes dans un réseau ad hoc, et les résultats obtenus ont définitivement montré les dégâts engendrés par ce genre de nœuds. Il est donc primordial de trouver une solution qui puisse régler ce problème ou du moins le réduire. Une telle solution doit pouvoir répondre à certains critères essentiels pour pouvoir être une solution complète et optimale.

Nous pouvons désormais dire qu'une bonne solution doit pouvoir détecter les nœuds aux comportements égoïstes. Effectivement, étant une attaque passive, sa détection devient beaucoup plus difficile et complexe.

On ne doit en aucun cas, laisser un nœud égoïste se comporter comme bon lui semble. Pour cela, des mesures de précaution doivent être prises, pour punir toute action malhonnête et pénaliser les nœuds responsables. Vu que les nœuds égoïstes ne transmettent pas les paquets qui leur ont été confiés pour retransmettre, il est donc primordial de pouvoir les éviter dans la phase de routage pour minimiser la perte des paquets et d'énergie.

Néanmoins, il peut arriver qu'un nœud désiste et désire arrêter de se comporter de façon égoïste, et peut avoir envie de se comporter correctement et coopérer avec l'ensemble des nœuds du réseau, un tel nœud doit avoir le droit de réintégrer le réseau, il est donc souhaitable que la solution offre le droit de se racheter pour les nœuds désirant se corriger, ce qui serait d'une autre part rentable pour les performances du réseau.

Les solutions qui existent peuvent être divisées en deux grandes catégories : les solutions basées sur la réputation [21, 34, 24] et les solutions basées sur le crédit [19]. Il est aussi possible de trouver des solutions hybrides qui combinent entre la réputation et le crédit [29].

Une solution hybride à l'avantage de pouvoir combiner les points forts des deux approches. Il est nécessaire de profiter des avantages de l'approche entreprise quel que soit sa catégorie.

Dans le cas où, une solution à base de réputation est choisie, on aura le choix entre faire le calcule de la réputation localement au niveau de chaque nœud, ou bien de faire une distribution des réputations gardées dans les nœuds au sein de tout le réseau. Une solution distribuée a l'avantage d'être plus précise, car on se base sur les connaissances de plusieurs nœuds, mais d'autres problèmes apparaissent par exemple le risque de diffamation de réputation, et offre l'inconvénient d'ajouter une charge supplémentaire au réseau en diffusant les messages de réputation. Une solution de calcul de réputation locale, évite la propagation de fausse réputation, et n'ajoute pas de charge au réseau, mais les informations collectées, seules ne suffît pas toujours à avoir une bonne réputation.

Une solution peut solliciter la supervision de son processus ou pas. Dans le cas où la solution nécessite une supervision, il est nécessaire de prévoir l'élection des nœuds superviseur, il faut que ce choix soit le plus précis possible. En effet, il faut choisir un nœud digne de confiance, et dont la durée de vie sera assez suffisante pour mener à bien son rôle.

3.3 Hypothéses

Dans cette partie, nous allons présenter ce que nous avons pris comme hypothèses dans l'environnement d'exécution de notre mécanisme. Nous allons voir que notre système et très flexible, pouvant s'exécuter dans divers environnements. En effet, ces hypothèses nous permettent d'avoir une exécution plus optimale du système, et non pas des conditions nécessaires pour son fonctionnement. De plus, nous n'avons pas pris beaucoup d'hypothèse, rendant ainsi notre système, plus libre.

- D'abords, nous avons supposé que les liens entre les nœuds du réseau sont bidirectionnels, ce qui veut dire que pour chaque couple de nœud (A, B), si A peut communiquer avec B, alors B est en mesure de faire de même.
- Nous avons également supposé qu'un service de certification à clés publiques est mis en place, permettant ainsi de chiffrer les messages circulant dans le réseau, et notamment les messages qui sont propres à notre système, permettant ainsi de garantir l'intégrité des données.

Nous remarquons que les deux hypothèses citées ci-dessus, sont des paramètres faciles à implémenter. En effet, les nœuds dans les réseaux mobiles ad hoc ont une portée de communications assez grande, et presque équivalente chez tous les nœuds. Aussi, il est facile d'implémenter une infrastructure de gestion de clés publiques dans les réseaux ad hoc, et généralement, la plupart des MANET le font.

3.4 Principe de la propostion

Dans ce chapitre, nous proposons un modèle basé sur la réputation, pour régler le problème d'égoïsme dans les MANETs. Cette solution permet de forcer les nœuds du réseau à coopérer pour pouvoir profiter des services des autres, et d'isoler les nœuds dont le comportement n'est pas honnête. Le principe est assez simple, et s'appuie sur les modèles à acquittement multisaut (multi-hop acknowledgement), pour essayer de localiser avec précision la position du nœud égoïste dans le chemin emprunté par un message n'ayant pas atteint sa destination.

En se basant sur plusieurs autres travaux [34, 22], qui ont justifié leur choix sur le nombre de sauts d'un acquittement et prouver leur efficacité, nous avons choisi de faire un acquittement à deux sauts, car cela rendra l'inculpation d'un nœud plus précis, que dans le cas ou le nombre de saut d'un acquittement est de trois sauts ou plus.

Ce principe peut être décrit comme suit :
Soit par exemple, un nœud N_s voulant envoyer un message à N_d. N_s construit le chemin vers N_d grâce à n'importe quel protocole de routage, tout en évitant les nœuds égoïstes. En envoyant un paquet, tous les noeuds attendent l'acquittement de la destination pendant un temps *Ack_ Delay*, lorsque cet acquittement arrive, chaque nœud augmente la réputation des nœuds successeurs sur le chemin du message en appliquant la formule (8) et (9). Si après, le *timeOut Ack_ Delay* aucun acquittement n'est reçu, chaque noeud N_i ayant participé au transfert du message enverra un message *2HopAck*, au noeud N_{i-2} qui est à deux sauts en arrière dans le chemin du message pour innocenter le nœud N_{i-1} prédécesseur à un saut. Le nœud N_{i-2} en recevant ce message augmentera la réputation des nœuds N_{i-1} et N_i en leurs appliquant respectivement la formule (1) et (2). Quand un nœud N_i voit après un délai *Exculpation_ Delay*, que le nœud N_{i+1} n'envoi pas de paquet *2HopAck* au nœud N_{i-1}, il le signal à tous les autres nœuds en leur envoyant un paquet *SelfExculpation* et le tient pour

responsable de l'échec de la transmission du message, et diminue sa réputation en appliquant la formule (4). Les nœuds recevant ce message, commencerons par augmenter la réputation des nœuds ayant transmis le message jusqu'au nœud N_i en appliquant la formule (5), et ignorant la réelle source du problème, ils vont pénaliser les deux nœuds N_i et N_{i+1} en diminuant leurs réputations en leur appliquant respectivement les formules (6) et (7), et en soustrayant plus au nœud qui leurs semble le plus probablement égoïste en comparent leurs réputation. De plus, chaque nœud recevant un paquet en provenance d'un autre, va vérifier la chaine de nœuds qu'a parcouru le message avant de lui parvenir, et va mettre à jour leurs réputations en l'augmentant en appliquant la formule (3). Dans le cas où un nœud N_i ne reçoit pas après un délai le paquet *2HopAck* d'un nœud N_{i+2} pour le nœud N_{i+1} et que ce dernier ne proclame pas son innocence, alors il sera inculpé par le nœud N_i qui le signalera au nœud source. Tous les nœuds du chemin recevant ce message réduiront la réputation du nœud N_{i+1} en appliquant la formule (4), et augmenteront la réputation des autres nœuds intermédiaires grâce à la formule (5).

Chaque nœud garde à son niveau une table *TrustTable* qui stocke les valeurs de réputation qu'il a des autres nœuds. À chaque fois qu'un nœud obtient une information sur un autre, il met à jour la valeur de sa réputation s'il a déjà une entrée dans la table pour ce nœud, si aucune entrée dans la table ne correspond à ce nœud, alors il va créer une nouvelle entrée et sauvegarder cette valeur à l'intérieure. Au début, chaque nœud donne une réputation initiale $\alpha_{\ reputation}$ aux autres nœuds.

De plus, chaque nœud maintient une table de données dite *PostTable*, qui stocke l'identifiant des messages qu'il transmet, ainsi que le chemin qu'il empreinte. Quand la destination envoi un acquittement à la source pour lui confirmer sa réception, tous les nœuds qui reçoivent cet acquittement, vérifie dans leurs *PostTable* s'il y a un paquet qui correspond au paquet acquitté, et supprime la ligne correspondante dans ce cas. Également, la réception d'un paquet *SelfExculpation* ou d'un paquet *SelfishDetection* aura le même effet. Si après un timeout aucun acquittement n'est reçu pour un certain message, la ligne correspondante à ce message sera supprimée après qu'il ait envoyé le message *2HopAck* au nœud à deux sauts en arrière, pour innocenter le nœud à un saut en arrière comme expliqué précédemment.

Dans notre approche, nous avons choisis de faire une distribution des valeurs de réputation entre les nœuds du réseau. En effet, pour détecter plus rapidement les nœuds égoïstes, il est préférable que les nœuds du réseau collabore ensemble en s'échangeant leurs connaissances sur le comportement des autres nœuds. Mais pour ne pas trop encombrer le réseau de messages en plus, les nœuds ne transmettent les valeurs sur la réputation d'un nœud que si celle-ci change d'un certain seuil $\triangle_{\ reputation}$. La valeur de ce seuil doit être bien étudiée, une valeur

assez petite rendra la collaboration plus forte et donc plus efficace, mais augmentera la charge du réseau, et sera une perte d'énergie pour les nœuds. De même, prendre une valeur assez grande réduira la charge du réseau, mais sa réduirai aussi la collaboration des nœuds, rendant ainsi la détection des nœuds égoïstes plus lente.
Lors de la réception d'une valeur de réputation d'un nœud, on calcule une nouvelle valeur de réputation pour ce nœud en prenant en compte cette valeur et la valeur qu'on avait déjà. Pour éviter que des valeurs diffamatoires faussent la réputation des nœuds, on peut choisir de prendre en considération les valeurs reçues qu'avec un faible facteur d'impacte. Nous avons choisi d'appliquer la formule suivante pour le calcul d'une nouvelle valeur de réputation prenant en compte les valeurs reçues :

$$Trust_i(j)=((\varphi * Trust_i(j)+ \sum_{i=1}^{n} receivedReputation_i))/(\varphi+ n) \quad .....(10)$$

Où φ est le facteur qu'on donne à la réputation qu'on a déjà calculée auparavant. *receivedReputation*, est la valeur de réputation reçue d'un nœud quelconque.

Quand la réputation d'un nœud passe en dessous d'un certain seuil *Threshold*, le nœud l'ayant détecté diffusera dans l'ensemble du réseau un paquet d'accusations*SelfishAlert*, contenant son identité et l'identité du nœud accusé. Chaque nœud recevant ce message le sauvegardera. Si à un certain moment on reçoit un certain nombre d'accusations pour un nœud donné, égale à *WitnessRate*, alors ce nœud sera considéré comme égoïste par l'ensemble des nœuds du réseau. Cette technique permet d'accélérer le processus de détection des nœuds égoïstes dans le réseau.

Le paramètre *WitnessRate* est un paramètre très important. En effet, un taux élevé réduirait le taux des fausses accusations, engendrées par la diffamation d'information par des nœuds malicieux, en contrepartie, la détection des nœuds égoïstes deviendra plus lente. Un taux moins grand, assure que la détection des nœuds égoïstes soit plus rapide, mais malheureusement, le taux de faux positif pourrait croitre considérablement si des nœuds malicieux se mettent à donner de fausses accusations.

Les formules utilisées par le système sont les suivantes :

N_i récompense N_{i+1} pour avoir envoyer le message à N_{i+2} comme suit :

$Trust_i\ (N_{i+1}) = Trust_i\ (N_{i+1}) + RSD$ ..... (1).

Et récompense aussi N_{i+2} pour l'avoir confirmé comme suit :
$Trust_i\ (N_{i+2}) = Trust_i\ (N_{i+2}) + RST$ ..... (2).

Quand un nœud N_i reçoit un message d'un nœud N_{i-1}, il appliquera la fonction suivante :

pour *tout nœud* N_j *allant de* N_1 *à* N_{i-1} **Faire**
$Trust_i\ (N_j) = Trust_i(N_j) + RSD$..... (3)
Finpour

Un nœud pénalise son successeur de ne pas avoir envoyé de paquet *2HopAck*, le tenant ainsi pour responsable du non-acheminement du message, en appliquant la formule suivante :
$Trust_i\ (N_{i+1}) = Trust_i\ (N_{i+1}) - PNSD$.....(4).

Lors de la réception d'un message *SeflExculpation* d'un nœud k, tout les nœuds N_i se trouvant sur le chemin exécuterons :

pour*tout nœud* N_j *allant de* N_{i+2} *à* N_{k-1} **Faire**
$Trust_i\ (N_j) = Trust_i(N_j) + RSD$..... (5)
Finpour

$Trust_i\ (N_k) = Trust_i\ (N_k) - (1 - Trust_i\ (N_k))^* PNSD - MPS(N_k, N_{k+1})^* APNSD + MPS(N_{k+1}, N_k)^* RSM$ .....(6).

$Trust_i\ (N_{k+1}) = Trust_i\ (N_{k+1}) - (1 - Trust_i\ (N_{k+1}))^* PNSD - MPS(N_{k+1}, N_k)^* APNSD$ .....(7).

Enfin lors de la réception d'un acquittement de la destination ceci sera appliqué pour chaque nœud N_i du chemin :

pour*tout nœud N_j allant de N_{i+1} à N_{d-1}* **Faire**
$Trust_i$ (N_j)= $Trust_i$(N_j)+ RSD..... (8)
Finpour

$Trust_i$ (N_d)= $Trust_i$ (N_d)+ RST .....(9).

Si aucune disculpation n'est envoyée ni par le nœud N_k lui-même ni par le nœud qui le suit, un paquet d'inculpation *Selfish_ Detection* sera émis, et la réputation du nœud N_k sera réduite par tous les nœuds du chemin recevant ce message en appliquant la formule (4). Dans ce cas également les nœuds intermédiaires seront récompensés en appliquant la formule (5).

Notation :
Les paramètres que nous avons utilisés pour établir notre solution sont les suivants :

- *RSD* : valeur positive à ajouter à la réputation d'un nœud ayant transmis un message.
- *PNSD* : valeur positive à soustraire de la réputation d'un nœud pour le punir de ne pas avoir transmis de message
- *APNSD* : Valeur positive en plus à soustraire à la réputation d'un nœud qui nous semble être le nœud égoïste.
- *RSM* : valeur positive à ajouter à la réputation d'un nœud pour le récompenser d'avoir signalé le mauvais comportement d'un nœud.
- *RST* : Valeur positive à ajouter à la réputation d'un nœud comme prime pour avoir envoyé un paquet *2HopAck*.
- *LastBroadcastedTrust*$_i$(j) : la dernière valeur de réputation qu'a diffusé le nœud i à propos de j.
- *Ack_ Delay(i)* : le délai qu'un nœud N_i doit attendre pour l'acquittement du message qu'il a collaboré à transmettre. Après ce délai, il déclenche le *Exculpation_ Delay*.
- *Exculpation_ Delay* : le temps que le nœud N_i doit attendre un *2HopAck* venant de N_{i+1} en direction de N_{i-1}. Dépassé ce délai, N_i enverra un paquet *SelfExculpation*.
- *Others_ Exculpation_ Delay* : le délai qu'un nœud N_i doit attendre un paquet *2HopAck* du nœud N_{i+2} disculpant le nœud N_{i+1}.
- $\alpha_{\ reputation}$:la réputation initiale de chaque nœuds.
- $\triangle_{\ reputation}$: le seuil de changement qu'on doit attendre avant de diffuser la nouvelle réputation d'un nœud. $\triangle_{\ reputation}$= Trust$_i$ (j)- *LastBroadcastedTrust*$_i$ (j).
- *Threshold* : le seuil à partir du quel un nœud sera considéré comme égoïste s'il passe en dessous.
- *WitnessRate* : c'est le taux nécessaire de nœud accusant un nœud d'être égoïste, pour le

considérer comme tel dans tout le réseau.

Nous avons aussi utilisé les fonctions suivantes :

- *MPS(i, j)* : une fonction qui retourne 1 si la réputation de j est supérieure à celle de i, et retourne 0 dans le cas contraire.
- $Trust_i(j)$: est une fonction qui retourne la réputation de j aux yeux de i, qui est sauvegardée dans la *TrustTable*.

Pour ce qui est des messages échangés dans le système voici les paramètres utilisés :

- *2HopAck* : message envoyé par un nœud vers le nœud qui le précède de deux sauts sur le chemin qu'empreinte le message.
- *SelfExculpation* :message envoyé par le dernier nœud ayant essayé de transmettre le message vers la source pour signaler le refus d'un nœud à transmettre le message.
- *Selfish_Detection* : paquet envoyé à un nœud n'ayant pas reçu la disculpation de son successeur, l'accusant ainsi d'être égoïste. Il faut préciser, que si par exemple ce message est envoyé par un nœud N_i, alors le nœud N_{i-1} n'acceptera de le transmettre que si N_{i+1} a disculpé N_i en envoyant un paquet*2HopAck* au nœud N_{i-1}.
- *SelfishAlert* : paquet envoyé dans le but de signaler la détection d'un nœud égoïste.

Sachant qu'on a mis l'hypothèse qu'un service de certification est mis en place dans le réseau, les messages *2HopAck* et *SelfExculpation* seront chiffrés pour garantir leur intégrité et leur authenticité.

- **Algorithmes**
  Pour mieux formaliser notre approche, nous avons essayé d'expliquer le fonctionnement de notre protocole sous forme d'algorithme. Nous présenterons l'algorithme du coté émetteur, et un autre pour les autres nœuds, et enfin l'algorithme à appliquer lors de toute mise à jour de la réputation d'un nœud.

Algorithm 1 Algorithme émetteur

lors de la transmission d'un message
data = Donnée à transmettre ;
Rechercher des chemins vers la destination ;
choisir le meilleur chemin ;
envoyer (data) ;
while *(non acquittement reçu ET Ack_ Delay!= 0)* **do**
  attente () ;
end while
if *acquittement reçu* **then**
  for *tout nœud* N_j *allant de* N_{i+1} *à* N_{d-1} **do**
    $Trust_i\ (N_j) = Trust_i(N_j) + RSD$;
  end for
  $Trust_i(N_d) = Trust_i(N_d) + RST$;
  else
  while *(Others_ Exculpation_ Delay!= 0 ET non 2HopAck reçu)* **do**
    attente () ;
  end while
  if *2HopAck reçu* **then**
    $Trust_i\ (N_1) = Trust_i(N_1) + RSD$;
    $Trust_i\ (N_2) = Trust_i(N_2) + RST$;
    else
    $Trust_i(N_1) = Trust_i(N_1) - PNSD$;
    if *SelfExculpation reçu* **then**
      for *tout nœud* N_j *allant de* N_2 *à* N_{k-1} **do**
        $Trust_i(N_j) = Trust_i(N_j) + RSD$;
      end for
      $Trust_i(N_k) = Trust_i(N_k) - (1 - Trust_i(N_k)) * PNSD - MPS(N_k, N_{k+1}) * APNSD + MPS(N_{k+1}, N_k) * RSM$;
      $Trust_i(N_{k+1}) = Trust_i(N_{k+1}) - (1 - Trust_i(N_{k+1})) * PNSD - MPS(N_{k+1}, N_k) * APNSD$;
    end if
    if *Selfish_ detection reçu* **then**
      for *tout nœud* N_j *allant de* N_2 *à* N_k **do**
        $Trust_i(N_j) = Trust_i(N_j) + RSD$;
      end for
      $Trust_i(N_{k+1}) = Trust_i(N_{k+1}) - PNSD$;
    end if
  end if
end if
Fin.

Table 2 Algorithme récepteur

lors de la reception d'un message au niveau du nœud i
for *tout nœud* N_j *allant de* N_1 *à* N_{i-1} **do**
$Trust_i\ (N_j) = Trust_i(N_j) + RSD;$
end for
if *destination_message* == adresse local **then**
envoyer acquittement
else
sauvegarder l'ID du message et son chemin dans la PostTable;
while *(acquittement non reçu ET Ack_Delay != 0)* **do**
attente ();
end while
if *acquittement reçu* **then**
for *tout nœud* N_j *allant de* N_{i+1} *à* N_{d-1} **do**
$Trust_i\ (N_j) = Trust_i(N_j) + RSD;$
end for
$Trust_i(N_d) = Trust_i(N_d) + RST;$
supprimer la ligne correspondante à l'ID du message acquitté dans la table;
else
emettre un paquet 2HopAck au nœud precedent à deux sauts;
while *(2HopAck non reçu ET Exculpation_Delay* != 0) **do**
attente ();
end while
if *2HopAck reçu* **then**
$Trust_i(N_{i+2}) = Trust_i(N_{i+2}) + RST;$
else
$Trust_i(N_{i+1}) = Trust_i(N_{i+1})$ - $PNSD;$
envoyer un paquet SelfExculpation à la source;
end if
while *(2HopAck non reçu ET Others_Exculpation_Delay != 0)* **do**
attente ();
end while

Algorithm 2 Algorithme récepteur (suite)

if *2HopAck reçu* **then**
  $Trust_i(N_{i+1}) = Trust_i(N_{i+1}) + RSD$;
  $Trust_i(N_{i+2}) = Trust_i(N_{i+2}) + RST$;
  else
  $Trust_i(N_{i+1}) = Trust_i(N_{i+1})$ - $PNSD$;
  envoyer un paquet Selfish_Detection contenant l'ID du nœud N_{i+1} *à la source*;
end if
if *SelfExculpation reçu* **then**
  for *tout nœud* N_j *allant de* N_{i+2} *à* N_{k-1} **do**
    $Trust_i(N_j) = Trust_i(N_j) + RSD$;
  end for
  $Trust_i(N_k) = Trust_i(N_k)$ - $(1 - Trust_i(N_k))$ * $PNSD$ - $MPS(N_k, N_{k+1})$* $APNSD$+ $MPS(N_{k+1}, N_k)$*RSM;
  $Trust_i(N_{k+1}) = Trust_i(N_{k+1})$ - $(1 - Trust_i(N_{k+1}))$ * $PNSD$ - $MPS(N_{k+1}, N_k)$* $APNSD$;
end if
if *Selfish_detection reçu* **then**
  for *tout nœud* N_j *allant de* N_{i+2} *à* N_k **do**
    $Trust_i(N_j) = Trust_i(N_j) + RSD$;
  end for
  $Trust_i(N_{k+1}) = Trust_i(N_{k+1})$ - $PNSD$;
end if
end if
Fin.

Algorithm 3 Algorithme de diffusion de la réputation

lors de la mise à jour de la réputation d'un nœud
if $(|Trust_i(j) - LastBroadcastedTrust_i(j)| > \triangle_{reputation})$ **then**
  diffuser $Trust_i(j)$;
  $LastBroadcastedTrust_i(j) = Trust_i(j)$;
end if

Pour mieux comprendre le fonctionnement de notre système, nous allons présenter un exemple concret de réseau contenant des nœuds égoïstes parmi ses nœuds.
Soit un réseau mobile Ad Hoc constitué de treize nœuds <A, B, ..., M>, dont trois sont égoïstes <B, F, M>, qui seront représentés en orange dans les schémas suivants. La topologie

du réseau est la suivante :

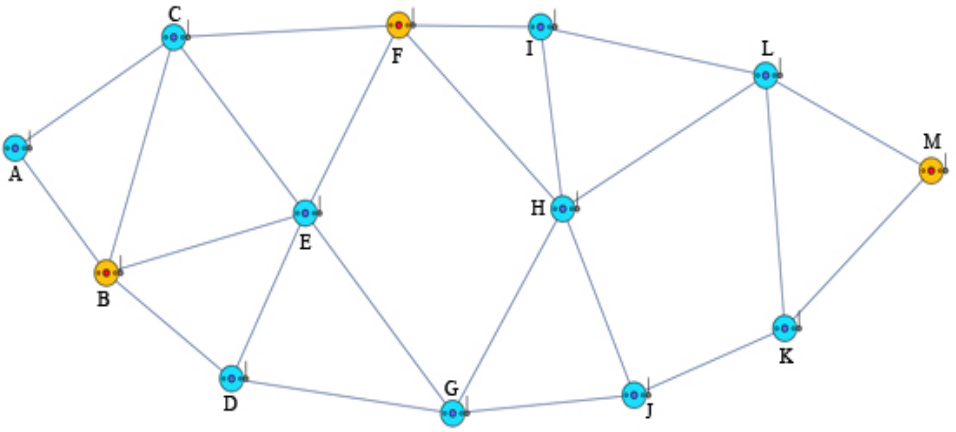

FIGURE 3.1 – Exemple pratique de topologie d'un réseau mobile Ad Hoc

Supposant qu'à l'instant t, un nœud M veuille envoyer un message à un nœud A. Le nœud M commence par rechercher les routes qui mènent au nœud A. plusieurs routes existent, mais comme on l'a précisé, il va choisir le plus court chemin ne contenant pas de nœud égoïste. Supposant qu'à l'instant t, la vision du nœud M sur la réputation de l'ensemble des nœuds du réseau est comme présentée dans la figure (4.2) :

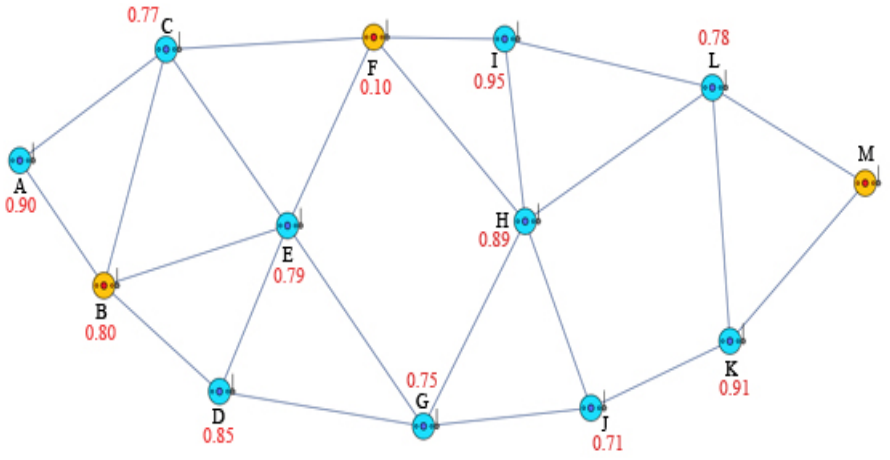

FIGURE 3.2 – Vision globale du noeud M sur la réputation des nœuds

Nous supposons que même les nœuds égoïstes tels que M, surveillent le comportement des autres nœuds, car c'est aussi dans leur intérêt d'éviter d'autres nœuds égoïstes, sauf s'il y a une conspiration entre eux. Donc, que ce soit pour un nœud ordinaire ou un nœud égoïste, l'exécution de notre système ce fait de la même manière.

En prenant en compte les critères de sélection du meilleur chemin, le chemin qui va logiquement être choisi par le nœud M est celui illustré dans le graphe suivant :

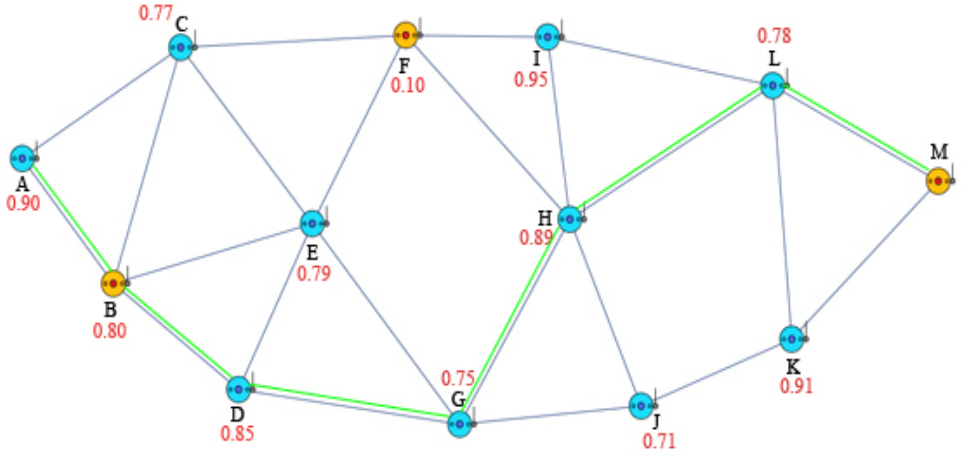

FIGURE 3.3 – illustration du chemin choisi par M pour atteindre A

Nous remarquerons qu'il a bien évité le chemin "*M, L, I, F, C, A*" contenant un nœud égoïste, même si ce chemin est le plus court. Le nœud M peut alors commencer l'envoi de son paquet.

Supposant qu'à cet instant, le nœud M, a encore une bonne réputation au sein du réseau,

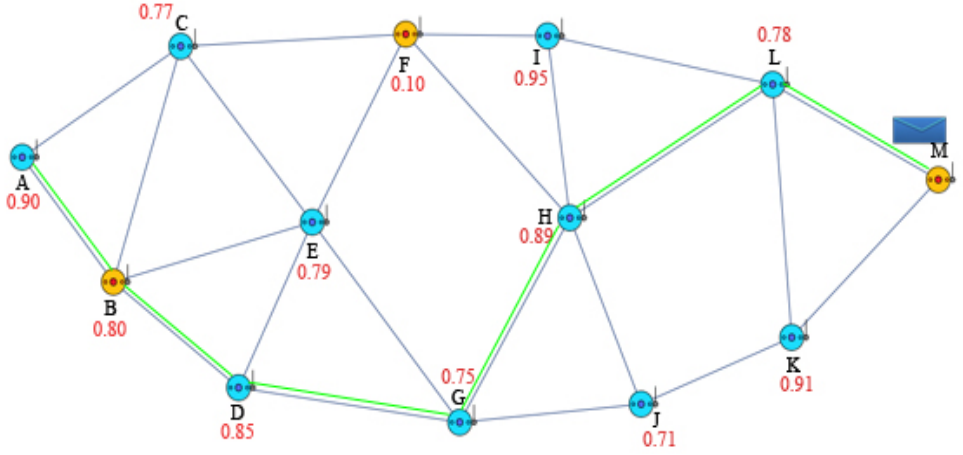

FIGURE 3.4 – début du transert de paquets

il sera donc servi normalement par les autres nœuds n'ayant eu pas encore le temps de bien

analyser son comportement. Si par contre sa réputation est descendue au dessous d'un certain seuil *Threshold*, alors aucun de ces paquets ne lui sera transmis.

Le message parcourt le chemin comme convenu jusqu'à arriver au nœud B qui est un nœud égoïste pas encore détecté. Durant le parcourt du chemin, chaque nœud l'ayant reçu fera en sorte d'augmenter la réputation des nœuds qui ont collaboré pour que le message arrive jusqu'à lui en appliquant la formule (3).

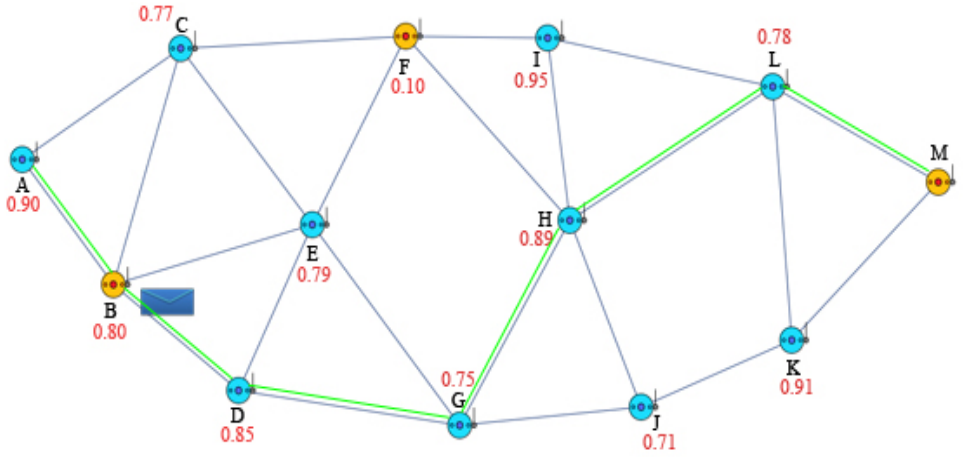

FIGURE 3.5 – arrivée d'un paquet chez un nœud égoïste

Ne voulant pas gaspiller son énergie à servir les autres, le nœud B choisi d'ignorer le message de M, et donc A ne recevra pas ce message.

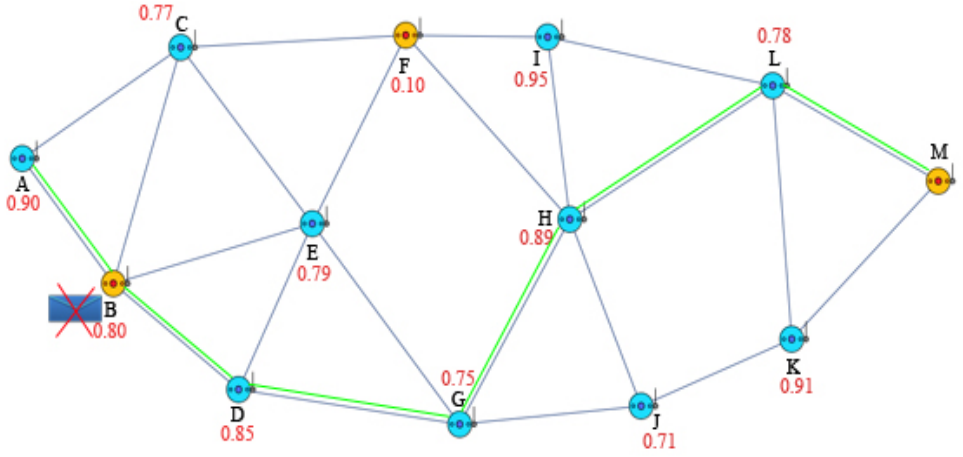

FIGURE 3.6 – suppression de messages par un nœud égoïste

Les nœuds ayant participé dans le transfert du paquet attendront pendant un *TimeOut* l'arrivée de l'acquittement du nœud A confirmant la réception du paquet. Comme le nœud

B n'a pas transmit le message à A, aucun acquittement ne sera délivrai par ce dernier. Les nœuds vont alors collaborer pour cerner le nœud égoïste.

La collaboration des nœuds réside dans le fait que chaque nœud envoie un paquet *2HopAck* au nœuds qui le précède de deux sauts pour lui confirmer que le nœud intermédiaire (prédécesseur à un saut) lui a bien transmis le message. Sa ressemblerai dans la vie courante à un témoignage pour disculper le nœud intermédiaire. Un nœud ne peut pas mentir sur le fait qu'un autre nœud lui ai envoyé le message, car le message qu'il doit envoyer doit contenir l'identifiant du message et aussi connaitre le chemin du message. Un nœud ne peut pas non plus envoyer un paquet *2HopAck* et prétendre qu'il provienne d'un autre nœud, vu qu'on a posé l'hypothèse qu'il existe un service de certification. Les messages *2HopAck* sont alors chiffrés avec la clé de l'émetteur pour garantir l'authenticité et l'intégrité du message.
Sachant tout ça, les paquets *2HopAck* générés sont les suivants : un provenant de H en destination de M, un autre de G pour L, et enfin un de D en direction de H.

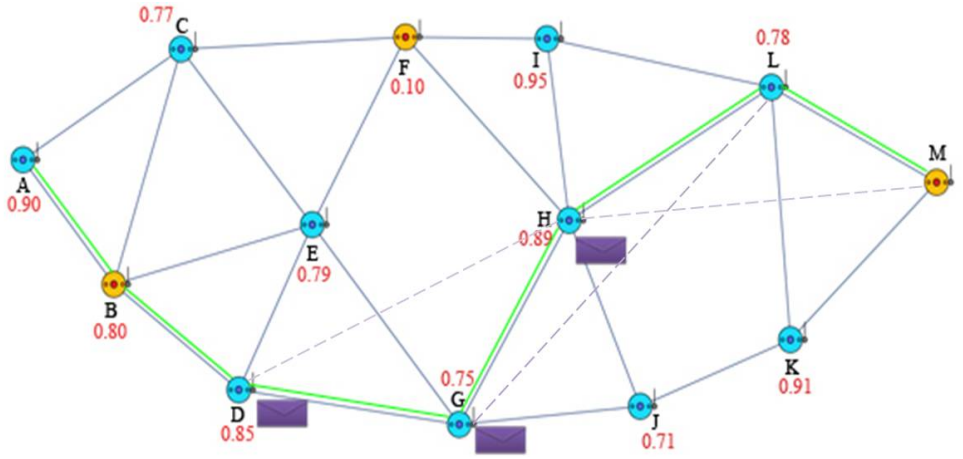

FIGURE 3.7 – génération des paquets *2HopAck*

Les nœuds M, L et H ayant reçus le paquet *2HopAck*, augmenteront la réputation de leurs nœuds successeur à un saut en appliquant la formule (1) décrite précédemment, mais aussi le nœud qui a émis le paquet *2HopAck* pour le remercier de sa collaboration en appliquant la formule (2).
Après l'écoulement d'un certain *TimeOut*, le nœud D ayant transmis le message au nœud B, remarque que ce dernier n'envoie pas de paquet *2HopAck* au nœud G, il sera donc forcé de générer un paquet *SelfExculpation*, pour signaler son innocence aux autres nœuds, et dénoncer ainsi le comportement de B. il va éventuellement diminuer la réputation de B en appliquant la formule (4).

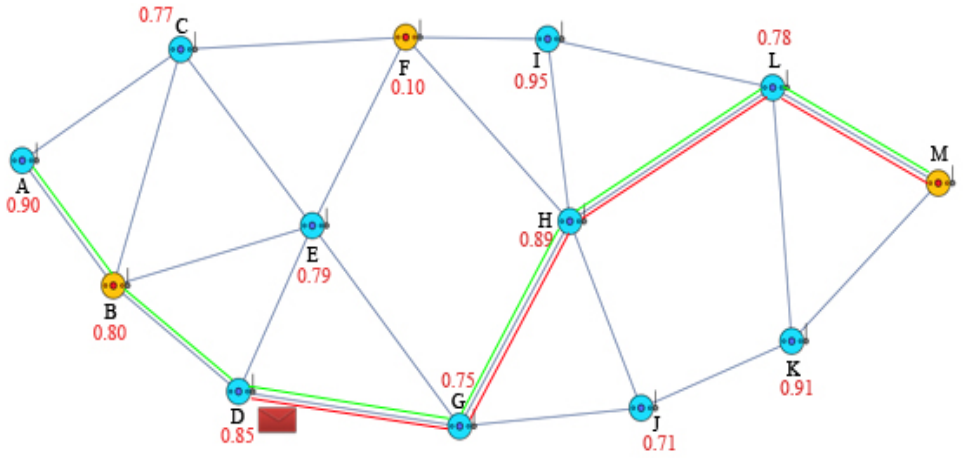

FIGURE 3.8 – génération du paquet *SelfExculpation*

Le paquet *SelfExculpation* emprunte le même chemin que celui du message. Les nœuds recevant ce paquet, étant sûrs que le message ait atteint le nœud B, augmenteront la réputation des nœuds qui ont fait l'intermédiaire entre eux en appliquant la formule (5). Malheureusement, on ne peut pas dire si le paquet a été envoyé par un nœud légitime ou bien par un nœud égoïste avec des intentions malicieuses. En effet, le nœud B aurait très bien pu envoyer un paquet *2HopAck* à G pour disculper D, ensuite envoyer un paquet *SelfExculpation* pour accuser le nœud A d'être égoïste. Pour éviter d'être injuste envers l'un ou l'autre, nous avons fait en sorte de leurs prélever de leurs réputations à tous les deux une valeur qui est plus grande quand la réputation est plus petite. De plus, une valeur supplémentaire sera réduite du nœud dont la réputation est inférieur à l'autre, c'est comme dire le nœud avec la réputation la moins élevée est le plus probable d'être la cause de l'échec. Les formules à appliquer dans ce cas seront la formule (8) et la formule (9) respectivement pour le nœud D et B. Dans le cas présent, on aura le nœud B qui sera plus pénalisé que le nœud D.

Supposant que le nœud B décide d'envoyer un paquet *2HopAck* au nœud G, puis envoi un paquet *SelfExculpation*, sa sera un avantage au nœud D, mais sa ne tirera pas le nœud B de l'affaire car lui comme A seront toujours suspectés par les autres nœuds.

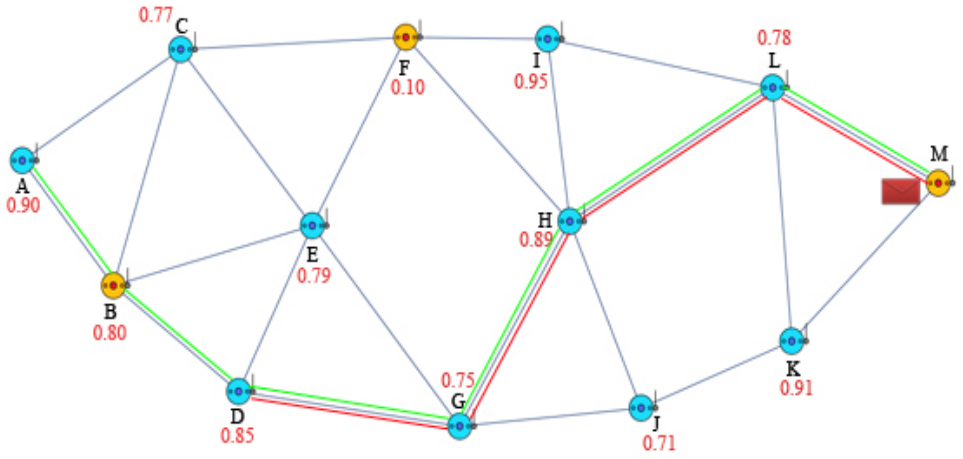

FIGURE 3.9 – arrivé du paquet *2HopAck* à la source

Supposant maintenant que le nœud B ait envoyé un paquet *2HopAck* au nœud B, D sera donc innocenté. Mais dans ce cas, D ne recevra pas la disculpation de B, car il n'a pas transmis le message. Après que le nœud D ait attendu un temps égal à *Others_Exculpation_Delay* , il envoie un paquet *Selfish_Detection* pour informer les autres nœuds que ce nœud est probablement le nœud égoïste, car aucune disculpation n'a été reçue, et il n'a pas proclamé son innocence. Un nœud ne peut envoyer ce genre de paquet que s'il a été lui-même innocenté. Après ce message, tous les nœuds réduiront la réputation du nœud accusé, ici c'est B, en appliquant la formule (4), et les nœuds intermédiaires seront récompensés avec la formule (5).

3.5 Conclusion

Dans ce chapitre, nous avons proposé un modèle basé sur la réputation des nœuds pour essayer de remédier au problème de l'égoïsme. Ce modèle permet de détecter les nœuds malveillants et de les éviter dans le protocole de routage des paquets. Aussi, cette solution minimise la charge du réseau en réduisant le trafic. En effet, elle évite de générer des messages *2hopack* lorsque l'acheminement des paquets se déroule correctement, ces messages ne seront transmis que dans le cas ou il existe des nœuds égoïstes dans le réseau et la transmission de paquet ne s'accomplit pas.

Chapitre 4

Simulation et Analyse des performances

4.1 Introduction

Après avoir présenté notre approche, nous allons dans cette section, procéder à son évaluation. Pour ce faire nous avons développé un simulateur en utilisant le langage java simulant le comportement d'un réseau mobile ad hoc, et on a implémenté notre approche dessus.

Dans notre simulateur les nœuds mobiles sont considérés comme n'importe quel équipement pouvant communiquer et envoyer des signaux (ils possèdent une interface réseau).

Pour ce qui des paramètres de simulation utilisés, nous avons simulé un réseau ad hoc contenant 50 nœuds mobiles. Les nœuds égoïstes sont choisis aléatoirement parmi ces 50 nœuds avec un pourcentage allant de 0 à 40 % avec un pas de 5 %. Ces nœuds se déplacent de manière aléatoire selon le modèle *Random WayPoint*, i.e. un nœud choisit aléatoirement une destination, et une vitesse de déplacement, une fois arrivé à destination il marque un temps de repos aléatoire puis recommence a nouveau la même procédure. La vitesse maximale d'un nœud est de 15 m/s, et le temps de repos maximal est de 3 s. Les nœuds du réseau ont tous la même portée qui est égale à 150 mètres. La surface de déploiement des nœuds est de 1000 m * 1000 m. l'envoi de message suit une loi de poisson de paramètre λ chaque 10 ms. Enfin, le temps de simulation est de 30 secondes.

Pour ce qui est des valeurs attribuées aux paramètres [1] de notre approche, elles sont présentées comme suit : La table de réputation *TrustTable* de chaque nœud est initialisée avec une valeur $\alpha_{reputation} = 0.75$. Le seuil pour lequel un nœud est considéré comme égoïste est *Threshold* =0.25. On récompense un nœud pour avoir envoyé un message avec RSD = 0.1, et on le pénalise avec PNSD = 0.15. Nous avons choisi de récompenser la collaboration des nœuds qui envoient des acquittements avec RST = 0.05. La valeur additionnelle qu'on soustrait au nœud qui nous parait le plus probablement égoïste est APNSD = 0.05. La récompense pour avoir signalé un mauvais comportement est de RSM = 0.03. La diffusion d'une valeur de réputation se fait quand elle change considérablement de $\Delta_{reputation} = 0.2$. Lors de la diffusion d'une nouvelle valeur de réputation, le nœud recevant cette valeur ne la prendra en considération que de 25% par rapport à la réputation qu'il a déjà.
Le taux nécessaire de nœuds accusant un nœud d'être égoïste pour le déclarer dans tout le réseau comme tel est *witnessRate* = 5%.
Dans notre simulation, nous avons supposé que le réseau est fiable, i.e. un paquet émis est un paquet reçu, il n'ya pas de perte du aux collisions ou a un autre problème sauf l'égoïsme.
Le tableau ci-dessous résume les différents paramètres utilises.

Paramétres	**Valeurs**	**Pas**
Nombre de nœud	50	-
Taux de nœuds égoïstes	0-40 %	5%
Vitesse maximale	15m/s	-
Repos maximal	3s	-
Rayon de porté	150m	-
Largeur de la zone	1000m	-
Longueur de la zone	1000m	-
Temps de simulation	30s	-
$\alpha_{reputation}$	0.75	-
Threshold	0.25	-
RSD	0.1	-
PNSD	0.15	-
APNSD	0.05	-
RST	0.05	-
RSM	0.03	-
$\Delta_{reputation}$	0.2	-
witnessRate	5%	-

TABLE 4.1 – Les paramètres de simulations utilisées

1. Ces paramètres n'ont pas fait l'objet d'une étude approfondie, pour le moment c'est juste une estimation. On en reparlera dans les perspectives

4.1.1 Evaluation des performances

Nous allons commencer par l'évaluation du taux de délivrance de messages (messages arrivés à destination) en variant le taux des nœuds égoïstes dans le réseau. Puis nous passerons aux taux de faux positifs (nœuds accusés à tord d'égoïsme) et de faux négatifs (nœuds égoïstes qui ne sont pas détectés) toujours en variant le taux des nœuds égoïstes dans le réseau.

- **Impact du taux des nœuds égoïstes sur la délivrance des messages** :
  Comme le montre la *figure 5.1* Notre approche donne d'excellents résultats, en effet la borne inférieure du taux de réussite de délivrance de messages dépasse les 91%. Au départ, en l'absence des nœuds égoïstes dans le réseau, le taux de délivrances de message atteint les 100%, puis en augmentant le taux des nœuds égoïstes dans le réseau, ce taux diminue jusqu'à atteindre 91.89% correspondant à 40% du taux des nœuds égoïstes dans le réseau.

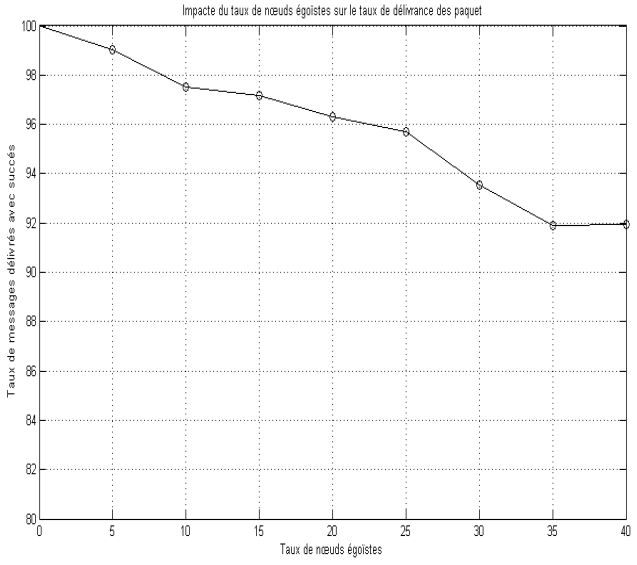

FIGURE 4.1 – Taux de paquets délivrés avec succès en fonction du taux des nœuds égoïstes

Pour ce qui est du taux des paquets perdus, on remarque qu'il n'excède pas les 9%, bien que le taux des nœuds égoïstes présent dans le réseau va jusqu'à 40%, cela est du au fait qu'un nœud détecté comme égoïste est évité dans l'algorithme de routage.

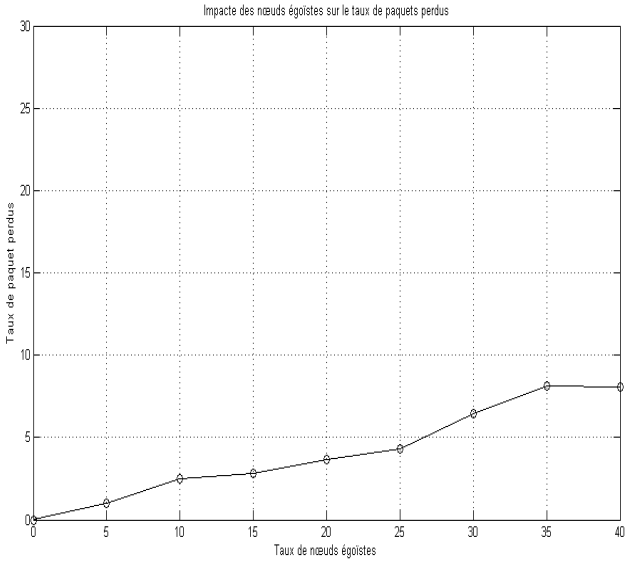

FIGURE 4.2 – Taux de paquets perdus en fonction du taux de nœuds égoïstes

Nombre de *faux négatifs* vs taux des *nœuds égoïstes* :
Dans notre approche, on a proposé un mécanisme de détection des nœuds égoïstes qui se base sur la réputation, en effet chaque nœud ayant sa réputation au dessous d'un certain seuil est déclaré comme égoïste, on remarque sur la *figure 5.3* que jusqu'à 25% de nœuds égoïstes dans le réseau, le taux de détection de ces derniers est de 100%, i.e. tous les nœuds égoïstes dans le réseau ont été détecté et à partir de 25% ce taux diminue jusqu'à atteindre 89%, cela est du a la durée de simulation, en effet plus la durée est longue, plus on a de chances de détecter les nœuds égoïstes.

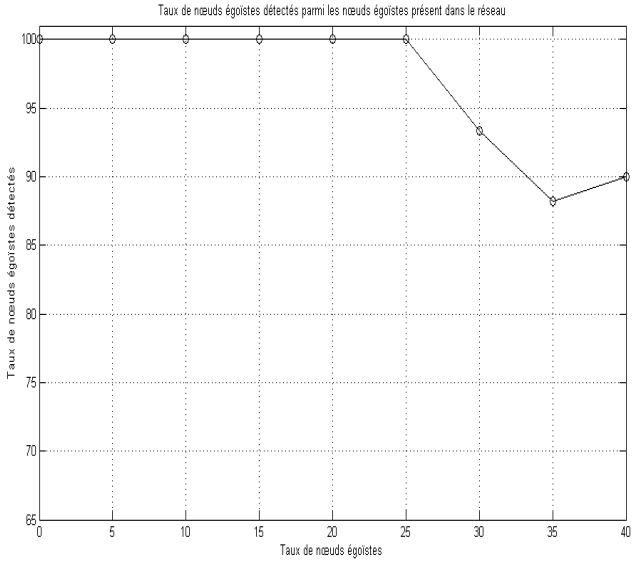

FIGURE 4.3 – Taux de faux négatifs

Nombre de *faux positifs* vs taux des nœuds égoïstes :
La *figure 5.4* montre le taux des nœuds accusés a tord en fonction du taux des nœuds égoïstes dans le réseau, on remarque que la courbe est régulière et fixée a la valeur de 0% i.e. aucun nœud n'est accusé a tord, ce qui est une preuve de l'efficacité de notre approche, cela est du au mécanisme ingénieux qu'on a utilisé pour déclarer un nœud comme égoïste, en effet on a fixé un taux de 5% des nœuds du réseau qui doivent accuser un nœud, pour que ce dernier soit déclaré comme égoïste, ce qui donne une certaine robustesse a l'approche vis-à-vis les nœuds malicieux.

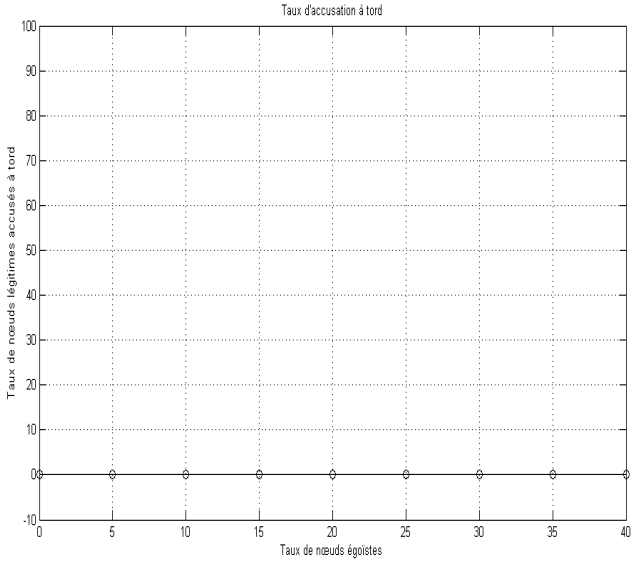

FIGURE 4.4 – Taux de faux positifs

Taux de délivrance de messages au fil du temps :
Au début, nous avons un taux de délivrance de message de 65% du fait qu'on ai pas encore détecté les nœuds égoïstes, et au fur et au mesure que l'on découvre ces nœuds égoïstes et qu'on les évite, on aurons le taux de délivrance de message qui augmente.

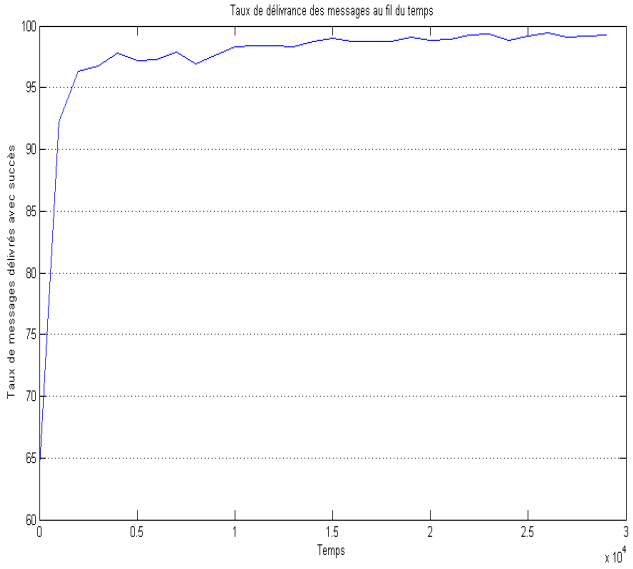

FIGURE 4.5 – Taux de délivrance de messages avec succès

Alors que le taux de perte de paquets suit le chemin inverse, en effet ce taux atteint 35% quand les nœuds égoïstes ne sont pas encore détectés, et donc ils ne peuvent pas être évités, puis ce taux commence à diminuer en découvrant les nœuds égoïstes au fil du temps.

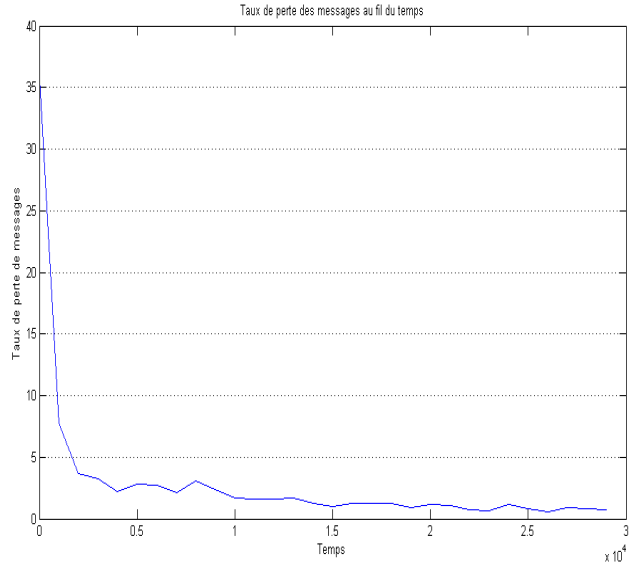

FIGURE 4.6 – Taux de perte de messages

La charge des acquittements de contrôle sur le réseau au fil du temps :
On remarque que la charge induite par les acquittements atteint presque les 22% du taux total des messages échangés sur le réseau au début de la simulation, et cela est dû au fait que

les nœuds égoïstes commencent à être détectés, puis au fil du temps, cette charge diminue considérablement au fur et à mesure de détection des nœuds égoïstes.

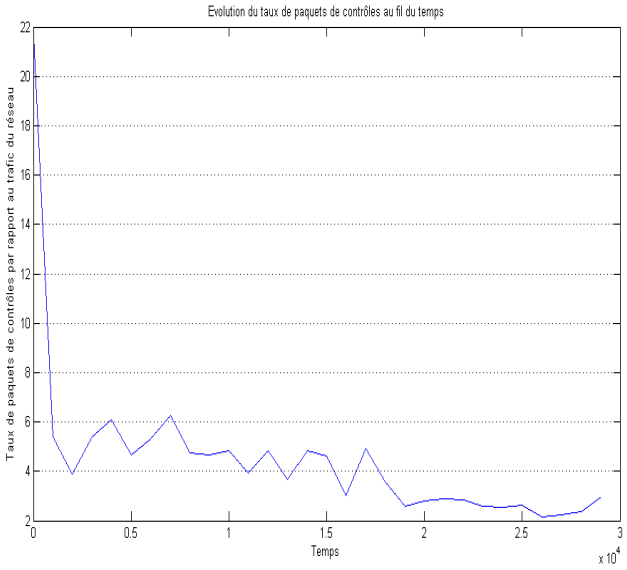

FIGURE 4.7 – La charge des paquets de contrôles au fil du temps

4.2 Conclusion

Les résultats de simulation ont été largement satisfaisants, et ont prouvé l'efficacité et la robustesse de notre approche. Comme on l'a vu dans cette section, notre approche dépasse largement les 90% en ce qui concerne le taux de détection des nœuds égoïstes, ce qui permet d'éviter ces derniers et offrir ainsi un taux de délivrance de messages assez élevé (plus de 91%), de plus le taux des nœuds accusé à tord est de 0% ce qui constitue un avantage majeur par rapport aux autres approches étudiées. L'évaluation de notre approche par rapport au temps met clairement en évidence ses avantages. En effet plus le temps passe, plus le taux de délivrance de messages réussi augmente, et celui de perte de messages diminue.

Conclusion et Perspectives

Les réseaux mobiles ad hoc ont apporté beaucoup de nouveautés à l'informatique d'aujourd'hui, et promettent d'en apporter encore plus pour celle de demain à condition de trouver des solutions aux problèmes qu'ils ont soulevé, on entend par la les problèmes de sécurité, d'énergie, de routage, et le comportement byzantin des nœuds, nous mettons le point sur le problème traité dans notre travail, qui est le problème d'égoïsme. En effet, il se peut qu'il y'ai des nœud égoïstes qui ne respectent pas leurs engagements et rôles dans le réseau, entrainant ainsi une dégradation des performances du réseau.

Plusieurs mécanismes ont été proposés dans la littérature afin de remédier au problème d'égoïsme. Ces mécanismes peuvent être classé en deux catégories : les modèles à base de *réputation* qui nécessitent des mécanismes de supervision et en général de la collaboration des nœuds, et ceux à base de *crédit* qui nécessitent bien souvent l'aide du matériel. Nous avons opté pour une solution à base de réputation pour ne pas exiger du matériel aux nœuds et ainsi être plus ouverts à l'hétérogénéité du réseau. Notre approche peut être implémentée avec n'importe quel protocole de routage. L'approche que nous avons proposée repose sur la notion de réputation et la collaboration des nœuds. Nous avons prouvé à travers les simulations son efficacité. Selon notre lecture, notre approche est la seule qui combine le modèle réputation et l'acquittement multisauts. L'idée de base de notre approche est qu'elle n'entre en jeu que dans le cas où la délivrance d'un paquet n'a pas abouti, ce qui est un avantage pour les nœuds qui essayent de préserver autant d'énergie possible.

En guise de perspectives, notre travail pourrait être enrichi dans un premier temps par une étude sur les paramètres utilisés dans notre approche, notamment :

- le RSD, le RST, le RSM, le PNSD, le APNSD : il est indispensable de faire le meilleur choix pour augmenter et diminuer la réputation d'un nœud pour le récompenser et/ou

le punir.
- *Threshold* : une grande valeur peut servir à détecter rapidement un nœud égoïste, mais peut par contre engendrer les fausses accusations.
- $\Delta_{reputation}$: une grande valeur réduit la charge du trafic supplémentaire, mais réduit la collaboration des nœuds à détecter les comportements égoïstes.

- *WitnessRate* : plus le taux est grand, plus on a moins de chance d'accuser à tord un nœud, mais en contrepartie, l'exclusion d'un nœud égoïste du réseau prendra plus de temps.

Nous prévoyons aussi de faire une simulation de notre approche dans des conditions réelles d'un réseau, où la perte de paquets est prise en considération, et où des nœuds malveillants collaboratifs sont présents afin de mesurer les faiblesses de notre solution et de concrétiser des résultats pour d'éventuelles comparaisons et améliorations.

Bibliographie

[1] *"Goals and challenges of the darpa glomo program [global mobile information systems"*, R. J. Ruther B. M Leiner and A. R. Sastry, IEEE Personal Communications, 3(6) : pages 34-43, 1996.

[2] *"The darpa packet radio network protocols"*, Jubin and J. D. Tornow, Proceedings of the IEEE, 75(1) : page 21-32, 1987.

[3] *"The low-cost packet radio."*, W.C Fifer and F. J. Bruno, Proceedings of the IEEE, 75(1) : page 33-42. 1987.

[4] *"Routing Protocol Performance Issues and Evaluation Considerations"*, S. Corson and J. Macker. Mobile Ad hoc Networking (MANET) : IETF RFC 2501. 1999.

[5] *"Rapport de stage sur Les protocoles de routage dans les réseaux ad hoc"*, HAGGAR Bachar Salim, Université de Reims. Le 21 juin 2007.

[6] *"Optimized Link State Routing Protocol (OLSR)"*, T. Clausen and P. Jaquet, RFC 3626 (Experimental). 2003.

[7] *"Fisheye State Routing Protocol (FSR) for Ad Hoc Networks"*, X. Hong M. Gerla and G. Pei.. IETF Internet Draft : draft-ietf-manet-fsr-03.txt. 2002.

[8] *" BTopology dissemination Based on Reverse- Path Forwarding (TBRPF)"*, G. Lewis Mark L. Fred Templin and G. Richard Ogier. 2004.

[9] *"Ad-Hoc On demand Distance Vector routing (AODV)"*, S. Das E. Belding-Royer and C. Perkins, IETF RFC 3561(Experimental). 2003.

[10] *"The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4"*, Y. Hu D. Johnson and D. Maltz, IETF RFC 4728. 2007.

[11] *" The Zone Routing Protocol (ZRP) for Ad Hoc Networks"*, R. Marc Pearlman and Zygmunt J. Haas, IETF Internet Draft. 2002.

[12] *"Cluster Based Routing Protocol (CBRP)"*, J. Li M. Jiang and Y.C. Tay, IETF Internet Draft. 1999.

[13] *"Consommation d'énergie dans les réseaux sans fil avec différentiation de services : implémentation et simulation"*, Mouna ABDELMOUMEN, rapport de fin d'étude. 2005/2006.

[14] *"La sécurité des réseaux sans fil ad hoc"*, F. Dupont S. Gombault V. Gayraud, L. Nuaymi and B. Tharon, IETF Computer Society. Rennes France,2003.

[15] *"Ad Hoc network specific attacks"*, A. Burg, Seminar Ad Hoc networking, concepts, Applications and security. Technishe Universitat Munshen 2003.

[16] *"Systèmes de Détection d'Intrusions dans les Réseaux Ad Hoc"*, Aymen MELLASSINE, Rapport de projet de fin d'études. 2004/2005.

[18] *"Mitigating routing misbehavior in mobile ad hoc networks"*, S. Marti, T. Giuli, K. Lai, and M. Baker, in Proceedings of The Sixth International Conference on Mobile Computing and Networking.Boston 2000.

[19] *"Sprite : A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks"*, Shen Zhong, Jiang Chen, Yang Richard Yang..

[20] *" Algorithmes distribués pour la sécurité et la qualité de service dans les réseaux Ad hoc mobiles"*, Ignacy GAWEDZKI. THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS-SUD 11..29 septembre 2008.

[21] *"Détection of Routing Misbehavior in MANETs"*, International Conference on Computer and Software Modeling IPCSIT vol.14 IACSIT Press. Singapore 2011.

[22] *"An Acknowledgment-based Approach for the Detection of Routing Misbehavior in MANETs"*, Kejun Liu, Jing Deng, Pramod K. Varshney, and Kashyap Balakrishnan.

[23] *"Using a cache scheme to detect selfish nodes in mobile Ad hoc networks"*, Hongxun Liu, José G. Delgado-Frias, and Sirisha Medidi, international conference communications, Internet and information technology. 2-4 July 2007, CANADA.

[24] *"Performance analysis of the CONFIDANT protocol"*, Sonja Buchegger and Jean-Yves Le Boudec, Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, pages 226-236, ACM Press. New York, NY, USA, 2002.

[25] *"A robust reputation system for mobile ad-hoc"*. Sonja Buchegger and Jean-Yves Le Boudec. Technical report, EPFL, November 11 2003.

[26] *" Statistical Decision Theory and Bayesian Analysis"*. James O. Berger, 2nd Edition. Springer.1985.

[27] *"Statistical Models"*, Anthony C. Davison. Cambridge University Press. 2003.

[28] *"Truthful Topology Control inWireless Ad Hoc Networks with Selfish Nodes "*. Jianfeng CAI, Yunhuai LIU, Jie LIAN, Mo LI, Udo POOCH and Lionel NI.

[29] *"Preventing selfishness in open mobile ad hoc networks"*. H. Miranda and L. Rodrigues In Proc. of the Seventh CaberNet Radicals Workshop, October 2002.

[30] *"Cooperation in mobile ad hoc networks"*. Jiangyi Hu, Technical Report TR-050111. Computer Science Department, Florida State University, Tallahassee, FL 32306, January 2005.

[31] *"A Reputation-based Mechanism for Isolating Selfish Nodes in Ad Hoc Networks"*. M. Tamer Refaei, Vivek Srivastava, Luiz DaSilva, Mohamed Eltoweissy.

[32] *"Reputation-based System for Encouraging the cooperation of Nodes in Mobile Ad Hoc Networks"*. Tiranuch Anantvalee, Jie Wu.

[33] *"Sustaining cooperation in multi-hop wireless networks"*. Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. 2nd Symposium on Networked System Design and Implementation, Boston, MA, USA, May, 2005.

[34] *"TWOACK : Preventing Selfishness in Mobile Ad Hoc Networks"*, Kashyap Balakrishnan, Jing Deng, Pramod K. Varshney. 2005.

[35] *"A Simulation Analysis of Node Selfishness in MANET using NS-3"*, Satyanarayana Vuppala, Alokparna Bandyopadhyay, Prasenjit Choudhury, Tanmay De. Int. J. of Recent Trends in Engineering and Technology, Vol. 4, No. 1, Nov 2010.

[36] *"Modeling the Behavior of Selfish Forwarding Nodes to Stimulate Cooperation in MANET"*, Sundararajan, A.Shanmugam. International Journal of Network Security & Its Applications (IJNSA), Volume 2, Number 2, April 2010.

[37] *"Impact of selfish node concentration in manets"*, Shailender Gupta, C. K. Nagpal, Charu Singla, International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 2, April 2011.

[38] *"SORI : A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks"*, Qi He, Dapeng Wu, Pradeep Khosla Carnegie Mellon University, Dept. of Electrical & Computer Engineering, Pittsburgh, PA 15213, University of Florida, Dept. of Electrical & Computer Engineering, Gainesville, FL 32611.

[39] *"Observation-based Cooperation Enforcement in Ad hoc Networks"*, Sorav Bansal, Mary Baker, arXiv :cs/0307012v2 [cs.NI], 2003.

[40] *"Sécurité des communications dans les groupes dynamiques"*. S. GHAROUT. PhD thesis, Université de Technologie de Compiègne, 2009.

Résumé

Les systèmes de communication basés sur le paradigme ad hoc ont reçu beaucoup d'attention dans le passé et les efforts des chercheurs ont été dévoués surtout à produire des protocoles décentralisés pour garantir le routage des paquets en tenant compte de la mobilité des noeuds du réseau. Dans ce rapport on questionne l'hypothèse de base assumé pendent la conception de ces protocoles selon laquelle les entités impliquées dans l'exécution de ces algorithmes suivent précisément les règles dictées par les concepteurs du système sans pour autant prendre en considération les mauvais comportements des noeuds. Pour faire face aux déviations des noeuds du réseau du comportement préétabli afin d'en tirer des bénéfices ; on analyse les caractéristiques des solutions proposées pour ce problème d'égoïsme qui sont basées sur des mécanismes de coopération classés en deux catégories : les systèmes à base de réputation et les systèmes à base de crédit, Et cela après avoir fait une présentation générale de la problématique en montrant les exigences et les défis.

Mots clés **:** les réseaux Ad hoc, routage, crédit, réputation, égoïsme.

Abstract

Communication systems based on ad hoc paradigm have received much attention in the past and the researcher's efforts have been devoted mainly to produce decentralized protocols to ensure packet routing taking into account the mobility of network nodes. In this report we question the basic assumption assumed hang the design of these protocols that the entities involved in the implementation of these algorithms precisely follow the rules dictated by the designers of the system without taking into consideration the bad behavior of nodes. To cope with deviations from the behavior of the network nodes in order to obtain pre-established benefits, we analyze the characteristics of the proposed solutions to this problem of selfishness that are based on cooperation mechanisms fall into two categories : systems based reputation and credit-based systems, and this after a general presentation of the issue by showing the requirements and challenges.

Keywords **:** Ad hoc Networks, selfishness, reputation, credit.