N° d'ORDRE : 06/2009-M/IN

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie « Houari Boumediene »

Faculté d'Electronique et d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTERE

En : INFORMATIQUE

Spécialité : Informatique Mobile

Par : SENOUCI Mustapha Reda

Sujet

LA RÉPLICATION DE DONNÉES DANS LES RÉSEAUX MOBILES AD-HOC NON COOPÉRATIFS

Soutenu le 07/02/2009, devant le jury composé de :

Mr- M. AHMED-NACER, Professeur, USTHB.	Président
Mr- N. BADACHE, Professeur, USTHB.	Directeur de thèse
Mr- O. NOUALI, Maître de recherche, CERIST.	Examinateur
Mme- S. MOUSSAOUI, Maître de conférences, USTHB.	Examinateur
Mr- A. DERHAB, Chargé de recherche, CERIST.	Invité

Sommaire

Liste des Figures

Liste des Tableaux

A la mémoire de ma mère

Remerciements

Je remercie tout d'abord DIEU, le tout puissant, pour m'avoir donné la santé et le courage pour finir ce travail de recherche.

Je remercie M. Abdelouahid DERHAB, qui a encadré en collaboration avec M. Nadjib BADACHE ce travail de magistère avec enthousiasme, et a su me conseiller efficacement tout en m'accordant la liberté et la confiance d'agir.

Je remercie M. Nadjib BADACHE pour m'avoir fait confiance durant mon inscription à l'USTHB, pour son hospitalité et aussi pour m'avoir fait l'honneur de participer au Jury de ma soutenance.

Les membres du Jury m'ont honoré par leur participation à l'évaluation de ce travail ; je les remercie profondément.

J'exprime toute mon amitié à M. Mohamed AISSANI et M. Abdelhalim LEHTIHET. Je les remercie pour l'aide apportée durant la rédaction de ce mémoire, pour leurs encouragements et leur assistance aussi bien matérielle que morale me permettant ainsi de finaliser mon travail dans de bonnes conditions.

Je remercie chaudement ma famille et mes camarades.

Introduction générale

Un réseau mobile ad-hoc (MANET : Mobile Ad-hoc NETwork) est un système autonome de nœuds mobiles reliés par des liens sans fil dont l'union forme un graphe arbitraire. Les nœuds du réseau jouent le rôle de routeurs et sont libres de se déplacer aléatoirement et de s'organiser arbitrairement. En conséquence, la topologie du réseau peut changer rapidement et de manière imprévisible.

Les nœuds interagissent et peuvent coopérer pour s'échanger des services. Un nœud peut communiquer directement avec ses nœuds voisins, comme il peut servir de relais permettant à des nœuds se trouvant hors de portée radio les uns des autres de communiquer. Ces réseaux sont dits ad-hoc dans la mesure où ils ne nécessitent pas d'infrastructure fixe et leur déploiement est ainsi instantané. Leur existence peut être temporaire afin de répondre à un besoin ponctuel de communication.

L'accès aux données et services distants est l'une des applications les plus utilisées dans les réseaux. Dans les MANETs, les pannes imprévisibles des nœuds et le partitionnement fréquent du réseau peuvent considérablement diminuer la disponibilité des données et dégrader les performances des services d'accès à ces données. Une solution possible à ces problèmes est de mettre en œuvre un protocole de réplication.

A notre connaissance, tous les protocoles de réplication proposés dans la littérature supposent que tous les nœuds coopèrent entre eux. Un nœud qui sera choisi comme serveur de réplication acceptera le rôle et utilisera son espace de stockage pour sauvegarder les données des autres nœuds et son énergie pour répondre aux différentes requêtes. Puisque les ressources de ce nœud sont modestes, en termes de mémoire et d'énergie, il adoptera probablement certains comportements égoïstes. Une autre supposition faite par ces protocoles, qui n'est pas d'ailleurs toujours vraie, est que le réseau ne comprend pas de nœuds malicieux. Finalement, les protocoles de réplication proposés dans la littérature sont conçus pour des MANETs coopératifs. Pour un déploiement dans les MANETs non coopératifs, ces protocoles doivent être sécurisés.

Notre travail consiste à proposer un mécanisme de coopération qui permet le déploiement des protocoles de réplication dans les MANETs non coopératifs. Le mécanisme que nous proposons dans ce mémoire isole les nœuds malveillants de telle sorte que les nœuds malicieux soient ignorés et que les nœuds égoïstes soient encouragés à participer dans l'exécution du service de stockage. Pour cela, nous étudions les mécanismes de gestion de la confiance dans les services collaboratifs entre mobiles mutuellement suspicieux. Dans ce contexte, des mécanismes basés sur la notion de réputation (pour une évaluation de la confiance et pour une imputabilité) et de récompense (pour l'incitation à collaborer) sont de premier intérêt.

Le présent mémoire est organisé en cinq chapitres. Le premier chapitre est consacré aux réseaux sans fil et particulièrement aux MANETs. Nous donnons la définition, les classes d'appartenance, les caractéristiques spécifiques et les applications possibles des MANETs. Le deuxième chapitre présente une classification des protocoles de réplication et une description succincte de chaque classe. Le troisième chapitre est réservé au mécanisme de coopération et à la discussion des mécanismes conventionnels de sécurité et de leurs limites. Une classification,

selon notre point de vue, des mécanismes de coopération est proposée dans ce chapitre. Ensuite, nous décrivons avec détails les systèmes de punition, les systèmes de réputation et les systèmes de paiement les plus importants. La description de ces systèmes est suivie par une comparaison et une discussion. Le quatrième chapitre décrit l'architecture de notre solution, discute la relation entre notre mécanisme et le protocole de réplication et enfin détaille chaque composant de notre architecture. L'évaluation des performances de notre solution est présentée dans le cinquième chapitre. Finalement, nous concluons notre travail et nous discutons les diverses pistes possibles pour la suite de ce projet de recherche.

Chapitre 1. Les réseaux sans fil

Les révolutions opérées ces dernières années dans les domaines de l'informatique et des télécommunications convergent vers l'omniprésence des services du réseau Internet. Ceci sera réalisé grâce à des connexions sans fil dans des environnements fixes (domicile et lieu de travail) et mobiles (automobile, avion, train, bateau, etc.). Cette évolution accroîtra la proportion d'utilisation des ordinateurs portables et présentera de nouveaux enjeux en termes de travaux de recherche et d'investissement.

L'environnement sans fil, fondé sur l'utilisation des liaisons radios, est caractérisé par deux modes de communication, à savoir : le mode avec infrastructure et le mode sans infrastructure. Dans le mode avec infrastructure, la communication entre deux points ou nœuds quelconques du réseau passe nécessairement par une station de base. Par contre, dans le mode sans infrastructure (réseaux ad-hoc), chaque point ou nœud du réseau peut communiquer directement avec ses voisins sans avoir recours à une station de base intermédiaire.

Les réseaux ad-hoc appartiennent à la catégorie des réseaux sans fil. Ils sont caractérisés, entre autres, par une topologie dynamique, une bande passante limitée et des contraintes dans la consommation d'énergie. Le lien sans fil a la caractéristique essentielle de varier en fonction du temps et de l'espace. Le changement de l'état du lien sans fil de "bon" à "mauvais" et vice-versa peut intervenir de façon asynchrone pendant des laps de temps très courts. En plus de la variation du canal à petite échelle, il y a aussi une variation à grande échelle qui implique le conditionnement de l'état moyen du canal par la position de l'utilisateur et le niveau des interférences possibles.

1.1. Classification des réseaux sans fil

La classification des réseaux sans fil peut être basée sur (1) la formation et l'architecture du réseau, (2) la zone de couverture assurée, (3) la technologie d'accès utilisée par le réseau ou (4) l'application et l'utilisation du réseau.

1.1.1. Formation et architecture du réseau

Les réseaux sans fil peuvent être répartis en deux classes, selon que le réseau utilise ou non une infrastructure préexistante. La première classe concerne les réseaux sans fil basés sur une infrastructure préexistante et qui utilisent généralement le modèle de communication cellulaire. Dans ces réseaux, les Unités Mobiles (UM) échangent des données avec un Point d'Accès radio (PA) sur un réseau filaire existant plutôt que directement entre eux. Le point d'accès peut aussi être relié à d'autres points d'accès, ce qui permet d'agrandir le réseau sans fil en plaçant des équipements à des endroits éloignés l'un de l'autre (figure 1.1). Par contre, la deuxième classe regroupe les réseaux sans fil et sans infrastructure qui utilisent la communication sans fil entre les différents équipements mobiles du réseau, sans l'aide d'une quelconque infrastructure préexistante ou d'une administration centralisée. Dans de tels réseaux, les équipements sont munis d'émetteurs et de récepteurs sans fil, utilisant des antennes pour communiquer directement entre eux (figure 1.2).

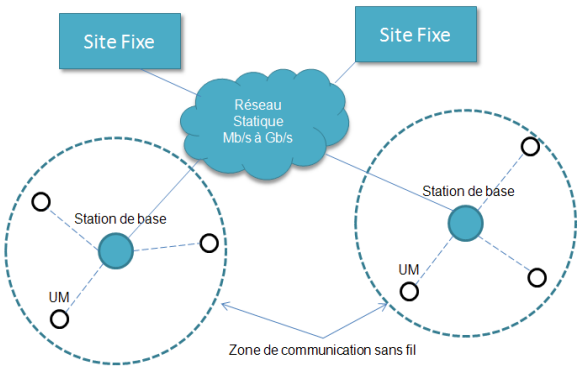


Figure 1.1. Modèle des réseaux cellulaires

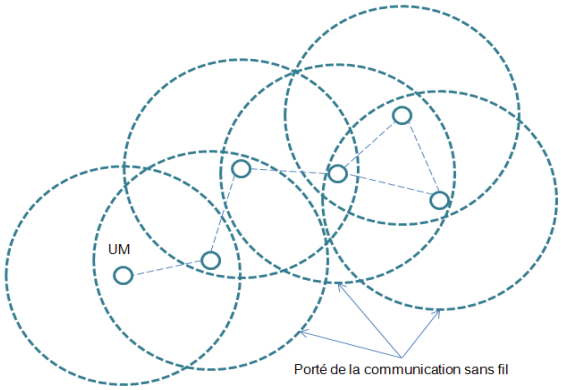


Figure 1.2. Modèle des MANETs

1.1.2. Zone de couverture du réseau

On distingue habituellement plusieurs catégories de réseaux sans fil avec infrastructure préexistante, selon le périmètre géographique offrant la connectivité. Néanmoins, l'architecture sous forme de réseau ad-hoc établie entre des ordinateurs dialoguant entre eux dans une zone de couverture limitée, est toujours possible.

- Le WPAN (*Wireless Personal Area Network*) concerne les réseaux sans fil de faibles portées (de l'ordre de quelques dizaines de mètres). Ce type de réseau sert généralement à relier des périphériques (imprimante, téléphone portable, appareils domestiques, etc.) ou un assistant personnel, à un ordinateur sans liaison filaire ou bien à réaliser une liaison sans fil entre deux machines très peu distantes.
- Le WLAN (*Wireless Local Area Network*) est un réseau sans fil permettant de couvrir l'équivalent d'un réseau local filaire d'entreprise, soit une portée d'environ une centaine de mètres. Il permet de relier les terminaux présents dans la zone de couverture entre eux.
- Le WMAN (*Wireless Metropolitan Area Network*) est connu aussi sous le nom de Boucle Locale Radio (BLR). La boucle locale radio offre un débit utile de 1 à 10 Mbit/s pour une portée de 4 à 10 kilomètres, ce qui destine principalement cette technologie aux opérateurs de télécommunication.
- Le WWAN (*Wireless Wide Area Network*), est également connu sous le nom de réseau cellulaire mobile. Il s'agit du réseau sans fil le plus répandu, puisque tous les téléphones mobiles sont connectés à ce type de réseau.

1.1.3. Technologie d'accès du réseau

Il existe plusieurs technologies d'accès utilisées dans les réseaux sans fil, parmi lesquelles on cite les principales :

- Bluetooth, lancée en 1994, avec un débit théorique de 1 Mbit/s pour une portée maximale d'une trentaine de mètres.
- HomeRF (*Home Radio Frequency*), lancée en 1998, avec un débit théorique de 10 Mbit/s et une portée d'environ 50 à 100 mètres sans amplificateur.
- La technologie ZigBee, permet d'obtenir des liaisons sans fil à très bas prix et avec une très faible consommation d'énergie, ce qui la rend particulièrement adaptée pour être directement intégrée dans les petits appareils électroniques.

- Les liaisons infrarouges sans fil de quelques mètres avec des débits pouvant monter à quelques Mbit/s. Cette technologie est largement utilisée pour les équipements domestiques mais souffre toutefois, des perturbations dues aux interférences lumineuses.
- Wi-Fi (*Wireless Fidelity* : IEEE 802.11), qui offre des débits allant jusqu'à 54 Mbit/s sur une distance de plusieurs centaines de mètres.
- HiperLAN2 (*High Performance Radio LAN 2.0*), permet d'obtenir un débit théorique de 54 Mbit/s sur une zone de couverture de 100 mètres.
- WiMAX (*Worldwide Interoperability for Microwave Access*), permettant d'obtenir des débits de l'ordre de 70 Mbit/s sur un rayon de plusieurs kilomètres.
- Le GSM (*Global System for Mobile communication*), le GPRS (*General Packet Radio Service*) et l'UMTS (*Universal Mobile Telecommunication System*).

1.1.4. Application du réseau

Une autre classification des réseaux sans fil peut être faite sur la base de la mission principale du réseau, c'est-à-dire son application. Par conséquent, on citera les réseaux d'entreprise, les réseaux domestiques, les réseaux tactiques, les réseaux sensoriels, les réseaux portables et les réseaux de véhicules automatisés.

1.2. Définition d'un réseau mobile ad-hoc (MANET)

Un réseau mobile ad-hoc (MANET) est généralement défini comme un système autonome dynamique composé de nœuds mobiles interconnectés par des liens sans fil, sans l'utilisation d'aucune infrastructure fixe et sans administration centralisée. Le groupe MANET de l'IETF fournit une définition précise dans son RFC 2501 [Corson99] : "Un réseau ad-hoc comprend des plateformes mobiles (appelées nœuds) libres de se déplacer sans contraintes". Ce système peut fonctionner d'une manière isolée ou s'interfacer avec des réseaux fixes aux travers des passerelles. Dans ce dernier cas, un réseau ad-hoc devient un réseau d'extrémité.

1.3. Caractéristiques des MANETs

Les réseaux ad-hoc héritent des spécificités liées aux réseaux sans fil telles que :

- La versatilité du médium radio : La puissance du signal radio reçu sur le nœud varie très fortement et très soudainement.
- L'asymétrie des liens sans fil : Un lien peut être opérationnel dans un sens et non pas dans l'autre.
- La non sécurisation des liens : Tout nœud à portée radio d'un nœud émetteur peut capter l'information.

En outre, il existe d'autres spécificités propres aux réseaux ad-hoc :

- Absence d'une administration centralisée : Dans un réseau filaire, il existe une distinction nette entre les nœuds terminaux (stations, hôtes) qui supportent les applications et les nœuds internes (routeurs) du réseau qui ont la charge d'acheminer les données. Cette différence n'existe pas dans un réseau ad-hoc car un nœud peut être émetteur, routeur ou récepteur. Le nœud émetteur génère ses propres données qui sont relayées par les

nœuds routeurs pour atteindre le nœud récepteur. Chaque nœud opère indépendamment de toute autre entité dans le réseau.

- Nœuds Mobiles et topologie dynamique : La mobilité des nœuds consiste à l'évidence une caractéristique très spécifique des réseaux ad-hoc. Cette mobilité est intrinsèque au fonctionnement du réseau. Elle se distingue du caractère nomade (mobilité des seuls nœuds terminaux) ou de l'itinérance (équipements statiques mais pouvant être déplacés). Dans un réseau ad-hoc, la topologie peut changer rapidement de façon aléatoire et non prédictible. En effet, un nœud peut joindre un réseau, changer de position ou quitter le réseau. Les techniques de routage des réseaux filaires, basées sur des routes préétablies, ne sont pas adéquates aux réseaux ad-hoc. Ainsi, dans ces derniers réseaux, les protocoles de routage doivent prendre en considération la mobilité des nœuds en supportant la maintenance des routes. En cas de défaillance, les protocoles de routage doivent reconstruire les routes en un temps limité tout en évitant la surcharge du réseau par un surplus de paquets de contrôle.
- Contrainte d'énergie : La consommation d'énergie constitue un facteur important pour les équipements fonctionnant à base de batteries. Les nœuds d'un réseau ad-hoc sont alimentés via des batteries dont la durée de vie est limitée. Il est, alors, important que les protocoles mis en place dans les réseaux ad-hoc prennent en compte cette contrainte.
- Ressources limitées : La limitation de la bande passante (par exemple : 11Mbit/s pour un réseau 802.11b) génère une difficulté supplémentaire pour les protocoles de routage. En effet pour gérer la forte dynamique dans un réseau mobile ad-hoc, les protocoles de routage génèrent beaucoup plus de messages de contrôle par rapport à un réseau filaire. Ces messages doivent être traités d'une façon prioritaire pour la prise en compte rapide des changements dans la topologie du réseau ad-hoc. Puisque les ressources des réseaux ad-hoc sont limitées, les protocoles doivent maintenir l'overhead (rapport entre les paquets de contrôle et les paquets de données) généré dans un intervalle raisonnable.
- Interférences : Dans les réseaux filaires, l'émission des données entre deux nœuds utilise un lien filaire reliant ces deux nœuds. Tous les nœuds qui n'ont pas accès à ce lien se voient privés de la trace de cet envoi. Par contre dans les réseaux ad-hoc, le même processus se traduit autrement. Le médium radio étant partagé, chaque paquet émis est physiquement reçu par tous les nœuds se trouvant dans la zone de portée radio du nœud émetteur, tandis que les nœuds se trouvant dans la zone d'interférence détectent l'occupation du médium. Seul le nœud destinataire récupère le paquet envoyé. Dans les réseaux 802.11, le partage du médium radio est effectué par des protocoles basés sur le mécanisme CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*). Dans le 802.11, par exemple, les collisions de paquets font décroître le débit utile du médium.
- Sécurité physique limitée : La sécurité des échanges est un problème inhérent à tout système de communication [Muhlethaler02]. Les réseaux sans fil sont par nature trop sensibles aux problèmes de sécurité. Pour les réseaux ad-hoc où la topologie est dynamique, les possibilités d'insérer une attaque dans le réseau sont plus grandes. La détection d'une intrusion est plus délicate dans la mesure où l'absence d'une administration centralisée pose le problème de la réaction du réseau sans fil vis-à-vis de cette intrusion. Pour palier à ce problème, des mécanismes d'authentification et de chiffrement s'avèrent nécessaires.

1.4. Applications des MANETs

Historiquement, le premier domaine d'application des réseaux ad-hoc fut le domaine militaire. Dans les années 1970, le DARPA (*Defence Advanced Research Projects Agency*) propose Packet Radio Network. Ce protocole permet de déployer un mécanisme de communication entre les différents groupes d'unités par l'intermédiaire de véhicules qui communiquent ensemble par liaison radio. Ces recherches ont mis en avant le potentiel de ce type de réseaux ainsi que leurs limitations. Cependant, ce protocole présente de nombreux défauts, notamment en ce qui concerne la mobilité et la taille de l'équipement. En effet, ce protocole se base sur l'hypothèse d'un mouvement assez lent et de faibles changements de topologie. De plus, les possibilités de l'électronique des années 1970 rendent le matériel encombrant et difficile à utiliser. L'intérêt des militaires pour une telle technologie s'explique par le caractère particulièrement adapté des réseaux ad-hoc aux situations hostiles. En effet, comme le réseau est auto organisé et qu'il ne nécessite pas d'infrastructure, il peut être déployé rapidement, sans difficulté et offrir une bonne tolérance aux pannes. D'où l'utilisation possible également dans le cadre de sinistres, comme les tremblements de terre ou les incendies, pour permettre aux équipes de sauvetage de communiquer alors que les infrastructures de communication classiques sont détruites ou pour permettre aux survivants d'établir un réseau aidant à leur localisation. De plus, la possibilité pour les communications d'emprunter plusieurs routes renforce la fiabilité d'un tel réseau ce qui est indispensable dans un tel contexte.

La non nécessité d'une infrastructure préexistante, et l'auto configuration, autrement dit, la simplicité et la rapidité de déploiement des MANETs, les rend capables d'apporter un potentiel commercial et un avantage industriel et militaire importants. Les opérations tactiques et les missions navales étaient parmi les premières applications. La nature dynamique des opérations militaires veut que l'on ne puisse pas compter sur l'accès à une infrastructure fixe durant la bataille. Un réseau ad-hoc peut fournir une plate-forme convenable pour répondre à tous ces besoins en mettant en œuvre une structure dynamique distribuée sans fil et en assurant une connectivité multi sauts entre les différentes composantes du réseau.

Il est évident que dans un contexte commercial les réseaux ad-hoc peuvent également servir à former, d'une manière très simple, des réseaux locaux. Dans le cas de réunions temporaires, comme des conférences, l'organisation est simplifiée, par la non nécessité de câblage. Cependant, des progrès sont encore à faire, notamment en ce qui concerne les couches physiques, car ce genre d'application peut nécessiter une bande passante importante, ce qui n'est pas encore fournit par la technologie Wi-Fi à l'heure actuelle.

Depuis 1990, l'infrastructure Internet a connu une augmentation considérable due à la révolution de la micro-informatique qui a crée un environnement adéquat pour implémenter les idées des réseaux de paquets radios. Plusieurs projets ont été élaborés durant les années 90 comme le WINGs (*Wireless Internet Gateways*) et le MMWN (*Multimedia Mobile Wireless Network*) qui ont été utilisés aux USA et en Europe [Corson99]. Le Tactical Internet (TI) a été implémenté par l'armée américaine en 1997 dans le but d'étendre l'implémentation des MANETs à grande échelle pour agrandir la zone de couverture [Corson99].

Le ELB ACTD (*Extending the Littoral Battle-space Advanced Concept Technology Demonstration*) de 1999 est une exploration de déploiement pour démontrer la réalisabilité des concepts de combat du corps de la marine américaine qui exigent la communication au delà de

l'horizon à partir les bâtiments de guerre vers les soldats sur terre via un relais aérien. ELB ACTD a eu un succès dans l'utilisation des relais aériens pour connecter les utilisateurs.

Dans le cadre de l'informatique omniprésente (telle que décrite par Mark Weiser [Weiser91]), les réseaux ad-hoc peuvent servir à relier entre eux tous les équipements de la maison ou établir les liens entre les différents composants informatiques des vêtements (dans le cadre de l'informatique vestimentaire ou *Wearable Computing*). Dans ce cas, on parle non plus de LAN (*Local Area Network*) mais de PAN (*Personal Area Network*) et le routage ad-hoc peut être utilisé pour faire communiquer tous les éléments grâce à des équipements de faible puissance, et donc de faible portée (ce qui est un avantage, tant au point de vue de la consommation énergétique qu'au point de vue de la santé de l'utilisateur).

Enfin, les réseaux ad-hoc s'avèrent très utiles dans le cas de mesures en milieu hostile. On parle alors de réseaux de capteurs [Carle04]. Les capteurs, chargés de mesurer les propriétés physiques des environnements (comme la température, la pression, etc.), sont dispersés (le plus souvent largués d'un avion ou d'un hélicoptère) par centaines, voire par milliers sur le site, effectuent leurs mesures et envoient les résultats à une station de base par l'intermédiaire d'un routage ad-hoc à travers le réseau, souvent très dense, ainsi formé. La caractéristique principale d'une telle application est la forte contrainte d'énergie, de mémoire et la faible capacité de traitement de ces dispositifs. Comme on peut le constater, les MANETs auront un très large potentiel dans un futur proche et l'intérêt que portent les chercheurs pour ce domaine s'en explique donc très largement. De nombreux défis sont posés avant de pouvoir utiliser pleinement ce type de réseaux.

1.5. Les réseaux contrôlés et les réseaux ouverts

A l'origine, les applications exploitant les réseaux ad-hoc ont été envisagées principalement pour des situations de crise (champs de bataille, opérations de secours, etc.). Dans de telles applications, tous les nœuds du réseau appartiennent à une même autorité (une unité militaire, une équipe de secours, etc.) et ont un but commun. Cependant, les technologies sans fil se sont sensiblement améliorées ces dernières années et des dispositifs peu coûteux basés sur la norme IEEE 802.11 ont envahi le marché et le déploiement des réseaux ad-hoc pour des applications commerciales est devenu réaliste. Des exemples incluent le déploiement des réseaux ad-hoc pour l'automobile ou pour la fourniture d'équipements de communication pour les régions éloignées ou les périmètres physiquement délimités (centres commerciaux, aéroports, etc.). Dans ces réseaux, les nœuds n'appartiennent pas à une même organisation ni à une même autorité et ils ne poursuivent pas un but commun. En outre, les réseaux commerciaux pourraient être plus grands et avoir une plus longue vie ; de plus ils peuvent être complètement autonomes, signifiant que le réseau fonctionnerait seulement grâce à l'opération des utilisateurs.

Selon le type d'application, il est possible de définir deux catégories principales de réseaux ad-hoc : les réseaux contrôlés et les réseaux ouverts. Nous nous referons aux réseaux ad-hoc contrôlés lorsque la phase d'initialisation du réseau peut être appuyée par une infrastructure provisoire et lorsqu'une confiance a priori est disponible entre les nœuds du réseau. Des relations de confiance a priori peuvent être établies, par exemple, par une autorité contrôlant les nœuds du réseau ou par une organisation commune entre les utilisateurs. D'autre part, dans un réseau ad-

hoc ouvert, les nœuds sont entièrement autonomes et dans la majorité des cas ne peuvent pas compter sur une infrastructure préétablie pour l'initialisation et l'opération de réseau. De plus, puisque les nœuds sont actionnés par les utilisateurs qui n'appartiennent pas nécessairement à la même organisation, et ne partagent pas une même autorité, une relation de confiance a priori entre les nœuds n'est pas disponible. La confiance entre les nœuds doit être établie par des mécanismes spécifiques conçus en fonction du scénario offert par un environnement ouvert.

1.6. La non coopération dans les réseaux mobiles ad-hoc

Le bon fonctionnement d'un réseau ad-hoc repose entièrement sur une relation de confiance implicite entre les nœuds. Le principe de fonctionnement des réseaux ad-hoc rend la coopération des nœuds une condition essentielle.

1.6.1. Définition de la coopération

Le mot "coopération" est formé de deux mots latins : "*cum*" (avec) et "*operare*" (faire quelque chose). La coopération est l'action de coopérer, de participer à une œuvre, à un projet commun. La coopération est la capacité de collaborer à cette action commune ainsi que les liens qui se tissent pour la réaliser. C'est un mode d'organisation sociale qui permet à des individus ayant des intérêts communs de travailler ensemble avec le souci de l'objectif général. Elle nécessite un certain degré de confiance et de compréhension.

Dans le cas des MANETs, la coopération est vue comme la bonne volonté d'un nœud d'exécuter des fonctions de gestion du réseau au profit d'autres nœuds.

1.6.2. La non coopération

La coopération engendre des échanges de messages entre les nœuds qui nécessitent un coût énergique non négligeable, ce qui peut mener les nœuds à un comportement égoïste, particulièrement dans un environnement alimenté en énergie par des batteries, tels que les MANETs. En effet, il n'y a aucune raison de supposer qu'un nœud participe activement dans le réseau car il peut, par exemple, modifier sa fonction d'expédition de paquets afin de prolonger sensiblement sa durée de vie.

Un nœud qui ne suit pas les spécifications des protocoles réseau est considéré comme non coopératif. On distingue deux types de comportement : (i) les nœuds malveillants et (ii) les nœuds égoïstes. Un nœud égoïste est un nœud qui veut profiter du réseau. Il coopère lorsque cette coopération maximise son propre bénéfice, sinon il économise ses ressources en refusant de coopérer. Un nœud malveillant (ou malicieux) est un nœud qui ne vise pas la préservation de ses ressources, mais plutôt essaye de monter des attaques pour endommager le fonctionnement normal du réseau.

La non coopération concerne chaque couche de la pile protocolaire. Dans [Conti04], les auteurs détaillent la non coopération au niveau des couches MAC, réseau (routage et expédition), transport et application.

La non-coopération a des conséquences graves sur le réseau, telles que : (i) le partitionnement du réseau, (ii) la congestion, (iii) la dégradation des performances réseau ; etc.

1.6.3. Les réseaux ad-hoc coopératifs et non coopératifs

Dans le cas où tous les nœuds participent activement, on parle d'un réseau ad-hoc coopératif (tous les nœuds coopèrent pour supporter les tâches du réseau). Un réseau ad-hoc non coopératif est un réseau qui comprend des nœuds égoïstes et/ou malicieux.

1.7. Conclusion

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 surtout dévoués à produire des protocoles décentralisés en tenant compte de la mobilité des nœuds. La nature des MANETs rend difficile la conception d'un protocole distribué car nous devons tenir compte des changements fréquents de la topologie, du partitionnement du réseau et des contraintes d'énergie.

L'hypothèse de base assumée pendant la conception des protocoles distribués pour les MANETs stipule que les entités impliquées dans l'exécution de ces protocoles suivent précisément les règles dictées par ces protocoles. Sachant que les nœuds sont actionnés par des utilisateurs qui n'appartiennent pas nécessairement à la même organisation, il peut y avoir des déviations des nœuds du réseau du comportement préétabli. On parle dans ce cas d'un comportement non-coopératif qui a des conséquences graves sur le réseau.

Dans les MANETs, les nœuds se déplacent librement et leur énergie est souvent limitée (utilisation des batteries), ce qui cause des pannes de nœuds et des ruptures fréquentes de liens. La panne de certains nœuds considérés critiques peut fractionner le réseau en plusieurs partitions disjointes. Cette situation réduit considérablement l'accès aux données et augmente la probabilité de trouver des données inconsistantes. De plus, la nature partagée du canal sans fil augmente le coût d'un accès à distance. Une solution possible à ces problèmes est de mettre en œuvre un protocole de réplication.

Les protocoles de réplication proposés dans la littérature sont conçus pour des réseaux ad-hoc coopératifs. Dans le chapitre suivant, nous discutons brièvement ces protocoles pour ensuite examiner les nouveaux problèmes posés auxquels nous devrons faire face.

Chapitre 2. La réplication dans les MANETs

L'accès aux données et services distants est l'une des applications les plus utilisées dans les réseaux fixes et mobiles. Dans les réseaux mobiles ad-hoc, les pannes imprévisibles des nœuds et le partitionnement fréquent du réseau peuvent considérablement diminuer la disponibilité des données et des services et dégrader les performances d'accès à ces derniers. Une solution possible pour ces problèmes est de mettre en œuvre un protocole de réplication.

2.1. Définition

La réplication est une technique fondamentale utilisée dans les systèmes répartis. Elle consiste à stocker les mêmes données et/ou services sur plusieurs sites et garantir ainsi une disponibilité élevée des données, une fiabilité, une tolérance aux fautes et une meilleure performance du système. Un site assurant une telle mission est appelé "serveur de réplication".

2.2. Les causes de la réplication

La réplication est une technique de base utilisée dans plusieurs systèmes, tels que le stockage distribué (distributed storage systems) et la sauvegarde (back-up systems). Nous recourons à la réplication de données et/ou services pour diverses raisons, telles que :

- Améliorer la disponibilité des données ;
- Améliorer l'accessibilité des données ;
- Diminuer le temps d'accès et le coût d'accès[1] aux données et/ou services ;

2.3. Nouvelles contraintes spécifiques aux MANETs

En plus des contraintes de disponibilité et de performance qui ont été largement discutées dans les réseaux fixes, la réplication dans les réseaux ad-hoc doit considérer des contraintes additionnelles résultantes des caractéristiques spécifiques à ces derniers. Ces contraintes sont :

- Partitionnement du réseau : en raison du partitionnement fréquent du réseau, les chances d'accéder à une donnée deviennent faibles puisque les nœuds mobiles peuvent ne pas être dans la même partition que le nœud qui détient la donnée. Le protocole de réplication devrait répliquer les données dans des futures partitions (avant l'occurrence du partitionnement de réseau).
- Consommation de l'énergie : les nœuds mobiles utilisent des batteries de faible capacité. Un serveur de réplication peut servir de nombreux clients. Pour répondre aux différentes requêtes, le nœud en question encourt une consommation importante de son énergie. Pour améliorer la disponibilité des données, le protocole de réplication devrait dupliquer les données critiques sur les nœuds possédant une énergie résiduelle importante par

[1] Le nombre de messages générés pour chaque opération d'accès.

rapport aux autres nœuds du réseau. D'ailleurs, il devrait également dupliquer des données de telle manière que la consommation de l'énergie des serveurs soit réduite et soit équilibrée entre tous les serveurs du réseau.

- Passage à l'échelle (*scalability*) : lorsque la taille du réseau augmente, une requête envoyée par un nœud client peut traverser un long chemin pour atteindre le nœud serveur, ce qui augmente le coût et la latence de la requête. De plus, un grand nombre de nœuds clients provoque un grand nombre d'accès au canal sans fil, ce qui diminue considérablement la bande passante disponible et augmente le temps d'accès au canal. Le protocole de réplication devrait être conçu de sorte que ses performances ne soient pas considérablement affectées si le nombre de nœuds ou la taille du réseau augmente.

2.4. Classification des algorithmes de réplication

Le partitionnement du réseau, la consommation d'énergie et le passage à l'échelle sont les points les plus importants à considérer lors de la conception d'un protocole de réplication pour les MANETs. Basée sur ces points, la figure 2.1 présente une taxonomie des protocoles de réplication spécifiques aux MANETs.

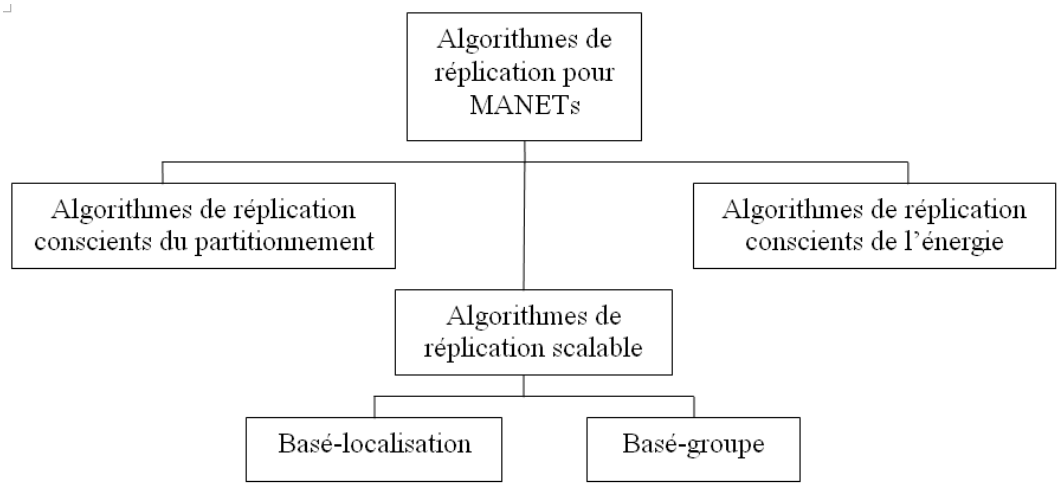


Figure 2.1. Taxonomie des protocoles de réplications pour MANETs [Derhab07]

Les algorithmes de réplication destinés aux MANETs peuvent être classés en : (i) Algorithmes de réplication conscients du partitionnement, (ii) Algorithmes de réplication conscients de l'énergie et (iii) Algorithmes de réplication supportant le passage à l'échelle. Les algorithmes de réplication conscients du partitionnement [Derhab07] essayent de prévoir le partitionnement du réseau et de dupliquer les données à l'avance. Les algorithmes de réplication conscients de l'énergie [Derhab07] sélectionnent leurs serveurs réplicas en se basant sur l'état de leurs batteries. Les algorithmes de réplication supportant le passage à l'échelle essayent de concevoir des solutions localisées, dans lesquelles chaque nœud peut prendre des décisions en se basant seulement sur les informations fournies par des nœuds se trouvant à une distance constante (le voisinage direct pour une distance égale à un saut, le voisinage à deux sauts pour une distance égale à 2). Les algorithmes de réplication supportant le passage à l'échelle suivent deux approches : (i) approche basée localisation [Derhab07] et (ii) approche basée groupe [Derhab07]. Dans la première, un

nœud choisit ses serveurs de réplication en fonction de leurs positions et la région où ils sont localisés. Dans la deuxième, tous les nœuds sont mis dans des groupes disjoints. Un nœud leader est désigné pour chaque groupe afin de s'occuper des requêtes de lecture et de mise à jour.

2.4.1. Algorithmes de réplication conscients du partitionnement

Le partitionnement du réseau dégrade considérablement la disponibilité des données. Un bon algorithme de réplication devrait assurer un niveau acceptable de disponibilité de données, même avec le partitionnement du réseau. Pour cela, la prédiction du partitionnement du réseau est le grand problème que ces techniques essayent de résoudre. Parmi les techniques de prédiction utilisées, nous citons :

- Utilisation du GPS : Cette classe de techniques utilise les informations de localisation fournies par le GPS pour prédire le partitionnement du réseau. Dans [Wang02] par exemple, les auteurs proposent une solution où chaque nœud est au courant de sa position et de sa vitesse (utilisation du GPS). Les auteurs supposent un modèle de mobilité de groupe (le modèle RPGM) où chaque nœud serveur essaye de prédire le partitionnement du réseau pour répliquer les données à l'avance. Ils utilisent leurs connaissances en matière de positions et de vitesses des nœuds pour calculer le temps et l'endroit d'occurrence des partitionnements réseau.
- Utilisation d'une métrique réseau : Cette classe de techniques utilise des métriques réseau pour prédire le partitionnement du réseau. Dans [Hauspie03] par exemple, les auteurs proposent une nouvelle métrique pour détecter le partitionnement réseau sans l'utilisation du GPS. Cette métrique basée sur l'ensemble des chemins disjoints[2] entre le nœud client et le nœud serveur permet de prédire le partitionnement réseau pour répliquer les données à l'avance. La décision pour répliquer un service ou des données est prise aussi lorsque la connexion entre un client et un serveur devient mauvaise en termes de fiabilité, bande passante, délai, etc.

La majorité des techniques de prédiction basées sur l'utilisation du GPS sont limitées à des modèles de mobilité de groupe, ce qui limite leur champ d'application ; en plus le GPS est très gourmand en énergie, il est donc considéré non approprié pour les petits dispositifs. D'autre part, les techniques qui n'utilisent pas le GPS requirent le calcul de certaines métriques, ce qui cause un supplément de charge réseau, de puissance de calcul et d'espace de stockage.

2.4.2. Algorithmes de réplication conscients de l'énergie

Les algorithmes de réplication conscients de l'énergie tiennent compte de la consommation d'énergie des nœuds serveurs et clients lors de la sélection des serveurs réplicas. Dans [Thanedar04] par exemple, les auteurs ont proposé un algorithme, appelé *Expanding Ring Replication (ERR),* qui combine deux approches de livraison de données (push-based et pull-based). Dans l'approche pull-based, lorsqu'un nœud veut accéder à des données, il diffuse à son

[2] Un ensemble de chemins disjoints est un ensemble de chemins qui n'ont aucun nœud commun excepté le nœud client et le nœud serveur.

voisinage un message contenant la description des données exigées. Si la requête n'est pas satisfaite au bout d'un temps seuil, le nœud envoie une requête directement au serveur de données. Dans l'approche push-based, le serveur de données mesure la fréquence des requêtes f_i pour chaque donnée D_i. Si f_i dépasse une valeur seuil, λ_i, fixée par le serveur de D_i, le serveur décide de répliquer la donnée sur un ou plusieurs nœuds mobiles capables. La fonction de capacité considère des paramètres tels que l'espace mémoire disponible, l'énergie résiduelle et la puissance du processeur. Dans cette solution, les auteurs ne considèrent pas le processus de mise à jour des données et le problème d'inaccessibilité du serveur de données.

2.4.3. Algorithmes de réplication supportant le passage à l'échelle

2.4.3.1. L'approche basée groupe

Les algorithmes basés sur la notion de groupe essayent de construire des groupes disjoints[3] où chaque groupe est géré par un nœud leader. Dans [Yu05] par exemple, un algorithme d'élection est utilisé pour construire les groupes. Dans la première phase de l'algorithme, les nœuds diffusent leurs caractéristiques (identifiant, espace de stockage résiduel, etc.), chaque nœud construit une liste triée par la capacité de stockage, les M premiers nœuds seront sélectionnés comme des leaders de groupe. Les auteurs supposent que le nombre de clusters est M et que ce dernier est connu par tous les nœuds. Après cette première phase, chaque nœud leader de groupe crée son propre groupe en diffusant un message spécial (CH-ANNOUNCE), chaque nœud rejoint le leader dont son message est arrivé le premier. Après la création des groupes, les leaders gèrent les requêtes de lecture et de mise à jour. La maintenance des groupes peut s'avérer très coûteuse en terme de charge réseau, ce qui constitue l'inconvénient majeur de cette approche.

2.4.3.2. L'approche basée localisation

Les solutions de l'approche basée localisation supposent qu'un nœud n'a pas besoin de connaître l'identifiant du serveur de réplication à qui il devrait envoyer ses requêtes et ses mises à jour, mais il doit savoir seulement la région où il devrait envoyer ses requêtes. En utilisant un protocole de routage géographique [Liao01, Jain01], les requêtes peuvent être envoyées vers des régions ou des positions fixes et non pas vers des nœuds. Certains nœuds ou tous les nœuds à l'intérieur des régions ou près des positions peuvent agir en tant que serveurs. En appliquant cette méthode, la nature dynamique du réseau ad-hoc est masquée par un schéma statique.

Un protocole de réplication, qui est implémenté sur un protocole de routage basé localisation, doit découvrir l'endroit des serveurs à qui il veut envoyer ses répliques. Des informations de localisation sont fournies par un service de localisation. Le rôle d'un service de localisation et de mapper l'identifiant d'un nœud dans sa position géographique. Pour distribuer l'information de localisation à un ensemble de serveurs, le service de localisation utilise un protocole de routage basé localisation. Dans [Ratnasamy02] par exemple, les auteurs utilisent une table de hachage géographique. Dans ce système, une donnée est associée à une clé. Le système hache les clés

[3] Chaque nœud appartient à un seul groupe à un instant donné.

dans des coordonnées géographiques et stocke l'information sur le nœud le plus proche géographiquement de ces coordonnées.

2.5. L'aspect sécurité

2.5.1. Définitions

Dans cette section, nous redéfinissons quelques concepts liés à la coopération en considérant le cas de la réplication de données.

- **La coopération :** Dans le cas de la réplication, un nœud est considéré comme coopératif s'il suit la spécification du protocole de réplication. Dans le cas où il est choisi comme étant un serveur de réplication, il utilisera son espace de stockage pour sauvegarder les données des autres nœuds et son énergie pour répondre aux différentes requêtes.
- **La non coopération :** La non coopération est une déviation du comportement coopératif. Dans le cas de la réplication, un nœud peut refuser de stocker les données des autres nœuds ou de répondre aux requêtes de lecture et/ou de mise à jour.
- **Le comportement égoïste :** Un nœud égoïste pour un protocole de réplication est un nœud qui utilise les ressources du réseau lorsqu'il en a besoin et refuse d'être un serveur de réplication pour les autres nœuds. Ceci pour économiser son énergie et son espace de stockage.
- **Le comportement malveillant :** Un nœud malicieux ne vise pas la préservation de ses ressources, mais essaye plutôt de monter des attaques de type déni de service (DoS) pour endommager le fonctionnement normal du réseau. Un exemple de telle attaque est l'inondation des serveurs de réplication avec de fausses requêtes de sauvegarde.
- **Réseau coopératif et réseau non coopératif :** Lorsque tous les nœuds du réseau participent activement dans le déroulement du protocole de réplication, on parle d'un réseau ad-hoc coopératif. Dans le cas contraire, on parle d'un réseau ad-hoc non coopératif. Un nœud peut être coopératif dans un service et non coopératif dans un autre. Dans le cas de la réplication, nous considérons un nœud comme coopératif s'il est coopératif dans le service de réplication.

2.5.2. Les problèmes de non coopération

Les protocoles de réplication proposés dans la littérature supposent que tous les nœuds du réseau coopèrent entre eux. Un nœud choisi comme étant un serveur de réplication va tout simplement accepter ce rôle. Puisque les ressources en termes d'espace de stockage et d'énergie de ce nœud de réplication sont modestes, il adoptera probablement certains comportements égoïstes.

Une deuxième hypothèse faite par les protocoles de réplication, et qui n'est pas toujours vraie, est que le réseau ne comprend pas de nœuds malicieux.

La présence d'un tel comportement exige des mécanismes de sécurité pour faire face aux différentes menaces qui peuvent affecter la disponibilité des données et le bon fonctionnement du

protocole de réplication. Comme menaces nous citons : (1) la perte de données suite à l'épuisement des batteries, aux dommages physiques, et au vol, (2) un serveur de réplication malicieux peut affecter l'intégrité des données stockées, (3) opération de lecture effectuée par un nœud non autorisé, (4) un nœud égoïste refuse de collaborer, (5) un nœud malicieux peut provoquer un déni de service par l'inondation des serveurs avec de fausses requêtes de sauvegarde.

En résumé, les protocoles de réplication proposés dans la littérature sont conçus pour les MANETs coopératifs. Pour leur déploiement dans les MANETs non coopératifs, les protocoles de réplication doivent être sécurisés.

2.6. Conclusion

Dans ce chapitre, nous avons défini la technique de réplication et discuté les nouvelles contraintes spécifiques aux MANETs, ensuite nous avons présenté une classification des protocoles de réplication pour les MANETs et discuté brièvement chaque classe.

Dans le chapitre suivant, nous détaillons les mécanismes de coopération, proposés dans la littérature, les plus importants et qui sont appropriés à notre besoin. Nous proposons une taxonomie des mécanismes de coopération suivie par une comparaison et nous discutons les résultats de cette comparaison. Finalement, nous formulons les résultats de cette étude sous forme de recommandations utiles pour la conception de notre nouveau mécanisme.

Chapitre 3. Les mécanismes de coopération

Traditionnellement, la sécurité a été considérée comme une question importante pour les réseaux avec infrastructure. Pour sécuriser de tels réseaux et les applications supportées, les services de sécurité suivants ont été traités par la communauté des chercheurs : la confidentialité, l'intégrité, l'authentification et la non répudiation.

La confidentialité assure dans le réseau qu'une certaine information n'est jamais révélée aux entités non autorisées. La transmission sur réseau des informations sensibles (stratégiques ou monétaires) exige de la confidentialité. La fuite d'information vers des oreilles indiscrètes pourrait avoir des conséquences graves. L'intégrité garantit qu'un message étant transféré n'a pas été corrompu. Un message pourrait être corrompu en raison des fautes non intentionnelles ou en raison des attaques malveillantes sur le réseau. L'authentification permet à un nœud d'assurer l'identité du nœud avec qui il communique. Sans authentification, un adversaire pourrait se faire passer pour un autre nœud afin d'accéder à une ressource non autorisée ou à une information sensible, interférant ainsi l'opération correcte des autres nœuds. Enfin, la non répudiation assure que la source d'un message ne peut pas nier l'envoi du message. D'autres objectifs de sécurité (par exemple, autorisation, détection d'intrusion, etc.) existent, ils sont une préoccupation pour certaines applications.

Les services de sécurité traditionnels tels que la confidentialité, l'intégrité, l'authentification, et la non répudiation ont été le sujet d'une intense recherche ces dernières années.

3.1. Les mécanismes de sécurité conventionnels

Les algorithmes cryptographiques employés par les mécanismes de sécurité se basent sur un service de gestion et de distribution de clefs symétriques ou asymétriques. Selon le type de clef, les algorithmes sont dits symétriques ou asymétriques (ou à clef publique). Le choix du type d'algorithme a un impact immédiat sur les exigences en terme de puissance de calcul disponible de chaque nœud du réseau.

Les systèmes cryptographiques traditionnels à clef publique sont basés sur l'algorithme RSA et exigent que la clef publique de tout utilisateur soit certifiée et que le certificat soit délivré et signé par une autorité de certification (CA). Ceci est instauré afin de prouver l'authenticité de la clef publique appartenant à un utilisateur. N'importe quel participant qui veut employer une clef publique doit d'abord vérifier le certificat correspondant pour vérifier la validité de la clef publique. Lorsque beaucoup de CAs son impliqués entre deux utilisateurs, des rapports de confiance entre ces CAs doivent également être vérifiés. En fait, les systèmes à clef publique se fondent sur une infrastructure à clef publique (PKI) qui est employée afin de contrôler le rapport de confiance entre les entités (d'une façon hiérarchique). Cependant, les solutions basées sur les certificats traditionnels sont affectées par l'inconvénient principal qui est la révocation des clefs publiques et du certificat correspondant, ce qui représente un problème et exige des capacités importantes de

stockage et de calcul. D'ailleurs, les services de révocation exigent souvent que l'autorité de confiance soit en ligne.

Les MANETs sont des systèmes répartis composés d'entités autonomes qui nécessitent de la coopération afin de fonctionner correctement. Le concept fondamental est l'exploitation de la synergie, résultat de la collaboration entre les composants du réseau, afin de fournir des services à chacun. La condition de base pour rendre opérationnel le paradigme coopératif est la contribution supposée de toutes les entités qui composent et, en même temps, se servent du système. Cependant, le déploiement récent des MANETs pour des applications civiles détend l'hypothèse initiale sur une coopération implicite entre les nœuds. Les applications des MANETs pour des situations de crise (militaire ou urgence) prévoient que tous les nœuds du réseau appartiennent à une même autorité et ont un objectif commun. Par conséquent, la coopération des nœuds peut être assumée. Dans le contexte des applications civiles, les nœuds n'appartiennent pas réellement à une même autorité et, par conséquent, la coopération des nœuds ne peut pas être directement assumée. L'absence de toute autorité centrale, pour garantir la collaboration, motive une tendance possible des entités à se conduire mal en n'adhérant pas au paradigme coopératif.

Le manque de coopération peut être provoqué par des entités qui sont malveillantes ou qui sont égoïstes. Les entités malveillantes visent intentionnellement à endommager l'opération normale du réseau. Ce genre de malveillance provoque plusieurs problèmes de sécurité qui peuvent être résolus par des services traditionnels de sécurité. Comme exemple, une entité malveillante peut effectuer une attaque de rupture d'acheminement par lequel l'attaquant envoie des paquets forgés pour créer une boucle d'acheminement ou pour diviser le réseau. Ce genre d'attaque peut être empêché par la vérification d'intégrité et d'authenticité des messages de contrôle d'acheminement. D'autre part, une entité égoïste ne prévoit pas d'endommager directement le fonctionnement global du système. Ce genre de comportement produit une nouvelle classe de problèmes que nous groupons en problèmes de non coopération. Le manque de coopération a gagné beaucoup d'attention dans le contexte de la gestion de réseau ad-hoc et plusieurs techniques ont été proposées pour mitiger ce nouveau type de menace.

Il est important de rappeler que le choix des primitives cryptographiques employées par n'importe quel mécanisme de sécurité proposé pour l'environnement ad-hoc doit être pensé en considérant les conditions spécifiques dues à la nature des nœuds mobiles formant le réseau (la puissance de calcul et la capacité mémoire peuvent être considérées en tant que ressources rares). Les frais additionnels dus au trafic introduit par des mécanismes de sécurité doivent également être considérés avec un œil attentif. La bande passante et la disponibilité énergétique sont une autre ressource rare qui ne devrait pas être gaspillée par des mécanismes de sécurité sub-optimaux. D'autre part, le manque de coopération se traduit par des exigences de sécurité spécifiques et nouvelles pour les réseaux ad-hoc.

Plusieurs méthodes basées sur les systèmes cryptographiques, telles que la cryptographie symétrique ou à clé publique et des techniques de hachage, ont été proposées pour fournir des services de confidentialité, d'authentification, d'intégrité et de non répudiation pour les MANETs.

Les solutions à clé publique intègrent une autorité de certification (CA) centralisée ou distribuée. La majorité des mécanismes [Zhou99, Kong01, Yi03] proposés pour la gestion distribuée de clés sont basés sur la cryptographie à seuil [Shamir79].

Dans [Cheambe04], les auteurs proposent des technologies GSM/GPRS, permettant aux nœuds ad-hoc d'accéder à des services de CA. Une CA offline est utilisée dans [Capkun03] pour décider quels nœuds peuvent joindre le réseau, et pour assigner une identité unique à chacun. Chaque nœud tient une copie de la clef publique du CA, de sorte qu'il puisse valider les certificats des autres nœuds.

3.2. Les mécanismes de coopération

Les mécanismes de sécurité conventionnels sont considérés comme inappropriés pour les MANETs, ceci est dû aux techniques compliquées de gestion de clés qui requirent une puissance de calcul et une bande passante importantes.

La coopération est une caractéristique principale des systèmes décentralisés, et encore plus pour les systèmes ad hoc. Elle permet de compenser le manque d'une entité centrale. Cependant, la coopération pour réaliser une certaine fonctionnalité peut être gênée par le fait que les utilisateurs ont la pleine autorité sur leurs dispositifs et, comme avéré par l'expérience, essaient de maximiser les avantages qu'ils obtiennent du réseau.

Généralement, le comportement coopératif d'un dispositif a comme conséquence une augmentation de la consommation de ses ressources (CPU, espace mémoire, bande passante). Par exemple, dans le cas de l'expédition de données dans les MANETs, le nœud expéditeur doit encourir une utilisation additionnelle de l'énergie et de la bande passante pour la réception et la transmission des paquets.

Sachant que pour les dispositifs mobiles les ressources sont modestes, chacun de ces dispositifs devrait, de son point de vue, de préférence ne pas coopérer. Pour équilibrer ceci et réaliser un meilleur résultat global, il est primordial de concevoir des mécanismes pour encourager la coopération et décourager le comportement peu coopératif, qu'il soit passif ou malveillant.

3.3. Classification des mécanismes de coopération

Les mécanismes utilisés pour imposer la coopération sont classés en deux catégories (voir figure 3.1) : les mécanismes basés sur la réputation (*reputation-based*) et les mécanismes basés sur la rémunération (*credit-based*).

Dans les mécanismes basés sur la réputation (systèmes de réputation), la décision d'interagir avec un pair est basée sur sa réputation. L'architecture de ces systèmes peut être centralisée, décentralisée ou hybride. La réputation d'un nœud augmente lorsque ce dernier participe correctement dans le réseau, et diminue dans le cas contraire. Les mécanismes basés sur la réputation intègrent des mécanismes pour mesurer la réputation des autres nœuds du réseau. Ils incorporent également des techniques qui isolent les nœuds malveillants, c'est-à-dire, ceux qui montrent une valeur de réputation petite.

Les mécanismes basés sur la réputation peuvent être divisés en deux sous-classes. La première sous-classe inclut les modèles selon lesquels les nœuds utilisent seulement leurs observations personnelles (*first-hand information* ou information locale) de la réputation de leurs nœuds voisins pour prendre une décision de coopération, nous les appelons mécanismes locaux. La deuxième catégorie inclut les modèles selon lesquels les nœuds prennent en compte les observations des autres nœuds dans le réseau (informations globales ou *second-hand information*). Les nœuds dans ces modèles diffusent des informations de réputation. Si un certain nœud observe qu'un autre nœud ne se comporte pas correctement, il rapporte cette observation aux autres nœuds du réseau, nous les appelons mécanismes globaux.

Les mécanismes basés sur la rémunération (systèmes de paiement) traitent la coopération d'un nœud comme un service dont le fournisseur peut être rémunéré. Ils incorporent une forme de devise virtuelle pour l'achat (resp. paiement) des services fournis (resp. demandés) par les nœuds. Ces mécanismes requirent l'utilisation d'un module de sécurité (*tamper-resistant hardware*) ou une banque virtuelle, cette dernière joue le rôle d'une tierce partie de confiance.

En effet, il existe une troisième catégorie moins importante, celle des mécanismes basés sur la punition. Cette catégorie n'utilise pas des informations de réputation mais emploie une simple stratégie de punition qui devrait être adoptée par tous les nœuds du réseau.

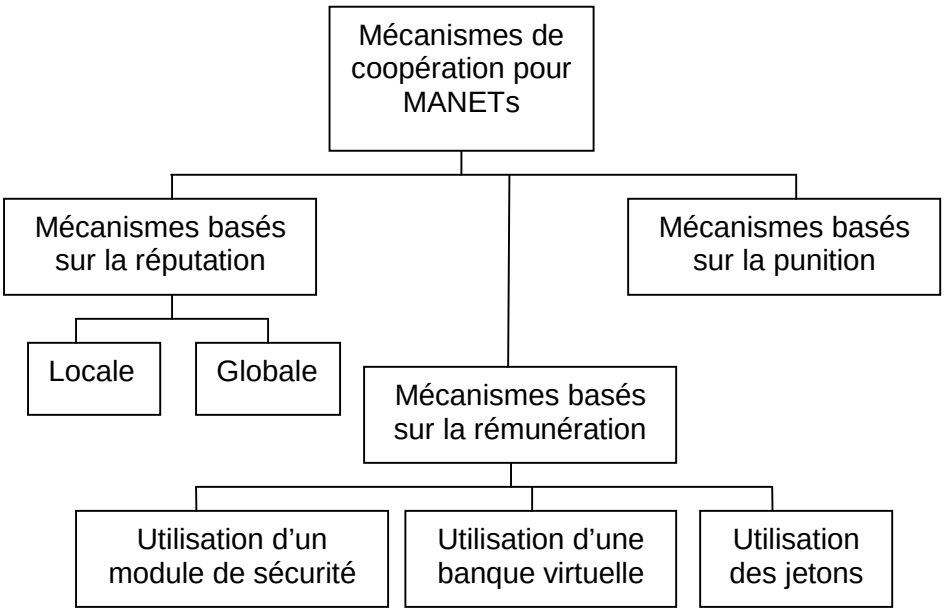


Figure 3.1. Taxonomie des mécanismes de coopération pour les MANETs

3.4. Les mécanismes basés sur la punition

En utilisant des idées de la théorie des jeux, des mécanismes de punition simples [Altman04, Srinivasan03] ont été conçus pour résoudre le problème de coopération dans les MANETs.

Dans [Altman04 et Srinivasan03], les auteurs voient le problème de coopération dans les MANETs comme un problème de jeux non coopératifs et essayent de le résoudre en consultant

les travaux antérieurs dans le domaine de la théorie des jeux. Les deux travaux [Altman04 et Srinivasan03] arrivent à la conclusion que la stratégie TFT[4] (Tit-For-Tat) est un bon algorithme qui peut être employé pour fournir des incitations pour des nœuds à coopérer.

Altman et son groupe dans [Altman04] ont défini une stratégie TFT spéciale, suivant laquelle tous les nœuds du réseau diminuent leurs propres probabilités d'expédition jusqu'à atteindre la probabilité minimale d'expédition parmi tous les nœuds. En d'autres termes, si à l'origine tous les nœuds expédiaient des paquets à un taux α, et si un nœud décide d'expédier à un taux $\beta < \alpha$, alors tous les autres nœuds réagiront en plaçant leurs probabilités d'expédition à β pour punir ce nœud.

Dans [Srinivasan03], les auteurs proposent un modèle théorique dans lequel l'information énergétique est prise en considération pour décrire l'interaction contradictoire entre les nœuds hétérogènes impliqués dans un jeu d'expédition. Dans ce jeu d'expédition, les nœuds qui appartiennent à un chemin reliant une source à une destination doivent relayer en collaboration les paquets de données à acheminer. Les auteurs étudient les propriétés d'une stratégie bien connue (generous-tit-for-tat, G-TFT) et démontrent que sous les contraintes d'énergie imposées aux nœuds, G-TFT encourage la coopération si chaque nœud du réseau se conforme à cette stratégie.

Le modèle présenté dans [Srinivasan03] fournit une description précise de la contrainte énergétique d'un nœud, qui est la raison principale d'un comportement égoïste, mais fournit seulement les directives à un niveau élevé pour la conception d'un mécanisme de coopération basé sur la stratégie G-TFT.

Le mécanisme TFT, bien qu'il soit simple, utilise un taux universel (dans ce cas un taux d'expédition) pour tous les nœuds au lieu d'utiliser un pour chaque nœud. Un taux d'expédition de 0 est toujours un équilibre, ce qui signifie que le mécanisme ne converge pas toujours à un point positif. Notons aussi que dans des scénarios réels, il existe des nœuds malveillants ou défectueux. Avec ces mécanismes, de tels nœuds ramèneront l'équilibre à 0 et rendront ainsi le réseau non fonctionnel. Nous pensons que cette catégorie de mécanismes n'est pas vraiment adaptée aux réseaux ad-hoc non coopératifs. Par conséquent, nous avons décidé de ne pas discuter cette catégorie de mécanismes dans la suite de notre travail.

3.5. Les mécanismes basés sur la réputation

Dans un environnement distribué, la réputation d'une entité est la perception globale sur le futur comportement de l'entité, elle est basée sur les expériences antérieures des autres entités avec cette entité. Le nœud qui se comporte correctement aura une réputation élevée ; un nœud se conduisant mal aura une petite réputation. Dans un système de réputation, les nœuds avec une réputation élevée seront récompensés en exécutant leurs demandes de service, alors que les demandes de service des nœuds avec une valeur de réputation petite seront rejetées. Afin de survivre (pour obtenir les services nécessaires), un nœud devrait essayer son meilleur pour offrir de bons services à d'autres nœuds.

[4] Coopérez la première fois et puis copiez le dernier mouvement de votre adversaire pendant toutes les périodes suivantes.

3.5.1. CONFIDANT

Buchegger et Le Boudec ont proposé un système appelé CONFIDANT (Cooperation Of Nodes Fairness In Dynamic Ad-hoc NeTworks) [Buchegger02], conçu comme une extension pour le protocole de routage réactif DSR [Hu03].

CONFIDANT consiste en quatre modules, chacun accompli une tâche bien spécifique. La figure 3.2 présente l'architecture modulaire de CONFIDANT, conçue pour faire face à différents types d'attaques liées à l'acheminement et à l'expédition de données.

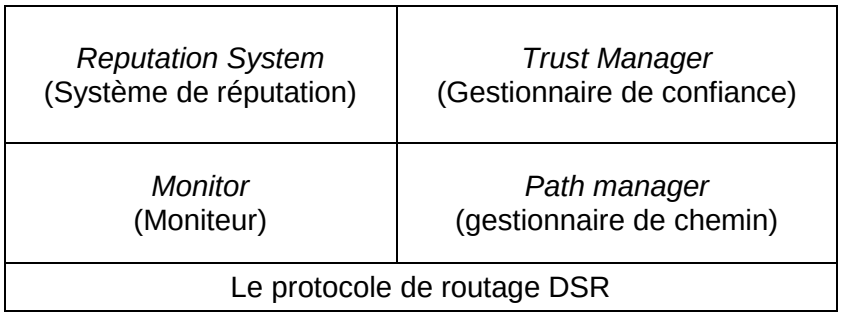


Figure 3.2. L'architecture de CONFIDANT [Buchegger02]

3.5.1.2. Le moniteur (The monitor, Neighborhood Watch)

Le but du moniteur est de recueillir des informations locales sur le comportement des nœuds dans le réseau (en termes d'acheminement et d'expédition de données). Ceci est réalisé en observant le comportement du nœud pour le classifier ensuite en tant que normal ou malveillant.

Le moniteur peut détecter par observation un comportement malveillant. Un exemple de comportement malveillant est la suppression des paquets, la modification illégitime d'en-tête de paquet et la fabrication des messages d'erreur. De nouveaux types de comportement malveillant observable peuvent être ajoutés au moniteur une fois que leurs signatures sont connues.

L'information obtenue par le nœud *i* à propos du nœud *j* par une expérience directe est appelée information de première main (*first-hand information*) ou information locale ($F_{i,j}$), et elle est utilisée comme entrée pour le composant système de réputation de CONFIDANT.

3.5.1.3. Le système de réputation (*The reputation system*)

Un nœud *i* maintient des estimations concernant chaque autre nœud *j* qui l'intéresse. *L'estimation de réputation* (ou simplement réputation) représente l'opinion constitué par le nœud *i* au sujet du comportement du nœud *j* en tant que acteur dans le système de base, c.-à-d. si le nœud *j* participe correctement dans le protocole de routage. L'estimation de réputation est représentée par une structure de données $R_{i,j}$.

L'utilisation des informations de seconde main ou informations globales (*second-hand information*) permet à un nœud de découvrir des nœuds malveillants avant de faire une mauvaise expérience avec eux. En outre, dans les MANETs un nœud peut ne pas rencontrer tous les nœuds impliqués dans l'acheminement de ses données, mais avec des informations globales il peut décider quelle route utiliser.

Le système de réputation gère une table contenant les nœuds et leurs réputations. La réputation d'un nœud est mise à jour si deux conditions sont vérifiées (i) il y a suffisamment de preuve d'un comportement malveillant, et (ii) le nombre d'occurrence du comportement malveillant dépasse un certain seuil. Dans ce cas, la réputation est mise à jour suivant une fonction qui assigne des pondérations suivant le type de détection, à savoir le plus grand poids pour sa propre expérience, un poids plus petit pour des observations dans le voisinage, et un poids encore plus petit pour une expérience rapportée. La raison de cet arrangement est qu'un nœud fait plus confiance à ses propres expériences et observations qu'à celles des autres nœuds.

Une fois que le poids a été déterminé, l'entrée du nœud malveillant est changée en conséquence. Si la réputation du nœud dans la table est inférieure à un seuil de tolérance, le gestionnaire de chemin est invoqué pour prendre les mesures nécessaires. Considérant que le comportement malveillant sera idéalement l'exception plutôt que la norme, le système de réputation est bâti sur les expériences négatives plutôt que sur les expériences positives.

3.5.1.4. Le gestionnaire de confiance (*The trust manager*)

La tâche du gestionnaire de confiance est de décider sur le moment de faire confiance à l'information globale reçu et à gérer la confiance attribuée aux autres nœuds. Le but est de minimiser le risque de fausses informations tout en se servant toujours de l'information globale reçue.

L'estimation de confiance (ou simplement confiance) représente l'estimation de l'honnêteté que le nœud *i* attribue au nœud *j* en tant qu'acteur dans le système de réputation (c.-à-d. si l'information locale publiée par le nœud *j* est susceptible d'être vraie). L'estimation de confiance est représentée par une structure de données $T_{i,j}$.

Des messages d'ALARME sont envoyés par le gestionnaire de confiance du nœud pour avertir les autres nœuds des nœuds malveillants. Les messages d'ALARME sortants sont produits par le nœud lui-même après avoir éprouvé, observé, ou reçu un rapport du comportement malveillant. Les destinataires de ces messages d'ALARME sont de prétendus amis, qui sont administrés dans une liste d'amis.

Les messages d'ALARME entrants proviennent des amis ou des autres nœuds. Ainsi la source du message d'ALARME doit être examinée avant de déclencher une réaction. À cet effet, il y a un filtrage des messages d'ALARME entrants selon le niveau de confiance du nœud reporteur.

Le gestionnaire de confiance comprend les composants suivants :

- Une table d'alarme contenant des informations sur les messages d'alarmes reçues ;
- Une table de confiance pour gérer la confiance attribuée aux nœuds ;
- Une liste d'amis.

3.5.1.5. Le gestionnaire de chemin (*The path manager*)

Une fois qu'un nœud *i* classifie un autre nœud *j* comme malveillant, *i* isole *j* des communications en n'employant pas *j* pour l'acheminement et l'expédition de données et en ne permettant pas à *j* d'employer *i*. Cette isolation vise trois buts. Le premier est de réduire l'effet du nœud malveillant en le privant de participer au réseau. Le second est de servir d'incitation à bien

se comporter pour ne pas être renié. Le troisième but est d'obtenir un meilleur service en n'employant pas des nœuds malveillants sur les routes utilisées.

Le gestionnaire de chemin exécute les fonctions suivantes :

- Réévaluation des routes selon une métrique de sécurité (par exemple la réputation des nœuds dans la route) ;
- Suppression des routes contenant des nœuds malveillants ;
- Action à la suite de la réception d'une requête de route (RREQ) de la part d'un nœud malveillant (par exemple ignorer la requête) ;
- Action à la suite de la réception d'un RREQ contenant un nœud malveillant dans la route source (par exemple ignorer la requête, alerter la source).

3.5.1.6. Le fonctionnement du protocole

Chaque nœud surveille son voisinage à un saut, et il met à jour les réputations des nœuds en conséquence. Précisément un nœud peut détecter le comportement égoïste du prochain nœud dans la route soit directement en utilisant l'écoute promiscuous des transmissions du prochain nœud (reconnaissance passive, *passive acknowledgment*), ou indirectement en observent le comportement du protocole de routage.

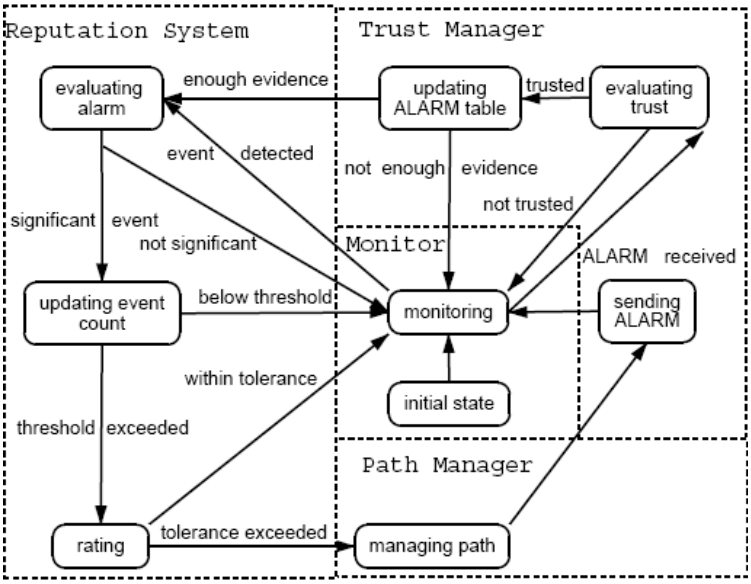


Figure 3.3. Le fonctionnement de CONFIDANT [Buchegger02]

Si un événement soupçonneux est détecté, l'information est fournie au système de réputation. Si l'événement est significatif pour le nœud, et s'il s'est produit un nombre de fois supérieur à un seuil[5] prédéfini, qui est assez grand pour distinguer le comportement malveillant délibéré des coïncidences simples telles que des collisions, le système de réputation met à jour la réputation du nœud qui a causé l'événement. Si la réputation du nœud s'avère être intolérable, l'information est transmise au gestionnaire de chemin, qui supprime du cache toutes les routes contenant le

[5] Le seuil en question peut être défini pour différents types de nœuds selon leurs conditions de sécurité.

nœud détecté. Le nœud continu à surveiller le voisinage, et un message d'ALARME est envoyé à la source de la route (figure 3.3).

Lorsque le moniteur d'un nœud reçoit un message d'ALARME, il le transmet au gestionnaire de confiance, où la source de message est évaluée. Si le nœud fait, au moins, partiellement confiance dans la source, la table contenant les alarmes est mise à jour. S'il y a suffisamment de preuve, que le nœud rapporté dans le message d'ALARME est malveillant, l'information est envoyée au système de réputation où elle est de nouveau évaluée pour la signification, le nombre d'occurrences, et la réputation accumulée du nœud. Suffisamment de preuve signifie que la source du message d'ALARME est entièrement de confiance, ou que plusieurs nœuds partiellement de confiance ont rapporté le même message d'ALARME et que leurs degrés de confiance assignée s'ajoute jusqu'à la valeur d'un nœud entièrement de confiance ou davantage.

La première version de CONFIDANT était vulnérable au phénomène de propagation de rumeur [Buchegger03]. Dans une amélioration récente, ce problème a été adressé à travers un modèle bayesian [Buchegger03, Buchegger04] qui classifient et excluent les menteurs. Dans cette version on utilise les deux types d'informations (c'est-à-dire, les expériences négatives et les expériences positives) pour estimer la réputation d'un nœud. L'hypothèse derrière cette amélioration est que les expériences positives et négatives rassemblées par un nœud devraient indiquer la même sorte d'information pour un nœud comme ce qui est recueilli par les autres nœuds. Le facteur $R_{i,j}$ est exprimé comme une fonction de α et β, où α (resp. β) est le nombre de comportement malveillant (resp. légitime). Ces deux nombres sont mis à jour à en se basant sur les expériences récentes. Une recommandation, à propos d'un nœud, est acceptée si (i) le nœud reporteur est entièrement de confiance ou (ii) si l'information rapportée est compatible (au sens du modèle bayesian). Cette technique réduit l'impact des accusations fausses. Le même modèle bayesian est utilisé pour gérer les estimations de confiance.

3.5.1.7. Discussion

La première version de CONFIDANT [Buchegger02] a été largement critiquée, surtout l'hypothèse faite sur l'existence d'une liste d'amis, où les auteurs supposent une relation de confiance préalable entre quelques nœuds, cette hypothèse a été abandonnée par la suite [Buchegger04].

La deuxième version de CONFIDANT est beaucoup plus robuste par rapport à la première version. L'approche bayesienne adoptée permet d'utiliser les deux types d'informations (positive et négative) tout en minimisant le risque de fausses informations. Par contre cette approche est binaire (2-niveaux), c'est-à-dire elle classe les nœuds en deux classes : fiable et non fiable ; un système à n-niveaux est considéré plus flexible et plus fiable [Quercia06].

Dans CONFIDANT, l'échange des informations de réputation entre les nœuds est crucial, alors que, CONFIDANT ne fournit aucune forme d'incitation pour l'échange des informations de réputation. Finalement, la plupart des valeurs seuils (d, r, t) restes empiriques.

3.5.2. CORE

Dans [Michiardi02] les auteurs proposent un mécanisme appelé CORE (A COllaborative REputation mechanism to enforce node cooperation in mobile ad-hoc networks) pour imposer la coopération entre les nœuds. CORE se base sur une technique de surveillance distribuée. Ce mécanisme de coopération n'empêche pas un nœud de nier la coopération ou de dévier d'un comportement légitime mais s'assure que les entités se conduisant mal soient punies en leur refusant graduellement les services de communication.

3.5.2.1. La réputation dans CORE

CORE fait la distinction entre trois types de réputation, la réputation subjective, indirecte et fonctionnelle.

a) La réputation subjective (*Subjective Reputation*)

C'est la réputation calculée directement à partir des informations locales. Plus d'importance est donnée aux observations passées. La raison pour laquelle plus de poids est donné aux observations passées est qu'un comportement malveillant sporadique dans des observations récentes devrait avoir une influence minimale sur l'évaluation de la valeur finale de réputation. En conséquence, il est possible d'éviter des fausses détections dues aux coupures de lien.

b) La réputation indirecte (*Indirect Reputation*)

La réputation subjective est calculée en considérant les interactions directes entre le nœud en question et son voisinage. La réputation indirecte permet de prendre en compte les informations fournies par les autres nœuds. L'information collectée à travers la réputation indirecte peut prendre seulement des valeurs positives : des attaques de déni de service basées sur la diffusion malveillante des estimations négatives pour des nœuds légitimes sont ainsi empêchées.

c) La réputation fonctionnelle (*Functional Reputation)*

Elle fait référence à la réputation subjective et indirecte calculées par rapport à une fonction *f*. Avec l'introduction de ce dernier type de réputation dans le modèle, nous aurons la possibilité de calculer une valeur de réputation globale qui prend en compte différents critères d'observation/évaluation. Comme exemple, un nœud *i* peut évaluer la réputation subjective d'un nœud *j* par rapport à la fonction de l'expédition de paquets et évaluer la réputation subjective par rapport à la fonction de routage, ensuite combiner les deux en employant différents poids pour obtenir une valeur de réputation globale du nœud *j*.

3.5.2.2. L'architecture de CORE

CORE se compose de trois éléments : l'entité réseau, la table de réputation et le mécanisme de surveillance.

a) L'entité réseau (*Network entity*)

L'entité réseau correspond au nœud mobile. Chaque entité est munie d'une table de réputation (RT) est d'un mécanisme de surveillance (WD). RT et WD constituent la base du mécanisme de collaboration CORE. Ces deux composants permettent à chaque entité d'observer et de classifier l'entité qui devient impliquée dans un processus de requête/réponse. La

classification des entités basée sur leurs comportements est alors employée pour imposer un lien fort entre le comportement coopératif d'une entité et l'utilisation des ressources communes rendues disponibles par toutes les autres entités du réseau.

Les auteurs appellent *demandeur* un nœud demandant l'exécution d'une fonction *f* et *fournisseur* n'importe quel nœud censé exécuter correctement *f*. Un nœud de confiance est un nœud avec une valeur de réputation positive.

b) La table de réputation (*Reputation table, RT*)

C'est une structure de données, stockée sur chaque nœud. Chaque ligne de la table inclut les données de réputation concernant un nœud. Chaque ligne se compose de quatre entrées : un identifiant unique de l'entité, une collection d'observations subjectives récentes faites sur le comportement de cette entité, une liste des valeurs de réputation indirectes récentes fournies par d'autres entités et la valeur de la réputation évaluée pour une fonction prédéfinie.

Chaque nœud a une RT pour chaque fonction qui doit être surveillée. Finalement une RT globale est employée pour combiner les différentes valeurs de la réputation calculées pour différentes fonctions.

c) Le mécanisme de surveillance (*The Watchdog mechanism, WD*)

Le mécanisme de surveillance permet de détecter un comportement malveillant. Chaque fois qu'un nœud (l'entité surveillante) doit surveiller l'exécution correcte d'une fonction mise en œuvre par un nœud voisin (l'entité observée), il déclenche un WD spécifique à cette fonction (*f*). Le WD stocke temporairement le résultat attendu *Er(f)* dans un tampon et vérifie s'il correspond bien au résultat observé *Or(f)*. Si la fonction surveillée est exécutée correctement, le WD supprime du tampon l'entrée qui correspond au couple *Er(f)/Or(f)* et entre dans un état d'attente. D'autre part, si la fonction n'est pas correctement exécutée ou si le couple demeure dans le tampon un temps supérieur à un seuil prédéfini, une valeur négative est ajoutée à l'estimation de réputation de l'entité observée.

3.5.2.3. Description du protocole

CORE implique deux types d'entité, un demandeur et un ou plusieurs fournisseurs. La nature du protocole et les mécanismes sur lesquels il se fonde s'assurent que si un fournisseur refuse de coopérer (c.-à-d. la demande n'est pas satisfaite), alors CORE réagit en diminuant la réputation du fournisseur, menant à son exclusion si le comportement non coopératif persiste.

a) Mises à jour et distribution de la RT

Les RTs sont mises à jour dans deux cas : durant la phase de requête et durant la phase de réponse qui correspond au résultat de l'exécution de *f*.

Dans le premier cas, seulement la réputation subjective est mise à jour. Si les résultats du WD prouvent que le fournisseur n'a pas coopéré, un facteur négatif sera assigné à l'observation et par conséquent la réputation liée à l'entité malveillante diminuera. Si aucun comportement malveillant n'est détecté, les RTs ne sont pas mises à jour.

Dans le deuxième cas, seulement la réputation indirecte est mise à jour. Les auteurs supposent que le message de réponse contient la liste des entités qui se sont correctement comportées : la réputation indirecte sera positive et par conséquent la réputation liée aux entités qui ont coopéré augmentera.

Les valeurs de réputation calculées pour chaque entrée de la RT ne sont pas constantes : si la valeur de réputation est positive alors elle est décrémentée au cours du temps. La raison pour laquelle on décrémente les valeurs de réputation positives vient d'une attaque possible à CORE : si un nœud entre dans un état oisif la plupart du temps sauf lorsqu'il doit communiquer, sa réputation doit être diminuée, même si pendant le temps actif il est coopératif. La réputation est décrémentée jusqu'à ce qu'elle atteigne une valeur nulle, qui correspond à un comportement neutre (le nœud n'est ni coopératif ni malveillant).

b) Renforcement de la coopération

La réputation est directement liée au comportement coopératif d'une entité : si la valeur de réputation est négative alors l'entité est classifiée comme entité malveillante, tandis que si la valeur de réputation est positive alors l'entité est considérée comme entité de confiance. L'exécution d'une fonction demandée par n'importe quel demandeur est conditionnée par la valeur correspondante de réputation stockée dans la RT du fournisseur : lorsque cette valeur est négative, le fournisseur niera l'exécution de l'opération demandée.

3.5.2.4. Discussion

La formule employée pour évaluer la réputation d'un nœud évite les fausses détections en employant un facteur de pondération qui donne plus de pertinence aux observations passées. Ceci est vrai seulement sous l'hypothèse que le comportement d'un nœud est constant dans le temps. Dans le cas contraire, une telle approche est vulnérable à une attaque où un nœud peut accumuler une bonne réputation avant de se conduire mal.

L'information collectée à travers la réputation indirecte peut prendre seulement des valeurs positives : des attaques de déni de service basées sur la diffusion malveillante des estimations négatives pour des nœuds légitimes sont ainsi empêchées. Cependant, deux nœuds ou plus peuvent s'entendre (Attaque par collusion, c.-à-d., envoyez des messages d'estimation positifs) afin d'augmenter leurs réputation. Pour empêcher de tels phénomènes, CORE assure implicitement une certaine protection, puisque la réputation subjective a plus d'impact (c.-à-d., poids) que la réputation indirecte. Cependant, cette protection reste limitée, puisque un nombre important de faux messages positifs permet de modifier considérablement la réputation d'un nœud. D'ailleurs, même si l'information diffusée est correcte, on ne peut pas distinguer entre un nœud malveillant et un nouveau nœud qui vient juste de joindre le réseau.

CORE ne fait pas la distinction entre la notion de réputation (liée à l'activité d'un nœud dans le réseau) et la notion de confiance (liée à l'activité d'un nœud dans le système de réputation).

Finalement, un mécanisme de deuxième chance n'est pas considéré, comme dans OCEAN [Bansal03]. Par conséquent, un nœud qui a subit un défaut de fonctionnement temporel ne peut pas reconstruire sa réputation, lorsqu'il regagne le réseau, à moins qu'il y ait un nombre suffisant de nouveaux nœuds arrivant dans le réseau qui n'ont aucune expérience antérieure avec lui.

3.5.3. SORI

Dans [Qi04], les auteurs proposent un mécanisme appelé SORI (A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks), qui se focalise, exclusivement, sur la fonctionnalité d'expédition de données.

3.5.3.1. L'architecture de SORI

SORI comprend trois composants : surveillance de voisinage (*neighbor monitoring*), propagation de la réputation (*reputation propagation*) et punition (*punishment*).

a) Surveillance du voisinage

La surveillance du voisinage est utilisée pour collecter des informations sur le comportement des nœuds voisins en termes d'expédition de données. Le mode promiscuous est assumé, et un nœud est capable d'écouter les transmissions de ses voisins et de maintenir une liste des nœuds voisins (noté NNL_N). En plus chaque nœud *N* garde deux nombres, pour chacun de ses voisins (noté *X*), comme suit :

$RF_N(X)$ (*Request-for-Forwarding*) : le nombre total de paquets que le nœud *N* a envoyé au nœud *X* pour une éventuelle expédition ;

$HF_N(X)$ (*Has-Forwarded*) : le nombre total de paquets expédiés par *X* et notés par *N*.

Les deux nombres sont mis à jour par les règles suivantes : lorsque le nœud *N* envoie un paquet au nœud *X* pour une éventuelle expédition, le compteur $RF_N(X)$ est incrémenté par 1. Ensuite *N* écoute le canal et contrôle si le nœud *X* expédie le paquet comme prévu. Si *N* détecte que *X* a expédié le paquet avant un timeout, le compteur $HF_N(X)$ est incrémenté par 1.

Ayant $RF_N(X)$ et $HF_N(X)$, le nœud *N* peut créer un enregistrement appelé *local evaluation record* (noté $LER_N(X)$) pour le nœud voisin *X*. L'enregistrement $LER_N(X)$ consiste en deux entrées, GN(X) et CN(X), où $GN(X) = RF_N(X)/HF_N(X)$ et CN(X) est une métrique appelée confiance, employée pour décrire à quel point le nœud *N* est confiant dans son jugement sur la réputation du nœud *X*. Les auteurs proposent que $CN(X) = RF_N(X)$. C.-à-d. plus les paquets sont transmis à *X* pour l'expédition, meilleure sera l'estimation de la réputation de *X*.

b) Propagation de la réputation

SORI utilise des informations locales et globales. Les nœuds échangent des informations de réputation avec leurs voisins. De cette manière, un nœud non coopératif sera puni par tous ses voisins (qui partagent des informations de réputation sur son comportement), au lieu d'être puni seulement par les nœuds affectés.

La propagation de la réputation fonctionne comme suit : chaque nœud *N* périodiquement met à jour son $LER_N(X)$ pour chaque nœud voisin *X*, en se basant sur les changements de $RF_N(X)$ et $HF_N(X)$. Si un changement remarquable dans GN(X) est noté, il sera diffusé aux nœuds voisins.

Le nœud *N* emploi son $LER_N(X)$ et $LER_i(X)$ (où *i* est dans le NNL_N) pour calculer son *overall evaluation record* de *X* (noté $OER_N(X)$) comme suit :

$$OER_N(X) = \frac{\sum_{i \in NNL_N \cup \{N\}, i \neq X} \lambda_N(i) \cdot C_i(X) \cdot G_i(X)}{\sum_{k \in NNL_N \cup \{N\}, k \neq X} \lambda_N(k) \cdot C_k(X)}$$

Où $\lambda_N(i)$ est la crédibilité du nœud *i* dans la vision du nœud *N*. Dans [Qi04] les auteurs proposent que $\lambda_N(i) = G_N(i)$.

c) Punition

Si $OER_N(X)$ est inférieur à un seuil prédéfini, le nœud *N* punit le nœud *X* en supprimant d'une manière probabiliste les paquets dont l'origine est *X*.

La probabilité de suppression est $p = \begin{cases} q - \delta & \text{si } q > \delta \\ 0 & \text{sinon} \end{cases}$

Où $q = 1 - OER_N(X)$ et $0 < \delta < 1$ est la marge introduite pour prendre en compte la suppression des paquets occasionnée par des phénomènes tels que les collisions et non pas en raison d'un comportement malveillant. Tel que mentionné dans [Qi04], ce mécanisme est conçu pour que les nœuds se traitent un peu plus généreusement.

Dans [Qi04], on propose un mécanisme de sécurité complémentaire pour faire face aux attaques suivantes : (1) personnification de l'identité d'un nœud adjacent, avec une bonne réputation, afin d'envoyer plus de paquets, et, (2) personnification de l'identité d'un nœud éloigné, avec une bonne réputation, pour diffuser de fausses informations afin d'amplifier sa réputation. Ce mécanisme est basé sur une technique de hachage en chaîne (*one-way hash chain*) et l'utilisation des MACs (*message authentication codes*).

3.5.3.2. Discussion

SORI ne fait pas la distinction entre la notion de réputation (liée à l'activité d'un nœud dans le réseau) et la notion de confiance (liée à l'activité d'un nœud dans le système de réputation). De plus, supposé que $\lambda_N(i) = G_N(i)$ rend le système vulnérable, du moment qu'un nœud qui participe activement dans le réseau peut influencer considérablement la réputation des autres nœuds.

Le mécanisme de sécurité complémentaire introduit un surplus de calcul et de communication dans le réseau, alors qu'il reste vulnérable à plusieurs attaques, telle qu'une attaque par des identités multiple "Sybil attaque", où un nœud génère dès le départ plusieurs identités. De plus, la diffusion des identités des nœuds dans tout le réseau pose un problème de passage à l'échelle.

SORI est vulnérable à une attaque par collusion. Finalement, la plupart des valeurs seuils restes empiriques (δ par exemple).

3.5.4. Le mécanisme de Liu et Issarny

Liu et Issarny [Liu04] présentent un modèle de réputation qui incorpore deux facteurs, le temps et le contexte ainsi que des mécanismes pour gérer la formation, l'évolution et la diffusion de la réputation.

Le modèle n'est pas limité seulement aux fonctionnalités niveau couche réseau, mais il vise divers types de services, tels que les services Web. Le modèle fournit des mesures de défense pour faire face aux attaques suivantes : (1) L'inactivité : un nœud refuse de partager des informations de réputation avec les autres nœuds, (2) La diffamation : diffusion de fausses informations de réputation pour diffamer un nœud victime, et (3) La collusion.

3.5.4.1. Le temps et le contexte

Dans ce modèle, chaque nœud maintien deux valeurs de réputation séparées pour les autres nœuds qui fournissent un service ou une recommandation. Ces deux valeurs sont la réputation de service (notée SRep) et la réputation de recommandation (notée RRep).

La réputation d'un nœud est construite au cours du temps. Pour prendre en compte le facteur temps dans le modèle, un facteur de pondération est introduit. Plus ce facteur est grand, plus d'importance sera accordée au comportement récent du nœud.

Le modèle considère aussi le facteur contexte. Un nœud estime la réputation d'un service C fourni par un autre nœud, en se basant sur ses propres observations et les observations des autres nœuds qui ont utilisé le service C. Dans certains cas il n'y a pas suffisamment d'information de réputation dans le contexte C, et suffisamment d'information de réputation dans un contexte relative à un service C'. Les auteurs proposent un concept, emprunté des arbres ontologiques et de DAML (DARPA Agent Markup Language), pour mesurer la distance entre deux services, et proposent une formule qui permet, sachant la réputation d'un service C' et la distance entre C et C', de calculer la réputation du service C.

3.5.4.2. L'architecture du mécanisme

Le modèle utilise trois composants fonctionnels : un gestionnaire d'expérience (*experience manager*), un gestionnaire de recommandation (*recommendation manager*) et un gestionnaire de réputation (*reputation manager*).

a) Le gestionnaire d'expérience

Le gestionnaire d'expérience enregistre les expériences directes avec un service. Après chaque interaction, les nœuds peuvent donner des scores de satisfaction. L'enregistrement comprend la catégorie du service (contexte C), le temps d'occurrence de l'expérience (t), et un score ($SExp_a(o,C)^t$). Ces scores sont habituellement subjectifs, et dépendent de facteurs multiples.

b) Le gestionnaire de recommandation

Le gestionnaire de recommandation stocke les recommandations diffusées par les autres nœuds, échange des informations de réputation avec les autres nœuds, et gère une table de RReps. Des informations de réputation sont échangées périodiquement dans tout le réseau. Chaque nœud communique seulement avec les nœuds qui ont une valeur de RRep élevée. Avec chaque nouvelle interaction (expérience) avec un nœud *X*, le nœud met à jour la RRep des nœuds qui ont diffusé des recommandations à propos du nœud *X*. Dans le modèle, la réputation est considérée subjective. Ainsi, des petites déviations entre les expériences obtenues par un nœud *X* et la recommandation qui a été faite pour le même service par un autre nœud *Y* est acceptée. Lorsque des déviations considérables se produisent, *Y* est classé en tant que non fiable. Si un nœud ne recommande jamais, son RRep sera classé comme " ignoré", et les autres hésiteront à échanger des informations de réputation avec lui. Ainsi, pour un nœud, il y a seulement une solution pour maintenir son RRep à un niveau décent, c'est de recommander activement et honnêtement.

c) Le gestionnaire de réputation

Le gestionnaire de réputation prend des entrées des autres composants pour calculer le SRep d'un nœud. Il assigne un poids important à sa propre expérience et un poids moins important aux recommandations rassemblées. Puisque, un nœud fait plus confiance dans ses propres expériences, que dans celles des autres.

Considérons un nœud *a* qui a des recommandations concernant un nœud *o*, d'un groupe de nœuds P ; les nœuds considérés non fiable ont été exclus de P. les auteurs proposent la formule suivante pour le calcul de SRep.

$$SRep_a(o)^t = \alpha * SExp_a(o)^t + (1-\alpha) * \frac{\sum_{p \in P}(RRep_a(p) * Rec_p(o))}{\sum_{p \in P} RRep_a(p)}$$

Où α est un paramètre qui reflète le degré de confiance attribué par un nœud en ses propres expériences, généralement α > 0.5.

3.5.4.3. Discussion

Le modèle proposé est générique, donc il peut être adapté à différents besoins. Les auteurs ne discutent pas un cas particulier. Les simulations effectuées démontrent que le système fait face aux attaques mentionnées auparavant.

Plus d'importance est accordée au comportement récent du nœud, ce qui rend le mécanisme vulnérable aux erreurs de mesures.

Le problème de passage à l'échelle (*scalability*) se pose, du moment qu'un nœud contacte tous les autres nœuds avec une RRep élevée. Le problème de changement de l'identité n'a pas été considéré et aucune hypothèse n'a été faite dans ce sens.

Le contexte a été incorporé dans le modèle, mais aucune analyse n'a été faite. Les auteurs proposent de mesurer la distance entre deux services en utilisant les arbres ontologiques et le langage DAML, ceci peut servir dans le cas des services niveau application (notamment les services Web). Les auteurs ne discutent pas les services de niveau inférieur (ex. le routage et l'expédition). Finalement, un mécanisme de deuxième chance n'est pas considéré, et la plupart des valeurs seuils restent empiriques.

3.5.5. OCEAN

Le système proposé dans [Bansal03], introduit une couche intermédiaire qui réside entre la couche MAC et la couche réseau. Cette couche aide les nœuds à prendre des décisions intelligentes lors de l'expédition de données. Elle est conçue pour le protocole DSR, mais elle peut être adaptée à d'autres algorithmes de routage.

3.5.5.1. Le comportement trompeur et le comportement égoïste

OCEAN (Observation-based Cooperation Enforcement in Ad-hoc Networks) considère deux types de comportement anormal : le comportement trompeur où un nœud répond correctement aux requêtes de route, mais il refuse d'expédier les paquets de données. Le deuxième comportement considéré est le comportement égoïste où un nœud ne répond même pas aux

requêtes de route mais peut néanmoins envoyer son propre trafic par le réseau, préservant injustement ses ressources tout en exploitant celles des autres.

3.5.5.2. L'architecture de OCEAN

OCEAN comprend cinq modules : le moniteur (*NeighborWatch*), gestionnaire de routes (*RouteRanker*), routage basé sur la réputation (*Rank-Based Routing*), gestionnaire du trafic malicieux (*Malicious Traffic Rejection*) et le mécanisme de deuxième chance (*Second Chance Mechanism*).

a) NeighborWatch

A l'expédition d'un paquet, le moniteur stocke le checksum du paquet dans un tampon, et surveille le canal sans fil après avoir envoyé le paquet à son voisin. S'il n'entend pas la tentative du nœud voisin pour expédier le paquet après un timeout (par défaut 1 ms), NeighborWatch enregistre un événement négatif pour ce voisin, et supprime du tampon le checksum. D'autre part, si le moniteur entend une tentative d'expédition par le nœud voisin, il compare le checksum stocké avec celui du paquet, dans le cas où les deux sont égaux, il enregistre un événement positif et supprime le checksum, sinon le paquet est considéré non expédié. Ces événements sont communiqués au RouteRanker, qui maintient les estimations des nœuds voisins.

Le module NeighborWatch n'est pas un service garanti. Il souffre de toutes les erreurs potentielles de WatchDog [Marti00], y compris par exemple, le fait d'observer un voisin expédier un paquet ne garantit pas que le paquet soit reçu avec succès par le prochain nœud de la route.

b) RouteRanker

Chaque nœud maintient des estimations pour chacun de ses voisins. L'estimation est initialisée à une valeur neutre est incrémentée et décrémentée après réception des événements, respectivement, positifs et négatifs du composant NeighborWatch. En raison des études empiriques, la valeur absolue d'un décrément est choisie pour être plus grande que la valeur d'un incrément. Lorsque l'estimation d'un nœud descend au-dessous d'un seuil (*faulty threshold*) le nœud est ajouté à la liste des nœuds malveillants (*faulty list*). Un chemin est évalué bon ou mauvais, en regardent si le prochain nœud dans la route appartient à cette liste ou non.

c) Rank-Based Routing

La liste des nœuds malveillants (*faulty list*) est jointe (comme un champ appelé *avoid-list*) au RREQ du protocole de DSR afin d'être inondée. Ce champ permet de spécifier les nœuds que l'émetteur du RREQ souhaite éviter dans ses futures routes. A la rediffusion d'un RREQ, un nœud ajoute sa *faulty list* au champ *avoid-list.* Si l'intersection de la liste *avoid-list* et le chemin source dans le RREQ est vide, le nœud récepteur du RREQ traite cette requête (par la rediffusion du RREQ ou l'envoi d'un RREP) sinon il la supprime. De même, un RREP est traité, si le chemin dans le RREP ne contient pas un nœud dans la liste *faulty list*. Autrement, le RREP est supprimé.

d) Malicious Traffic Rejection

Ce module rejette le trafic des nœuds considérés trompeurs. La politique employée est de rejeter tout le trafic d'un nœud trompeur de sorte qu'un nœud ne puisse pas transmettre par relais son propre trafic sous l'apparence de l'expédition au nom de quelqu'un d'autre.

e) Second Chance Mechanism

Le mécanisme de deuxième chance est prévu pour permettre à des nœuds précédemment considérés trompeurs de redevenir normaux. Sans ce mécanisme, une fois un nœud est ajouté à la liste *faulty list*, tous les futurs chemins passant par lui seront évités, ne donnant à ce nœud aucune occasion de démontrer sa "qualité". Ceci peut être injuste du moment que le fonctionnement du NeighborWatch n'est pas garanti. Un nœud a pu simplement avoir éprouvé des échecs passagers de lien, ou il a pu avoir dû redémarrer son interface réseau. Pour cela, après un temps prédéfini (*faulty timeout*), un nœud trompeur est enlevé de la liste *faulty list*. Quoique le nœud soit enlevé de la liste, son estimation n'est pas remise à neutre, de sorte qu'il puisse être rapidement ajouté à la liste dans le cas où son comportement malveillant persiste.

3.5.5.3. Le comportement égoïste

OCEAN emploie une politique différente pour traiter les nœuds qui ne participent pas au protocole de routage (la phase de découverte de routes). Cette politique, basée sur le concept de monnaie virtuelle n'exige ni un dispositif de sécurité ni un serveur central.

Chaque nœud mesure le comportement de ses voisins lors de ses interactions directes avec eux. Les nœuds mesures "l'équilibre d'expédition" avec leurs voisins en maintenant un compteur, appelé le nombre de jetons, par nœud. Le compteur augmente lorsque le nœud sollicite un autre nœud pour expédier un paquet et diminue avec une demande entrante de ce dernier. Supposons qu'un nœud B n'a pas participé à l'établissement du chemin avec un nœud source A. Si B demande de A d'expédier ses paquets, alors, A punira B et rejettera ses demandes, tant que le nombre de jetons de B est bas.

Cette politique est considérée injuste pour les nœuds appartenant au périmètre (extrémité) du MANET, puisque ils ne sont pas fréquemment sollicités pour expédier les messages des autres. La pénalisation de ces nœuds pourrait faire rétrécir le réseau. Pour surmonter de tels phénomènes, OCEAN introduit un paramètre de taux d'accumulation de jetons (*chip accumulation rate* CAR), qui exprime le taux auquel le nombre de jetons dans le réseau est augmenté par unité de temps. Ainsi, l'expédition des paquets envoyés par des nœuds périphériques est possible, même à un taux réduit.

3.5.5.4. Discussion

La valeur du seuil *faulty threshold* reflète la vitesse et l'exactitude de la détection du comportement malveillant. Une valeur basse ajoute des nœuds plus vite à la liste *faulty list*. Les valeurs élevées pourraient diminuer la vitesse de détection. La vitesse de détection est importante pour les modèles locaux, puisque l'évaluation d'un nouveau nœud jointif a lieu à partir de rien. En revanche, les modèles globaux obtiennent des informations de réputation pour des nœuds à distance qui par la suite deviendront adjacents, et ainsi, fonctionnent d'une manière proactive. Les simulations ont prouvé qu'avec un seuil bas, OCEAN atteint des performances meilleures que celles d'un modèle global générique. D'autre part, OCEAN est très sensible à la valeur de ce seuil. Les auteurs proposent seulement une valeur empirique, pour des conditions réseau particulières.

Les nœuds malicieux pourraient tirer profit du champ *avoid-list* inclus dans les RREQs, et modifier ce champ pour effectuer des attaques de type wormhole. Les simulations effectuées ont prouvé qu'OCEAN garde des performances acceptables en présence de telles attaques tant que la topologie de réseau n'est pas statique.

OCEAN n'est pas efficace dans la réduction de la bande passante des nœuds malveillants. Le mécanisme des jetons fournit une méthode simple pour décourager le comportement égoïste dans le réseau. Cependant, ce mécanisme réduit la bande passante du réseau. Finalement OCEAN ne prend aucune contre-mesure pour faire face à la collusion.

3.6. Les mécanismes basés sur la rémunération

L'idée derrière les systèmes de paiement est tout à fait simple et elle est inspirée de notre modèle de vie. Dans notre société les gens fournissent des services aux autres et sont payés pour leur travail. L'argent que les personnes gagnent peut être employée pour acheter des produits ou des services des autres personnes à n'importe quel moment, et n'importe où. De même, dans des réseaux informatiques, les nœuds offrent des services à d'autres et obtiennent de la monnaie virtuelle ; une fois qu'ils gagnent assez d'argent, ils peuvent alors acheter des services.

Dans les systèmes de paiement, les nœuds voulant aider les autres peuvent accumuler de la richesse et alternativement gagner un meilleur service, alors que le nœud égoïste ou malveillant ne gagnera pas d'argent. Si un nœud continue à se comporter d'une manière égoïste ou avec malveillance, il sera isolé du réseau après que son compte d'argent soit à zéro.

Dans les systèmes de paiement la gestion des comptes des nœuds doit être sécurisée. Pour cela certains systèmes emploient des modules de sécurité (le projet TermiNodes) ; d'autres systèmes utilisent des jetons (le mécanisme OCEAN [Bansal03]) et d'autres emploient une autorité centrale. Le problème principal des systèmes de paiement c'est que l'existence d'une autorité centrale, pour la gestion des comptes, est toujours nécessaire. Par conséquent, c'est n'est peut être pas la meilleure solution pour les réseaux ad-hoc mobiles.

3.6.1. Les nuglets

Le projet TermiNodes[6] [Buttyan01, Buttyan03] adresse le problème de coopération dans les MANETs par un mécanisme de rémunération. Le premier mécanisme de rémunération a été proposé dans [Buttyan01]. L'approche de stimulation utilisée est basée sur une monnaie virtuelle appelée les nuglets, qui sont employées comme moyen de paiement pour l'expédition de paquets.

3.6.1.1. Les modèles de paiement

En utilisant les nuglets, les auteurs ont proposé deux modèles de paiement : Packet Purse Model (PPM) et Packet Trade Model (PTM). Dans le modèle PPM, la source d'un paquet paie en chargeant quelques nuglets dans le paquet avant de l'envoyer. Les nœuds intermédiaires acquièrent quelques nuglets du paquet lorsqu'ils l'expédient. Si le paquet manque de nuglets, alors il est supprimé.

[6] http://www.terminodes.org/.

Dans le modèle PTM, c'est la destination d'un paquet qui paie le paquet. Pour mettre en place le modèle PTM, chaque nœud intermédiaire "achète" un paquet du nœud précédent pour quelques nuglets et le "vend" au prochain nœud pour plus de nuglets. De cette façon chaque nœud intermédiaire gagne quelques nuglets et tout le coût d'expédition d'un paquet est couvert par la destination.

Pour mettre en place le modèle PTM ou le modèle PPM, un module de sécurité est nécessaire sur chaque nœud pour s'assurer que la quantité correcte de nuglets est déduite ou créditée à chaque nœud. Les auteurs proposent que chaque nœud possède un module de sécurité (*tamper resistant hardware*) qui contrôle son compte en maintenant un compteur de nuglets.

3.6.1.2. Une amélioration : le compteur de crédit

Buttyan et Hubaux [Buttyan03] ont également proposé un mécanisme similaire au précédent, où chaque nœud suit l'évolution de son énergie restante et son crédit. Le protocole proposé exige du nœud de passer chaque paquet (générer localement ou reçu pour l'expédition) à son module de sécurité. Le module de sécurité maintient un compteur, appelé le compteur de nuglet, qui est décrémenté lorsque le nœud veut envoyer un paquet comme source, et est incrémenté lorsque le nœud expédie un paquet. La valeur du compteur de nuglet doit demeurer positive, ce qui signifie que si le nœud veut envoyer ses propres paquets, alors il doit expédier des paquets au profit des autres nœuds. Le compteur de nuglet est protégé contre la manipulation illégitime par le module de sécurité. Par rapport à la proposition précédente, le paquet n'est pas chargé par des nuglets, c'est seulement le compteur de nuglet qui est manipulé localement. Dans ce protocole, chaque deux nœuds voisins essayent d'établir une association de sécurité qui permet de partager des informations, telle qu'une clé de session utilisée pour protéger l'intégrité et assurer l'authenticité des paquets envoyés entre les deux nœuds.

Pour assurer qu'un nœud ne soit récompensé que s'il expédie vraiment le paquet, son compteur de nuglets n'est pas incrémenté immédiatement. Au lieu de cela, c'est le module de sécurité du nœud suivant, qui incrémente un compteur de nuglets temporaire (appelé compteur des nuglets en attentes, *the pending nuglet counter*) qu'il maintient pour le premier nœud. Un protocole de synchronisation de nuglets est exécuté régulièrement par les nœuds. Il permet le transfère des nuglets en attentes, et remet à zéro le compteur des nuglets en attentes.

Il se peut que deux nœuds voisins se déplacent, alors que leurs modules de sécurité veulent exécuter le protocole de synchronisation de nuglets. Si les deux nœuds ne sont plus l'un à la portée de l'autre, alors les compteurs des nuglets en attentes sont remis à zéro, et les nuglets en attentes sont perdus. Par conséquent, le mécanisme ne garantit pas que le nœud reçoive son nuglet pour chaque paquet expédié.

3.6.1.3. Discussion

Bien que le travail présenté dans [Buttyan01] soit le premier à introduire le concept de nuglets, plusieurs problèmes sont à soulever. Un inconvénient majeur du modèle PTM est qu'il permet la surcharge du réseau, puisque la source ne doit pas payer. Pour cette raison, les auteurs de [Buttyan01] ont principalement étudié le modèle PPM. Cependant, le modèle PPM pose un problème aussi : Il semble être difficile d'estimer le nombre de nuglets que la source devrait mettre

dans le paquet au départ. Si la source sous-estime ce nombre, alors le paquet sera supprimé avec une probabilité élevée, et la source perd son investissement. La source peut surestimer le nombre de nuglets, mais ceci mène à une diminution rapide de tout le nombre de nuglets dans le système dû à la suppression des paquets (pour des raisons liés à l'état du réseau) avec beaucoup de nuglets à l'intérieur. Le mécanisme proposé dans [Buttyan03] surmonte ce problème d'évaluation, parce que les paquets n'ont pas besoin de porter des nuglets. En même temps, le problème de surcharge du réseau est résolu, mais le mécanisme reste très sensible à la mobilité des nœuds vu son protocole de synchronisation de nuglets. De plus, il introduit une surcharge considérable en termes de calcul et de bande passante réseau. D'autres problèmes existent dans les deux approches : un nœud qui n'est pas sollicité pour expédier assez de paquets, en raison de sa position et/ou aux communications de ses voisins, ne pourrait pas gagner des nuglets et ne pourra pas envoyer ses propres paquets. Les deux approches exigent un paiement en temps réel. Par conséquent, si le système n'a pas assez de nuglets, il y aura une dégradation dans les performances du réseau.

En conclusion, les deux propositions requirent l'utilisation d'un module de sécurité. Cette exigence n'est pas recommandée pour les réseaux ouverts tels que les MANETs [Marias06].

3.6.2. Sprite

Le système Sprite (A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks) a été proposé dans la référence [Zhong03].

3.6.2.1. L'architecture globale de Sprite

La figure 3.4 montre l'architecture globale de Sprite qui comprend une entité centrale CCS (*Credit Clearance Service*) et un ensemble de nœuds mobiles. Les nœuds sont équipés d'interfaces réseau leur permettent d'envoyer et de recevoir des messages, par exemple en utilisant GPRS dans le réseau WAN, et IEEE 802.11 ou Bluetooth dans un réseau LAN.

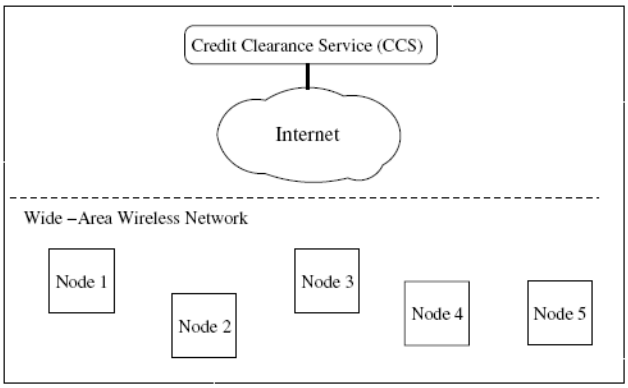


Figure 3.4. L'architecture de Sprite [Zhong03]

Pour identifier chaque nœud, les auteurs supposent que chaque nœud a un certificat délivré par une autorité de confiance (CA), et que chaque nœud source connaît le chemin complet de la source à la destination, en utilisant un protocole de routage sécurisé basé sur le protocole DSR.

Lorsqu'un nœud envoie ses propres messages, il perdra de l'argent (argent virtuelle) au profit du réseau, car d'autres nœuds encourent un coût pour expédier ces messages. D'autre part,

lorsqu'un nœud expédie à d'autres nœuds des messages, il devrait gagner de l'argent pour pouvoir envoyer ses messages plus tard. Dans Sprite, seulement le nœud source paiera l'envoi des messages.

En recevant un paquet, un nœud garde l'accusé de réception signé de ce paquet, qui a été généré par la source. Lorsque le nœud a une connexion au CCS, il signale les paquets reçus en envoyant ses accusés de réception signés.

3.6.2.2. Principe de fonctionnement

Sprite comporte deux fonctionnalités principales : gestion des accusés de réception et un système de calcul du paiement.

A) Gestion des accusés de réception lors de l'expédition

Chaque nœud n_i dispose d'une paire de clés (PK_i, SK_i) ;

$seq_i(j,k)$ est le numéro de séquence du message envoyer de n_j à n_k, tel que observé par n_i ;

($sign_{SK}()$, $verify_{PK}()$) est un mécanisme de signature digitale.

1) Envoi d'un message

Supposons que le nœud n_0 veuille envoyer un message m avec le numéro de séquence $seq_0(0,d)$ au nœud n_d à travers le chemin p. La figure 3.5 résume les étapes exécutées par le nœud n_0, telle que MD() est une fonction de hachage tel que MD5 ou SHA-1.

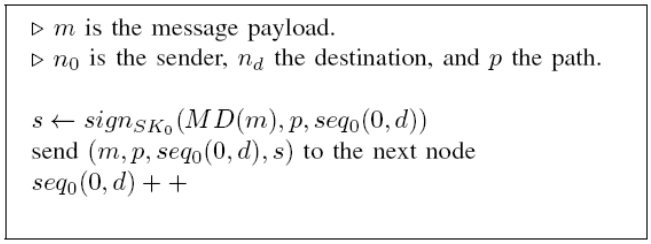


Figure 3.5. Le nœud n_0 envoie un message au nœud n_d [Zhong03]

2) Réception d'un message

Supposons que le nœud ni reçoive (m,p,seq,s).

Premièrement, le nœud vérifie trois conditions : (1) n_i est sur le chemin ; (2) le message a un numéro de séquence supérieur à $seq_i(0,d)$; et (3) la signature est valide. Si l'une des conditions n'est pas vérifiée, le message est supprimé sinon le nœud n_i sauvegarde (MD(m),p,seq,s) comme accusé de réception. Si n_i n'est pas la destination et décide d'expédier le message, il envoie (m,p,seq,s) au prochain nœud. La figure 3.6 résume les étapes exécutées par le nœud n_i.

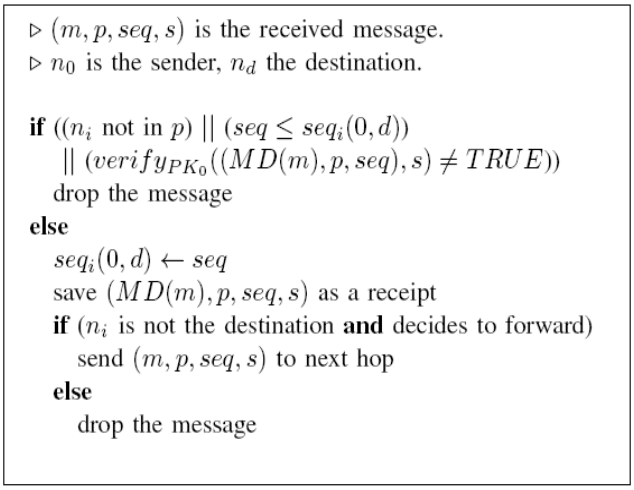


Figure 3.6. Le nœud n_i reçoit (m,p,seq,s) [Zhong03]

B) Calcul du paiement

Un accusé de réception (D, p, seq, s) soumis par le nœud n_i est considéré comme valide si $verify_{PK0}$((D, p, seq), s) = TRUE, où PK_0 est la clé publique de l'émetteur.

Les auteurs supposent que p = (n_0, n_1,…, n_e,…n_d), où n_e est le dernier nœud, sur le chemin p, qui a envoyé un accusé de réception valide avec le numéro de séquence seq. CCS débite le nœud n_0 avec C unités, et crédite le nœud n_i avec P unités.

$$C = (d-1)\alpha + \beta - (d-e)\gamma\beta,$$

$$P_i = \begin{cases} \alpha & \text{if } i < e = d \\ \beta & \text{if } i = e = d \\ \gamma\alpha & \text{if } i < e < d \\ \gamma\beta & \text{if } i = e < d. \end{cases}$$

3.6.2.3. Discussion

Dans [Zhong03], les auteurs ont prouvé formellement que le système de paiement est correct (il motive chaque nœud pour rapporter son comportement honnêtement) en utilisant la théorie des jeux. L'inconvénient de Sprite est l'hypothèse d'une connexion au CCS pour qu'un nœud envoie ses accusés de réception signés. Un nœud central (CCS) n'est pas vraiment conseillé pour un réseau ad-hoc pur. De plus, ce nœud central constitue un goulot d'étranglement, et sa défaillance bloque tout le réseau. Le mécanisme utilisé pour la gestion des accusés de réception introduit une charge supplémentaire en termes de puissance de calcul et de bande passante réseau.

En plus de la disponibilité d'une autorité centrale, Sprite suppose un routage par la source, et une infrastructure à clé publique. Les auteurs n'expliquent pas comment le paiement de l'expéditeur aux nœuds est fait, si les nœuds ont des comptes avec le CCS qui transfère le paiement ou si les nœuds se rémunèrent directement.

3.7. Comparaison et discussion

3.7.1. Comparaison

Pour comparer les mécanismes présentés dans ce chapitre, nous utilisons un ensemble de métriques qui permettent d'effectuer une comparaison significative. Ces métriques sont définies comme suit :

- Classe : nous distinguons deux classes : les mécanismes basé-réputation (notée Rep) et les mécanismes basé-rémunération (notée Rém).
- Type : pour les systèmes basés réputation, nous distinguons deux types : local (noté L) et global (noté G), et pour les systèmes basés rémunération nous distinguons ceux qui utilisent un module de sécurité (noté MS) et ceux qui utilisent une banque virtuelle (noté BV).
- Confiance : le mécanisme utilise la notion de confiance et fait la différence entre la réputation et la confiance.
- Contexte : le mécanisme supporte la notion du contexte. La réputation calculée est relative à un service (contexte) fourni par un nœud.
- Temps : le mécanisme incorpore la notion du temps. La réputation est dynamique si elle varie en fonction du temps, dans le cas contraire elle est dite statique.
- Type de réputation : indique le type de réputation diffusée dans le réseau. Nous distinguons trois types, réputation positive (P), réputation négative (N), ou les deux (PN).
- Mécanisme de deuxième chance : le modèle de coopération utilise un mécanisme de deuxième chance qui permet de réintégrer des nœuds précédemment exclus.
- Passage à l'échelle : le mécanisme support le passage à l'échelle (*scalability*). Pour les mécanismes basés réputation, cette caractéristique est liée directement au protocole d'échange des informations de réputation.
- Authentification : le modèle de coopération utilise un mécanisme pour l'authentification des nœuds. Un modèle qui utilise l'authentification est un modèle robuste contre une attaque par personnification.
- Robustesse contre les attaques : le mécanisme est robuste contre les attaques suivantes ; Robustesse contre la collusion (notée C), Robustesse contre l'inactivité (notée I), Robustesse contre la diffamation (notée D).
- Protection des messages de réputation : le mécanisme protège les messages de réputation, échangés entre les nœuds, contre des attaques telles que la modification.
- Valeurs de réputation (de confiance) : les systèmes de réputation emploient typiquement des valeurs de réputation et de confiance pour représenter concrètement ces notions, ces valeurs peuvent être des valeurs discrètes ou continues. Quelques modèles utilisent des valeurs binaires, impliquant qu'un nœud, soit il est confiant, soit il est méfiant à l'égard d'un autre nœud. Pour la comparaison nous utilisons la notation suivante : RD (valeurs de réputation discrètes), RC (valeurs de réputation continues), RB (valeurs de réputation binaires), CD (valeurs de confiance discrètes), CC (valeurs de confiance continues), CB (valeurs de confiance binaires).

- Le coût en bande passante : il est lié à la quantité d'information, échangée entre les nœuds, induite par le mécanisme de coopération. Pour les mécanismes basé-réputation c'est l'échange des informations de réputation entre les nœuds, nous distinguons quatre cas :
  - pas d'échange, un coût égal à 0.
  - échange avec le voisinage direct : complexité moyenne $O(pxn)$, avec p le nombre moyen de nœuds voisins par nœud, et n le nombre total de nœuds dans le réseau.
  - échange avec les nœuds amis ou les nœuds ayant une bonne réputation (utilisé par [Buchegger02, Liu04]), la complexité dans ce cas dépend de la configuration du réseau, nous pouvons supposer qu'elle est généralement supérieure à $O(pxn)$ et peut atteindre $O(n^2)$.
  - échange avec tous les nœuds du réseau : complexité moyenne $O(n^2)$, avec n le nombre de nœuds dans le réseau.

  Pour les mécanismes basé rémunération, ce coût est variable est dépend directement de la quantité d'information échangée (paquets de données et paquets de contrôles).
- Le coût en stockage : il est lié à la quantité d'information qu'un nœud doit stocker, tel que exigé par le mécanisme de coopération.

  Pour les mécanismes basé réputation, c'est l'information de réputation et de confiance, nous distinguons trois cas :
  - le nœud stocke une partie de l'historique de ses expériences à propos de son voisinage direct.
  - le nœud stocke une partie de l'historique de ses expériences à propos un ensemble restreint de nœuds (ses amis ou les nœuds avec une bonne réputation).
  - le nœud stocke une partie de l'historique de ses expériences à propos de tous les nœuds.

  Pour les mécanismes basé rémunération, ce coût est limité à (i) des accusés de réception et un compteur de crédit, sur chaque nœud du système Sprite [Zhong03] et à (ii) un compteur de crédit et un compteur de nuglets temporaire, sur chaque nœud, pour le mécanisme de nuglets [Buttyan03].
- Le coût en puissance de calcul : puisque le coût lié à l'émission et à la réception de paquets est linéairement proportionnel au coût en bande passante, nous discutons seulement le coût lié au traitement local introduit par le mécanisme de coopération.

La comparaison entre les mécanismes de coopération est résumée dans le tableau 3.1.

	CONFIDANT		CORE	OCEAN	Liu et Issarny	SORI	Sprite	Nuglets
	v1	v2						
Classe	Rép	Rép	Rép	Rép et Rém	Rép	Rép	Rém	Rém
Type	G	G	G	L	G	G	BV	MS
Confiance	Oui	Oui	Non	Non	Oui	Non	-	-
Contexte	Non	Non	Oui	Non	Oui	Non	Non	Non
Temps	Oui	Oui	Oui	Non	Oui	Non	Non	Non
Mécanisme de 2^{eme} chance	Non	Oui	Non	Oui	Non	Non	-	-
Type de réputation	N	PN	P	N	PN	PN	-	-
Passage à l'échelle	Non	Non	Non	Oui	Non	Non	Non	Oui
Authentification	Non	Non	Non	Non	Non	Oui	Oui	Oui
Robustesse contre les attaques	Non	C, D	Non	Non	C, I, D	Non	C	C
Protection des messages de réputation	Non	Non	Non	Non	Non	Oui	-	-
Valeurs de réputation et de confiance	RC, CD	RC, CC	RC	RC	RC, CC	RC	-	-
Le coût en bande passante	moyen	très élevé	très élevé	très faible	moyen	faible	très faible	faible
Le coût en stockage	faible	élevé	élevé	très faible	moyen	faible	faible	très faible
Le coût en puissance de calcul	très faible	très faible	très faible	très faible	très faible	faible	très faible	faible

Tableau 3.1. Tableau de comparaison des mécanismes de coopération

3.7.2. Discussion

3.7.2.1. Les systèmes de paiement et les systèmes de réputation

Les systèmes de paiement servent d'incitation pour fournir un service bien défini, tel que l'expédition de paquets. Pour assurer le paiement, le module de sécurité (utilisé dans [Buttyan03]) et la banque virtuelle (utilisée dans [Zhong03]) ont été suggérés, ceci n'est pas conseillé pour des réseaux ouverts tels que les MANETs [Marias06]. Les systèmes de paiement peuvent empêcher le comportement égoïste. Cependant, ils ne traitent pas le comportement malveillant ou défectueux.

Les systèmes de réputation s'appliquent à une gamme plus large du comportement tant qu'il est observable et classifiable. Ils peuvent, s'ils emploient l'information globale et les moyens nécessaires pour faire face aux fausses informations, empêcher partiellement le comportement malveillant en excluant les nœuds malveillants. De cette façon, les nœuds peuvent se protéger avant de rencontrer les nœuds malveillants.

Si les systèmes de réputation se fondent exclusivement sur l'information locale pour établir des estimations de réputation, ils peuvent seulement empêcher le comportement malveillant déjà éprouvé par un nœud.

Par opposition aux systèmes de paiement, les systèmes de réputation ne supposent pas un équilibre en termes de service fourni et service utilisé. Un système de réputation pénalise simplement un nœud s'il ne fait pas ce qui est supposé faire. Cette différence offre un avantage dans les situations où un nœud n'est pas simplement en position à coopérer, par exemple lorsqu'il occupe l'extrémité du réseau et par conséquent il n'obtient pas assez de demandes.

3.7.2.2. Les systèmes de paiement

Sprite [Zhong03] permet d'inciter les nœuds à coopérer d'une manière efficace (il motive chaque nœud pour rapporter son comportement honnêtement). Il a été modélisé et prouvé en utilisant la théorie de jeux. Sprite s'est limité à l'expédition des paquets et n'incorpore pas la notion du contexte. Son utilisation d'une autorité centrale limite son passage à l'échelle, par contre, cette autorité permet d'authentifier les nœuds du réseau et le modèle de paiement proposé résiste à l'attaque par collusion. Les auteurs ont montré que le coût induit par le système en termes de bande passante, d'espace de stockage et de puissance de calcul n'est pas significatif. Sprite ne peut pas être utilisé dans des réseaux ad-hoc purs. Pour des réseaux ad-hoc disposant d'une connexion Internet, l'utilisation de Sprite est possible, mais les auteurs n'expliquent pas comment le paiement de l'expéditeur aux nœuds se fait.

Le mécanisme proposé dans [Buttyan01] n'utilise pas une autorité centrale mais requiert l'utilisation d'un module de sécurité. Il est limité à l'expédition des paquets et n'incorpore pas la notion du contexte. Le mécanisme proposé supporte le passage à l'échelle. L'amélioration de ce mécanisme proposée dans [Buttyan03] est très sensible à la mobilité des nœuds vu son protocole de synchronisation de nuglets qui introduit une surcharge considérable en termes de calcul et de bande passante réseau.

3.7.2.3. Les systèmes de réputation

CONFIDANT [Buchegger04] est un système de réputation global robuste. Il incorpore la confiance et le temps et emploi une approche bayesienne binaire pour utiliser les deux types d'informations (positive et négative) tout en minimisant le risque de fausses informations. CONFIDANT n'utilise pas la notion de contexte et il s'est limité aux fonctionnalités de routage et d'expédition. Bien que CONFIDANT soit simple, ce qui limite la consommation d'énergie lors des calculs, il induit un coût important en termes d'espace de stockage et de bande passante, ce qui limite son passage à l'échelle.

CORE [Michiardi02] utilise la notion du temps et du contexte mais il s'est limité aux fonctionnalités de routage et d'expédition. Il est vulnérable à l'attaque par collusion et aucun mécanisme de deuxième chance n'est considéré. Dans CORE, la consommation d'énergie lors des calculs est très réduite, mais le stockage des informations de réputation et leur diffusion génèrent un coût important d'espace de stockage et de bande passante, ce qui limite son passage à l'échelle.

OCEAN [Bansal03] est un mécanisme local de réputation qui se limite à l'expédition de données. Son traitement local le rend plus simple que les autres mécanismes. L'utilisation exclusive des informations locales permet (i) d'éviter des mécanismes compliqués pour la gestion de la confiance et des informations globales et (ii) de réduire considérablement le coût induit en termes d'espace de stockage et de bande passante. D'autre part, le fonctionnement réactif d'OCEAN limite considérablement la punition des nœuds malveillants. OCEAN utilise un mécanisme de paiement basé sur l'utilisation des jetons. Il permet de faire face au comportement égoïste, mais il détériore la bande passante réseau. Finalement, OCEAN supporte le passage à l'échelle, mais il reste sensible à la mobilité des nœuds. Les simulations effectuées ont prouvé qu'OCEAN garde des performances acceptables en présence d'attaques de type wormhole tant que la topologie de réseau n'est pas statique.

SORI [Qi04] est un système global de réputation qui se focalise exclusivement sur la fonctionnalité d'expédition de données. Il est statique et n'emploi ni la notion de confiance ni la notion de contexte. Les caractéristiques principales de SORI sont (i) l'authentification des nœuds et (ii) la protection des messages de réputation lors de leur diffusion. Bien que la diffusion des messages de réputation soit limitée au voisinage à un saut, le mécanisme de sécurité complémentaire limite le passage à l'échelle de SORI.

Le mécanisme de Liu et Issarny [Liu04] est un mécanisme générique. Il n'est pas limité à un service donné, mais il vise divers types de services. Il a été démontré dans [Liu04] que ce mécanisme est robuste contre les attaques suivantes : (i) inactivité, (ii) diffamation et (iii) collusion. Le mécanisme ne supporte pas le passage à l'échelle et aucun mécanisme de deuxième chance n'est considéré.

3.7.2.4. Remarques

Il a été constaté dans [Buchegger03-2] que le temps de détection des nœuds malveillants peut être relativement grand, voir même inacceptable. Pour cela, l'utilisation d'un système de réputation global est fortement recommandée pour des applications exigeant un temps raisonnable de détection de nœuds malveillants. Il faut en même temps employer la notion de confiance combinée avec un mécanisme efficace pour faire face aux fausses informations. Le modèle bayesian employé dans CONFIDANT est une des techniques utilisées.

La notion du temps est importante et il nous semble crucial que la réputation soit dynamique, tel que le *fading* utilisé dans CONFIDANT et qui introduit un affaiblissement exponentiel au comportement passé. Ce qui permet une sorte de rachat et empêche en même temps un nœud de se conduire mal par le profit d'une bonne réputation établie dans le passé.

Un mécanisme de deuxième chance, qu'il soit explicite (tel que dans OCEAN) ou implicite (tel que dans CONFIDANT), est fortement recommandé pour un système de réputation pour les MANETs, où la surveillance du comportement n'est pas tout le temps un service garanti.

Certains systèmes de réputation, tel que CORE, sont bâtis sur l'échange des informations de réputation positive. Ils sont utilisés beaucoup plus lorsqu'on a le choix entre plusieurs entités (par exemple des vendeurs) et qu'on souhaite trouver le meilleur (tel que les sites d'achat sur Internet, eBay par exemple). Dans les MANETs, l'objectif visé est différent, c'est plutôt l'isolement des nœuds malveillants qui prime. Nous pensons que pour les MANETs un système de réputation doit utiliser toute l'information de réputation disponible, qu'elle soit positive ou négative.

L'échange des informations de réputation entre les nœuds du réseau est un facteur déterminant la propriété "passage à l'échelle". Un système local de réputation supporte le passage à l'échelle. Pour un système global de réputation, ceci dépend de l'algorithme utilisé pour la diffusion des informations de réputation. Employer des algorithmes de diffusion supportant le passage à l'échelle est certainement bénéfique pour les systèmes de réputation. Les algorithmes de diffusion épidémiques nous semblent un choix intéressant pour la diffusion des informations de réputation.

Bien que les informations de réputation puissent être manipulées par des nœuds malveillants lors de leur diffusion, la plupart des systèmes de réputation ne protègent pas les messages de réputation diffusés contre ce type d'attaque.

La mobilité est une caractéristique intrinsèque et elle ne doit pas être une entrave pour le mécanisme de coopération. Dans OCEAN par exemple, les nœuds malicieux pourraient tirer profit du champ avoid-list inclus dans les RREQs, et modifier ce champ pour effectuer des attaques de type wormhole. OCEAN garde des performances acceptables si la mobilité des nœuds reste importante. Par contre, pour le mécanisme développé dans [Buttyan03], une mobilité significative peut poser des problèmes (la perte des nuglets en attentes mène à une diminution de tout le nombre de nuglets dans le système et par conséquent, si le système n'a pas assez de nuglets, il y aura une dégradation dans les performances du réseau).

3.8. Conclusion

Dans ce chapitre, nous avons recensé, décrit et discuté les mécanismes de coopération les plus importants existants dans la littérature. Les mécanismes de punition proposés sont simples et basés sur des idées de la théorie des jeux, mais ils ne sont pas vraiment adaptés aux réseaux ad-hoc non coopératifs. Les mécanismes de paiement peuvent empêcher le comportement égoïste, mais pas le comportement malveillant. Par contre, les mécanismes de réputation traitent tout type de comportement tant qu'il est observable et classifiable. Nous avons aussi proposé une taxonomie des mécanismes de coopération, définit un ensemble de métriques et effectué une comparaison qualitative des mécanismes de coopération. Finalement, nous avons discuté les résultats de cette comparaison.

Chapitre 4. PARDON : le mécanisme de coopération proposé

4.1. Introduction

Dans le chapitre précédent, nous avons présenté un état de l'art sur les mécanismes de coopération utilisés principalement dans les MANETs et spécialement dans la couche réseau. Dans le présent chapitre, nous présentons notre mécanisme de coopération PARDON (**P**robabilistic and **A**daptive **R**eputation system for **D**ynamic ad-h**O**c **N**etworks).

PARDON est un mécanisme de coopération basé réputation avec une architecture modulaire qui repose sur une approche bayesienne pour la gestion de la réputation. C'est un système de réputation globale qui exploite toute l'information de réputation disponible. La collecte des informations de réputation repose sur un protocole de dissémination épidémique et les recommandations reçues sont filtrées à l'aide d'un algorithme de filtrage hybride.

4.2. Modèle et hypothèses

4.2.1. Le protocole de réplication

Dans notre approche, nous supposons l'existence d'un protocole de réplication push-based[7] générique (sans aucun mécanisme de coopération), modélisé comme suit (voir figure 4.1).

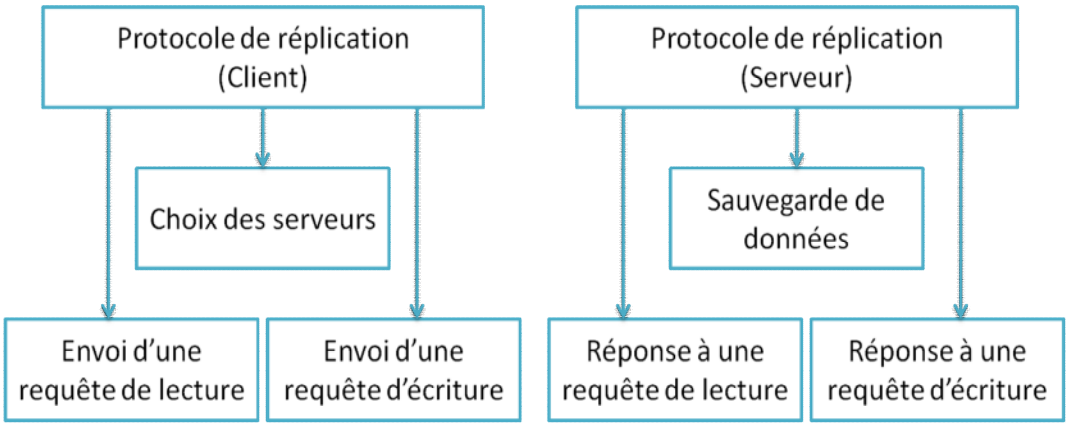


Figure 4.1. Modélisation du protocole de réplication

L'utilisation d'un protocole de réplication push-based générique permet de concevoir une architecture indépendante de tout protocole de réplication. Le mécanisme de coopération peut ainsi être intégré à n'importe quel protocole de réplication. Le principe de fonctionnement du protocole de réplication est comme suit :

1) Pour des raisons simplification, nous supposons que n'importe quel nœud peut être un serveur de réplication ;

[7] Les nœuds client choisissent leurs serveurs de réplication.

2) Un nœud client choisit un ensemble de nœuds serveurs : des serveurs de réplication candidats ;

3) Un nœud décide des nœuds avec lesquels il doit interagir (une sélection selon un critère prédéfini qui dépend du protocole de réplication) ;

4) Une fois les serveurs de réplication choisis, le nœud client envoie les données à répliquer aux serveurs. Notons qu'il existe deux types de protocoles de réplication : les protocoles push-based, où les nœuds choisissent eux mêmes les serveurs de réplication, et les protocoles pull-based, où les nœuds demandent de stocker des données. Dans les protocoles pull-based, le problème de la non coopération ne se pose pas du moment qu'un nœud décide de son propre gré d'être un serveur de réplication pour d'autres nœuds. Pour le problème de non coopération, nous nous intéressons particulièrement aux protocoles push-based.

4.2.2. La relation entre le protocole de réplication et le mécanisme de coopération

L'architecture que nous proposons comprend le protocole de réplication comme composant de la solution. La figure 4.2 illustre la relation qui existe entre le protocole de réplication et notre mécanisme de coopération PARDON.

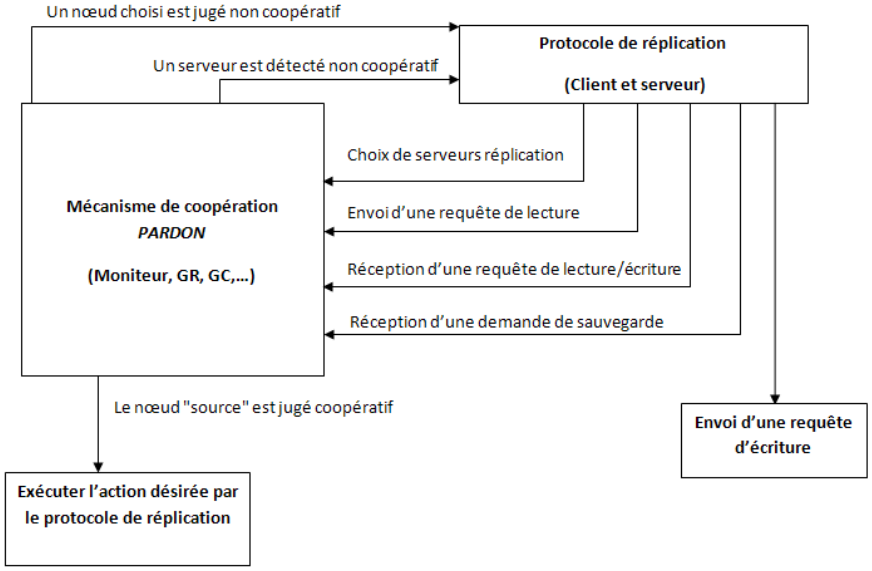


Figure 4.2. La relation entre le protocole de réplication et le mécanisme de coopération

Une fois que le protocole de réplication sélectionne les serveurs de réplication, le choix est communiqué au mécanisme PARDON. Celui-ci décide du choix final des serveurs et informe le protocole de réplication sur les serveurs non coopératifs exclus pour prendre les décisions nécessaires (refaire le choix, réduire le nombre de serveurs de réplication, etc.). Lorsqu'un ou plusieurs serveurs sont jugés non coopératifs après le choix final, PARDON exclut ces serveurs et informe le protocole de réplication sur les serveurs exclus pour prendre les décisions nécessaires (par exemple répliquer de nouveau une partie des données).

Lorsque le protocole de réplication reçoit une requête de lecture/écriture ou une demande de sauvegarde, le protocole consulte PARDON pour étudier la possibilité de répondre ou non à cette requête. Pour l'envoi d'une requête de lecture, le protocole de réplication informe PARDON, ceci permet de surveiller de prés le serveur destination.

4.2.2.2. Gestion de l'identité

L'identité est un concept important dans les systèmes de coopération. Nous supposons qu'elle est persistante et unique. (i) L'identité doit être persistante, ainsi un nœud ne peut pas facilement changer son identité. L'une des méthodes proposés est l'utilisation des "*expensive pseudonyms*" [Friedman01] ou l'utilisation d'un module de sécurité et (ii) L'identité doit être unique, ainsi aucun nœud ne peut employer l'identité d'un autre nœud. Montenegro et Castelluccia [Montenegro02] proposent l'utilisation des identifiants uniques générés par cryptographie.

4.3. L'architecture du mécanisme de coopération PARDON

Le mécanisme de coopération PARDON que nous proposons comprend cinq composants à savoir : (i) le moniteur, (ii) le gestionnaire de réputation (GR), (iii) le gestionnaire de recommandation (GC), (iv) le gestionnaire de l'espace de stockage (GE) et (v) l'estimateur de l'état du réseau (ER).

Les figures 4.3 et 4.4 représentent l'architecture modulaire du mécanisme PARDON conçue pour faire face aux différents types d'attaques liées aux MANETs non coopératifs. Chaque module de PARDON accomplie une tâche bien spécifique. Dans la suite de cette section, nous détaillons ces cinq composants.

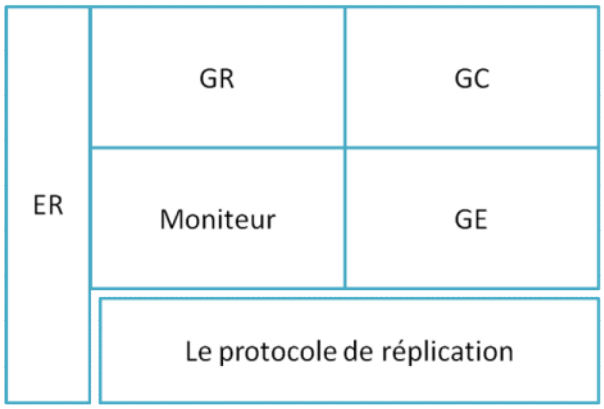


Figure 4.3. L'architecture générale de PARDON

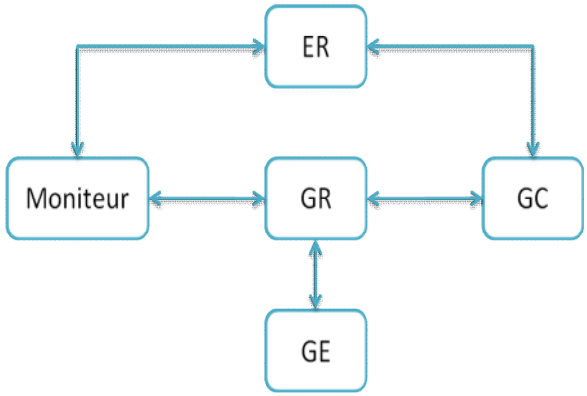


Figure 4.4. L'architecture détaillée de PARDON

4.3.1. Le moniteur

Le moniteur proposé permet d'effectuer deux types de vérification : (i) une vérification proactive, qui permet de vérifier si le serveur de réplication stocke réellement une copie des données localement, et (ii) une vérification réactive, qui permet de vérifier si le serveur de réplication répond aux différentes requêtes.

Le rôle du moniteur est donc double : (i) envoyer des requêtes de vérification de manière proactive aux serveurs de réplication et reporter les réponses (positifs ou négatifs) au gestionnaire de réputation. (ii) lors de l'envoi des requêtes de lecture, il faut vérifier d'une manière réactive que le serveur répond correctement à ces requêtes.

Dans le cas de la réplication de données (ou services), un problème crucial se pose : comment s'assurer que les serveurs de réplication stockent réellement une copie des données localement et ne prétendent pas à le faire ?

Les solutions standard [Caronni03] pour vérifier si un serveur de réplication stocke réellement une copie d'un certain fichier sont : (i) soit de lui demander d'envoyer le fichier de nouveau au client pour une éventuelle vérification, (ii) soit de lui demander de calculer et d'envoyer le hach du fichier (si une preuve de fraîcheur est nécessaire, le hachage se fait en utilisant un hachage paramétré avec une clé définie par le client). Malheureusement, cette solution exige une bande passante considérable (premier cas) ou une puissance de calcul importante (deuxième cas).

Un compromis viable est que le client demande le hach d'un bloc (ou d'un ensemble de blocs) de données choisi d'une manière aléatoire [Caronni03]. Cette solution est très pratique dans les MANETs en raison de leurs contraintes et c'est la solution que nous adoptons. Dans la suite de ce rapport, nous nous référons à cette solution par le nom "acquittement".

Pour accomplir la première tâche (la vérification proactive), nous proposons d'utiliser le principe d'écoute active en utilisant un protocole de défi-réponse qui consiste en : (i) exiger un acquittement lors de l'envoi des données pour un éventuel stockage, et (ii) exiger un acquittement d'une manière périodique.

L'idée du protocole défi-réponse a été utilisée principalement dans les réseaux peer-to-peer, plus précisément dans les systèmes de stockage distribués, tels que Samsara [Cox03], Pastiche [Cox02], CIBS [Lillibridge03], Free Haven [Dingledine00] et autres. Dans ces systèmes, le protocole défi-réponse consiste principalement à exiger un acquittement d'une manière périodique avec une période de durée fixe.

Les protocoles défi-réponse, tels qu'utilisés dans les réseaux peer-to-peer, consomment une quantité importante d'énergie et génèrent un trafic supplémentaire considérable. L'utilisation directe de ce type de protocole pose un sérieux problème dans les MANETs en raison de leurs contraintes. On peut penser à une solution partielle qui consiste à utiliser une fonction de hachage moins gourmande en énergie, telle que Adler-32 [Deutsch96], mais les problèmes d'énergie et du trafic supplémentaire restes toujours posés. Pour faire face à ces problèmes, nous proposons dans un premier temps d'introduire l'aspect probabiliste dans le moniteur pour concevoir un moniteur probabiliste.

4.3.1.1. Le moniteur probabiliste

L'idée de la vérification probabiliste est la suivante : du moment que le nœud serveur ne peut pas prévoir le temps d'occurrence de la procédure de vérification, le nœud client (propriétaire de la donnée) n'est pas obligé de lancer une vérification de manière périodique à période fixe. Au lieu de cela, le nœud client lance une vérification de manière périodique avec une période de durée variable.

L'aspect probabiliste de notre moniteur est basé sur un schéma que nous appelons le schéma On/Off. Nous définissons deux types de période : OnPériode (notée OnP) et OffPériode (notée OffP). Nous considérons dans un premier temps que les deux périodes sont fixes.

Le monitoring ne se fait que durant la période OnP. La période OnP démarre avec une probabilité P_{On} et la période OffP démarre avec une probabilité P_{Off}, tel que :

$$P_{On} + P_{Off} = 1.0$$

La figure 4.5 représente la chaîne de Markov (CM) associée à notre schéma On/Off.

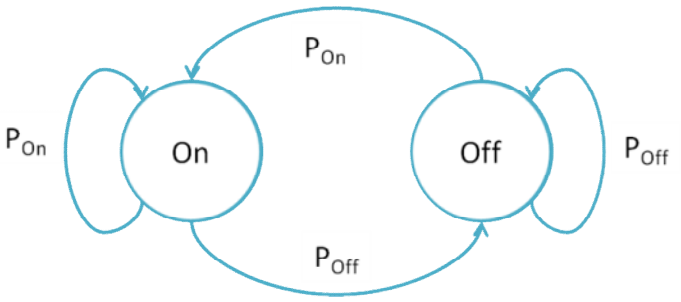


Figure 4.5. La chaîne de Markov associée au schéma On/Off

Discussion :

La matrice de transition M associée à cette CM (figure 4.5) est :

$$M = \begin{bmatrix} P_{on} & P_{off} \\ P_{on} & P_{off} \end{bmatrix}$$

1) **Cas 1** : $P_{On} = 1$ ($P_{Off} = 0$), nous aurons :
   - Etat On : absorbant avec une période égale à 1
   - Etat Off : transitoire
   - La distribution stationnaire se résume à l'état On avec une probabilité égale à 1.
2) **Cas 2** : $P_{Off} = 1$ ($P_{On} = 0$), nous aurons :
   - Etat Off : absorbant avec une période égale à 1
   - Etat On : transitoire
   - La distribution stationnaire se résume à l'état Off avec une probabilité égale à 1.
3) **Cas 3** : $0<P_{On}<1$ et $0<P_{Off}<1$, cette CM contient une classe dont les états sont On et Off. Le type de cette classe est persistant avec une période égale à 1.La distribution stationnaire de cette chaîne est :

Etat	On	Off
Probabilité	P_{On}	P_{Off}

Donc, si on souhaite plus de vérification, il suffit d'augmenter P_{On} (donc diminuer P_{Off}) et vice versa.

Un monitoring probabiliste permet, d'un coté, d'économiser de l'énergie pour les deux nœuds impliqués dans le processus de vérification (le nœud client et le nœud serveur), ce qui constitue un gain considérable. De plus, cela permet en même temps de diminuer le trafic supplémentaire généré par le protocole défi-réponse. D'un autre coté, la vérification probabiliste n'influence pas négativement l'efficacité du protocole défi-réponse du moment que le nœud serveur impliqué dans le processus de vérification n'à aucun moyen de deviner à quel moment le processus de vérification sera lancé par le nœud client.

Par rapport au cas des réseaux peer-to-peer, la probabilité de perdre un paquet dans les MANETs est beaucoup plus importante. La perte des paquets[8] peut fausser les conclusions tirées par le client après un défi-réponse (un nœud est détecté comme étant un nœud non coopératif alors que des paquets ont été perdus lors du processus de vérification). On parle dans ce cas de faux positifs. Le problème des faux positifs n'a pas été traité dans les protocoles défi-réponse utilisés dans les réseaux peer-to-peer [Dingledine00, Cox02, Lillibridge03 et Cox03]. Notre schéma On/Off permet de réduire le trafic supplémentaire généré et donc de réduire les faux positifs.

Bien que le schéma On/Off permet de résoudre partiellement le problème de consommation de l'énergie et le problème du trafic supplémentaire généré, le protocole défi-réponse pose toujours le problème des faux positifs lorsqu'il est utilisé dans les MANETs. Les faux positifs ne peuvent pas être entièrement éliminés [Caronni03], mais on peut plutôt les réduire.

Pour réduire encore plus les faux positifs nous proposons deux idées : (i) réduire au maximum les périodes de vérification (les périodes OnP) sans influencer négativement l'efficacité du protocole et (ii) lancer le protocole défi-réponse lorsque nous estimons que les conditions réseaux sont bonnes (réduire la probabilité de perte des paquets envoyés).

4.3.1.2. Le moniteur probabiliste adaptatif

Pour réduire les périodes OnP sans influencer négativement l'efficacité du protocole défi-réponse, nous essayons d'ajuster les deux probabilités P_{On} et P_{Off} en considérant la réputation du nœud à vérifier :

(a) Pour des nœuds ayant des réputations relativement bonnes, nous essayons de pratiquer moins de vérification. Cependant, il faut que la réduction des fréquences de vérification soit toujours probabiliste pour éviter l'exploitation du mécanisme lui-même par des nœuds malveillants. Pour cela, il suffit de diminuer P_{On} (ou d'augmenter P_{Off}).

(b) Pour des nœuds ayant des réputations relativement mauvaises, nous essayons de pratiquer plus de vérification. Ceci, permet de détecter les nœuds non coopératifs le plus tôt possible et de réduire ainsi leur influence négative. Puisqu'une réputation relativement mauvaise n'indique pas nécessairement qu'il s'agit d'un nœud non coopératif alors l'augmentation des fréquences de vérification doit être toujours probabiliste. Pour cela, il suffit de diminuer P_{Off} (ou d'augmenter P_{On}).

[8] Paquets de requêtes/acquittements pour le protocole défi-réponse et requêtes/réponses de lecture pour le protocole de réplication.

Pour accomplir (a) et (b) il faut donc ajuster P_{On} en fonction de la réputation du nœud. Nous proposons la fonction d'ajustement suivante :

$$P_{On} = f(réputation) = 1 - RSL$$

La réputation de service locale (RSL) est discutée dans la section 4.3.4.4.

Lancer le protocole de vérification alors que le nœud client se situe dans une région congestionnée augmente la probabilité d'occurrence des faux positifs. Pour résoudre ce problème, nous proposons d'estimer l'état du réseau avant de lancer la procédure de vérification. Si le nœud client peut estimer l'état du réseau, il peut adapter son comportement comme suit : si l'état du réseau est bon (pas de congestion : le nœud estime l'état du réseau à partir de son état local et l'état de son voisinage) alors la procédure de vérification peut être lancée. Sinon, retarder la vérification.

L'algorithme 1 représente le pseudo code de notre protocole défi-réponse. L'aspect probabiliste et l'aspect adaptation de notre moniteur sont intégrés dans l'algorithme 2.

Algorithme 1 Le protocole défi-réponse

```
1: Envoi d'une requête de vérification
2: réponse_reçue = false
3: SI réponse reçue [positive ou négative] avant timeout alors
4:       réponse_reçue = true
5:       Communiquer l'évènement au GR
6: FSI
7: SI timeout et réponse_reçue == false alors
8:       Communiquer l'évènement au GR
9: FSI
```

Algorithme 2 Le moniteur probabiliste et adaptatif

```
1: Si (le temps restant dans l'état actuel de la CM est nulle) alors
2:       Calculer Pon
3:       Passage de la CM à l'état X
4:       Si (X == On et conditions réseau bonnes) alors
5:                Déclencher le protocole défi-réponse
6:       FSI
7: FSI
```

Le comportement adaptatif du moniteur pour prendre en charge la perte des paquets et son aspect probabiliste pour économiser de l'énergie constitue une solution intéressante pour le monitoring dans les MANETs. Il faut noter que le moniteur reste néanmoins un service non garanti, mais ce point sera considéré dans la conception du GR.

4.3.1.3. Attaques possibles

Bien que la solution proposée jusqu'à maintenant soit très pratique dans les MANETs, vu leurs contraintes, elle reste vulnérable à quelques attaques. Quatre attaques ont été identifiées dans [Caronni03] concernant les protocoles défi-réponse dédiés aux systèmes de stockage distribués et qui sont toujours valables dans notre cas. Ces attaques sont :

1) Pré-calculer les acquittements ;

2) Le serveur de réplication peut envoyer la requête de vérification à un autre serveur de réplication ;

3) Collusion : le serveur de réplication peut envoyer la requête de vérification à un serveur de réplication complice ;

4) Le serveur de réplication peut télécharger à la demande les blocs de données nécessaires pour accomplir la vérification.

L'attaque (1) employant des données pré-calculées est contrecarrée en utilisant un hachage paramétré avec une clé *message authentication code* (MAC) définie par le client [Tsudik92], ou en modifiant la gamme des données à vérifier suffisamment. Cette solution a été proposée dans [Caronni03].

Nous remarquons que les attaques (2, 3 et 4) nécessitent un temps supplémentaire. Pour faire face à ces attaques, une solution simple consiste à contrôler le temps écoulé entre l'instant d'envoie des requêtes de vérification et l'instant de réception des réponses pour détecter les délais additionnels. Donc, si le nœud client peut estimer la durée moyenne d'un va-et-vient[9] (requête-réponse), il peut détecter ces attaques en utilisant un timer lors de l'envoi de la requête. Malheureusement, il a été constaté que, généralement, il est impossible de détecter les délais additionnels même si tous les nœuds et liens fournissent des garanties en temps réel dures [Caronni03].

Pour faire face à l'attaque (2), les auteurs de [Caronni03] proposent de produire des répliques "personnalisées" pour chaque serveur de réplication, en les attachant à l'identité du serveur par exemple. Dans cette solution, aucun serveur ne pourrait expédier sa copie à quelqu'un d'autre sans passer d'abord par le client d'origine. Ceci contredit le but même de la réplication qui est d'éviter des points centraux de défaillances et des goulots d'étranglement.

Une solution simple et efficace est d'employer un MAC où la clef est dérivée des entrées aléatoires du client et du serveur. De cette façon, ni le client ni le serveur ne peut, a priori, forcer un résultat avec des paramètres spécifiques. Par conséquent, le serveur ne peut pas expédier la requête de vérification car une tentative de délégation créerait une clef différente.

L'accord initial sur la clef peut être maintenu simple parce qu'il n'y pas de problème de confidentialité ou d'authenticité ; une attaque par homme-au-milieu (man-in-the-middle) permet au pire des cas de persuader le client que le serveur est non coopératif, une accusation qui peut être réalisée en supprimant simplement les paquets requête/réponse. Il faut noter que dans les MANETs une attaque par homme-au-milieu n'est pas si évidente, vu leurs caractéristiques intrinsèques (ex. mobilité des nœuds).

Nous proposons un protocole simple pour l'échange d'une clé entre deux nœuds (le nœud *A* et le nœud *B*) :

(i) M1 : A->B : A, R_1 ; R_1 nombre aléatoire généré par le nœud *A*

(ii) M2 : B->A : B, R_2 ; R_2 nombre aléatoire généré par le nœud *B*

(iii) La clé du MAC est k = $H(A||B||R_1||R_2)$

[9] Le temps de calcul du hach est négligeable par rapport au temps d'acheminement des paquets.

Pour faire face à l'attaque (4), les auteurs de [Caronni03] proposent une procédure qui permet de différencier entre un serveur honnête, qui accède aux données à partir de son propre disque, et un serveur malhonnête, qui accède aux données à partir du réseau.

L'idée de cette procédure est la suivante : demander le hach de plusieurs blocs. Les positions des blocs sont sélectionnées comme suit : la position du 1er bloc est fixée par le client dans la requête, la position du bloc suivant (deuxième bloc) est calculée en fonction du hach du 1er bloc (la fonction de déplacement est connue par tous les nœuds) et ainsi de suite. En résumé, la position du bloc *i* dépend du hach du bloc *i* - 1. La procédure détaillée est présentée dans [Caronni03]. Avec cette procédure, si le serveur stocke le fichier localement, il aura besoin d'environ 1s pour vérifier 30 Mb de données. Mais s'il essaie de télécharger les blocs à la demande, il aura besoin de 30s pour vérifier la même quantité d'information.

Dans [Caronni03], les auteurs discutent l'attaque (3) mais ne présentent aucune solution. Ils affirment que c'est difficile de faire face à cette attaque. Dans notre cas, nous voyons les choses d'une autre manière. Premièrement, le serveur de réplication n'abuse pas d'un nœud honnête mais plutôt d'un nœud complice lequel est au courant et est d'accord pour l'exploitation de ses ressources. Deuxièmement, nous pouvons simplement considérer que le nœud serveur et le nœud complice constituent une seul entité représentée par le nœud serveur. Ensuite, si pour une raison quelconque cette entité ne répond pas correctement aux requêtes de vérification et/ou de lecture/écriture alors c'est le nœud serveur qui assumera les conséquences. Troisièmement, un nœud client n'a pas à chercher où un nœud serveur stocke ses données du moment qu'il n'abuse pas d'un autre nœud honnête. Quatrièmement, le seul effet négatif de cette attaque est un délai additionnel dans le temps de réponse aux différentes requêtes. Pour toutes ces raisons nous considérons que la collusion n'a pas d'effet négatif significatif sur les performances du protocole de réplication et que finalement ce n'est pas vraiment une attaque à laquelle il faut faire face à tout prix.

4.3.2. L'estimateur de l'état du réseau

Dans un réseau ad-hoc, le comportement d'un nœud est grandement influencé par le scénario de déploiement. Des facteurs comme la position et la densité peuvent influencer le comportement d'un nœud.

Le rôle principal de notre estimateur d'état du réseau est de fournir des indications sur l'état du réseau (les conditions réseau) pour pouvoir adapter le comportement de PARDON par rapport aux conditions réseau.

Nous pouvons définir trois types d'état du réseau :

- L'état local : qui indique les conditions réseau d'un nœud ;
- L'état du voisinage : qui indique les conditions réseau du voisinage d'un nœud ;
- L'état global : qui indique les conditions réseau de tous les nœuds du réseau.

Il n'y a rien de définitif qui indique les conditions réseau des autres nœuds du réseau [Refaei07]. Cependant, il y a des observations internes qui pourraient indiquer approximativement l'état réseau des autres nœuds. Par exemple, la taille actuelle de la file d'attente, des statistiques

au niveau de la couche réseau (nombre de paquets reçus, expédiés, supprimés), des statistiques au niveau de la couche liaison de données (nombre de trames reçues, retransmises, supprimées), la valeur courante du temporisateur aléatoire (*backoff timer*) et le nombre de collisions peuvent indiquer partiellement les conditions réseau du nœud. Si cette information (les statistiques) est maintenue par les nœuds voisins, elle peut également servir comme indication de l'état du voisinage.

L'estimation de l'état global ou de l'état du voisinage nécessite de rassembler et de combiner les informations sur les conditions réseau de tous les nœuds (ou les nœuds voisins) ce qui est couteux et souvent impraticable.

Dans notre cas, nous considérons seulement les conditions réseau locales. Basé sur l'estimation de l'état du réseau local, PARDON peut s'adapter aux conditions réseau pour augmenter l'exactitude de l'évaluation de comportement.

4.3.3. Le gestionnaire de réputation

4.3.3.1. La réputation de service

***Définition** : La réputation de service (RS) est définie comme étant l'opinion constituée par un nœud i au sujet du comportement d'un nœud j en tant qu'acteur dans le système de base, c.-à-d. si le nœud j participe correctement dans le protocole de réplication.*

4.3.3.2. Représentation de la réputation

Puisque la réputation agrège essentiellement des expériences antérieures et évolue dynamiquement, elle est similaire à l'analyse bayesienne, qui est un procédé statistique qui estime des paramètres d'une distribution basée sur des observations. Commençant par une distribution a priori (antérieure), qui est l'état initial avant que n'importe quelle observation soit faite, l'analyse bayesienne tient compte continuellement de nouvelles expériences et dérive la probabilité postérieure [Liu06]. Une distribution intensivement utilisée dans l'analyse bayesienne est la distribution beta.

Un nœud *i* suppose qu'il y a un paramètre θ tel qu'un nœud *j* se conduit mal (ne coopère pas) avec une probabilité θ. Ce paramètre est le résultat des observations indépendantes effectuées par le nœud *i* à l'égard du nœud *j* (le nœud *i* pense qu'il y a un paramètre θ différent pour chaque nœud *j*, et chaque nœud *i* peut croire dans différents paramètres θ ; donc le paramètre θ devrait être indexé par *i* et *j*, mais pour des raisons de simplicité nous omettons les index ici). Les paramètres θ sont inconnus, un nœud *i* modélise cette incertitude en supposant que θ lui-même suit une distribution a priori qui est mise à jour par les nouvelles observations. Cette approche constitue l'approche bayesienne standard. Nous utilisons une approche bayesienne modifiée qui utilise comme distribution a priori la distribution beta(α,β).

a) La distribution beta

Selon la théorie des probabilités, la probabilité postérieure pour des événements binaires peut être estimée par la distribution beta. Par exemple, étant donné un processus avec deux résultats possibles (T, ¬T), soient r et s le nombre observé de T et de ¬T respectivement. La fonction de

densité de probabilité (fdp) de la probabilité p d'avoir le résultat T pour la prochaine fois peut être donnée par la distribution beta (avec α = r + 1 et β = s + 1) :

$$f(p \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1}(1-p)^{\beta-1}, \text{ avec } 0 \le p \le 1 \, et \, \alpha, \beta \ge 0$$

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt$$

B(α,β) est la fonction beta et f(p| α,β) représente la fonction de densité de probabilité.

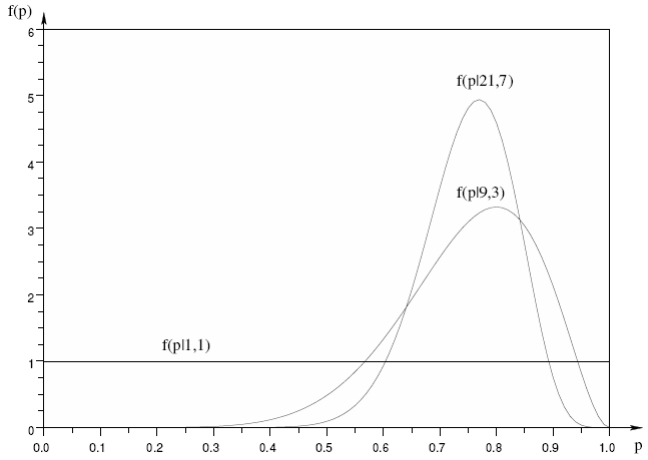


Figure 4.6. Exemples de distribution beta

L'espérance de la distribution beta f(p| α,β) est :

$$E(p) = \frac{\alpha}{\alpha + \beta}$$

Elle représente la valeur moyenne de p basée sur (α + β - 2) observations. Par exemple, dans la figure 4.6 la valeur moyenne des deux f(p|9, 3) et f(p|21, 7) est 0.75. Elle peut être interprétée comme suit : « la probabilité d'observer le résultat T à l'avenir est incertaine, mais la valeur prévue est 0.75 ». En outre, f(p|21, 7) a plus de certitude, (c.-à-d., f(0.75|21, 7) > f(0.75|9, 3)), grâce aux observations accumulées.

La procédure bayesienne standard est comme suit. Initialement, la distribution a priori est beta(1,1). Elle correspond à la distribution uniforme sur [0,1] ; ceci représente l'absence d'information sur la valeur du paramètre θ. Quand une nouvelle observation est faite, soient n expériences négatives (comportement non coopératif) et p expériences positives (comportement coopératif), la distribution a priori est mise à jour selon α := α + p et β := β + n. Si le paramètre θ est constant, alors après un nombre *m* important d'observations, α ~ mθ, et β ~ m(1 - θ) (dans l'espérance). L'avantage d'utiliser la distribution beta est qu'elle a besoin seulement de deux paramètres qui sont continuellement mis à jour pendant que des observations sont faites ou rapportées (voir figure 4.7).

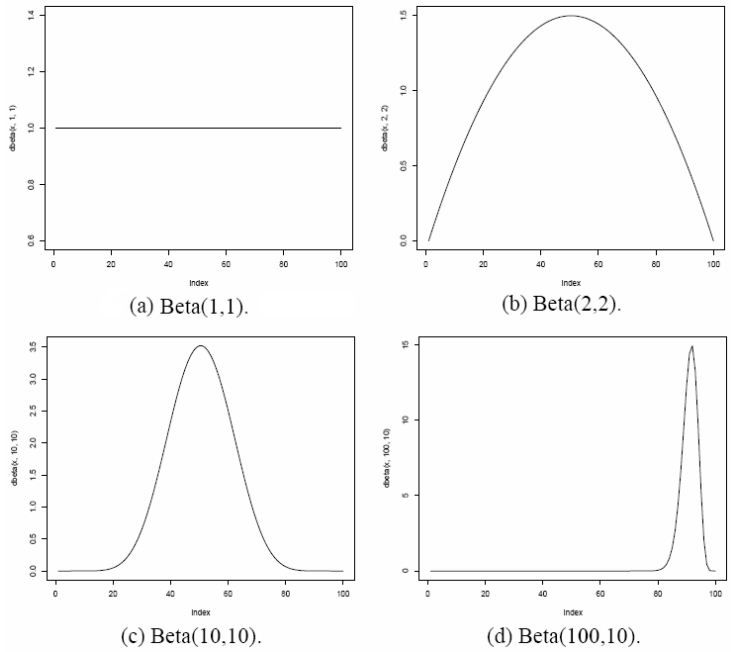


Figure 4.7. Densité de la fonction Beta

b) Modélisation de la réputation avec une distribution Beta

La réputation est essentiellement une estimation a posteriori basée sur un historique d'expériences (expériences directes ou indirectes). La distribution beta a été reconnue comme un modèle utile pour modéliser la réputation [Buchegger04, Liu06, Quercia06, Jøsang02, Whitby04 et Marias06]. Par conséquent, nous représentons la réputation dans notre mécanisme par une distribution beta (abrégée en tant que réputation beta ou simplement réputation). La valeur de réputation est représentée par un couple (α, β) (α, β $\geq$ 1), où α et β représentent les expériences positives et négatives, respectivement.

Grâce aux propriétés statistiques de la distribution beta, la réputation beta offre les avantages suivants :

- Il est facile d'évaluer la réputation d'un nœud en calculant simplement l'espérance de la distribution beta f(p| α,β)

$$E(p) = \frac{\alpha}{\alpha + \beta}$$

- Il est facile d'évaluer combien d'expériences (c.-à-d., α + β - 2) ont contribuées à la réputation courante. Plus cette valeur est grande, plus la réputation calculée assumera la valeur prévue plus probablement. Seulement la réputation des nouveaux nœuds est basée sur 0 expériences.
- Elle facilite la combinaison des expériences des sources multiples, y compris du nœud lui-même et des différents nœuds recommandeurs. Par exemple, étant donné deux recommandations f(p| α_1,β_1) et f(p| α_2,β_2), la combinaison des deux est f(p|α_1,β_1) + f(p|α_2,β_2) = f(p|α_1+α_2 ,β_1+β_2) si elles sont considérées de même importance.

- Elle reflète la nature de la réputation, qui est l'agrégation des observations. Une entité ajuste dynamiquement la réputation en fonction des expériences accumulées, qui est semblable à dériver la distribution postérieure après que des observations soient faites.
- Elle capture l'incertitude de la réputation. La distribution beta donne seulement la fonction de densité de probabilité (fdp) de la probabilité d'avoir un résultat, qui assortit le fait que la réputation donne seulement l'évaluation probabiliste du futur comportement d'une entité.

4.3.3.3. Formation de la réputation

Nous modélisons la réputation de service (RS) d'un nœud avec une distribution beta(α,β). Initialement α = 1 et β = 1. Chaque nœud client stocke la réputation de service des nœuds serveurs avec lesquels il est en interaction ou a eu une interaction. Les réputations de service des nœuds sont stockées dans une table de réputation de service (TRS) gérée par le GR. La figure ci-dessous représente une entrée de cette table.

ID nœud	RS		
unID	S_p	S_n	T_s

Figure 4.8. Une entrée dans la TRS

Où S_p (représente α) : le nombre d'expériences positives
S_n (représente β) : le nombre d'expériences négatives
T_s : le temps de la dernière mise à jour

Si la table est trop petite pour stocker toutes les RS des entités que le nœud a rencontrées, une certaine politique de remplacement est appliquée. Par exemple, les entrées sont supprimées selon leurs âges.

4.3.3.4. Calcul de la réputation

a) La mise à jour de S_p et S_n

Lorsque le gestionnaire de réputation (GR) reçoit une notification de la part du moniteur à propos un nœud, il traite cette notification en mettant à jour S_p et S_n du nœud signalé dans la notification.

Durant une période On le moniteur met à jour S_p et S_n selon le type d'évènement. Nous définissons trois types d'évènement, à savoir :

(1) réponse positive à une requête de vérification/de lecture ; dans ce cas $S_p = S_p + 1$
(2) réponse négative à une requête de vérification ; dans ce cas $S_n = S_n + 1$
(3) timeout pour une requête de vérification/de lecture ; dans ce cas $S_n = S_n + 1$

b) La réputation de service locale

Un nœud *i* calcule la réputation de service locale d'un nœud *j* (notée $RSL_i(j)$) en se basant exclusivement sur ses expériences directes avec le nœud *j*. Si les expériences directes du nœud *i* avec le nœud *j* sont significatives, c.-à-d., le nombre d'expériences directes accumulées est assez grand pour dériver une décision (ceci peut être jugé en vérifiant si toutes les ($S_p + S_n - 2$)

expériences accumulées atteignent un certain seuil), le nœud *i* utilise seulement la RSL calculée pour prendre des décisions à l'égard du nœud *j* dans le future.

$$\mathrm{RSL}_i(\mathrm{j}) = \frac{Sp_i(j)}{Sp_i(j) + Sn_i(j)}$$

c) La réputation de service globale

Si les expériences directes d'un nœud *i* avec un nœud *j* ne sont pas significatives, c.-à-d., le nombre d'expériences directes accumulées n'est pas assez grand pour dériver une décision, le nœud *i* demande des recommandations, à propos du nœud *j,* des autres nœuds. Les recommandations reçues par le nœud *i* et ses expériences directes sont alors combinées pour calculer la réputation de service globale du nœud *j* (notée $RSG_i(j)$).

Le gestionnaire de recommandation (GC) filtre les recommandations reçues et fournit au GR les recommandations jugées vraies. À chaque recommandation *r* et attribuée un poids w_r. Les poids de différentes recommandations sont normalisés par la division sur la somme de tous les poids. Soit R l'ensemble des recommandations reçu par le GR, la RSG est calculée comme suit :

$$RSG = \lambda \times RSL + (1-\lambda) \times \frac{\sum_{r \in R} (r \times w_r)}{\sum_{r \in R} w_r}$$

Où λ est le poids donné à son expérience propre directe (RSL). Ce poids est généralement supérieur à 0.5. Attribuer plus de poids à sa propre expérience s'explique par le fait qu'un nœud accorde plus de confiance à ses propres expériences et observations qu'à celles des autres nœuds.

4.3.3.5. Evolution de la réputation

Une entité peut changer son comportement avec le temps, faisant que les anciennes expériences deviennent non pertinentes pour l'évaluation réelle de la réputation [Jøsang02, Liu06]. L'approche bayesienne standard donne le même poids à chaque observation indépendamment de son âge (sa date d'occurrence). Comme nous voulons donner moins d'importance au comportement passé, nous avons donc développé une approche bayesienne de mise à jour modifiée en introduisant un taux d'affaiblissement au comportement passé. Ce qui permet une sorte de rachat et empêche en même temps un nœud de se conduire mal en profitant d'une bonne réputation acquise dans le passé.

Dans l'approche bayesienne modifiée, nous définissons deux types de mises à jour : (i) mise à jour suite à une observation et (ii) mise à jour périodique.

a) Mise à jour suite à une observation

Le comportement récent et le comportement passé participent tous les deux au calcul de la réputation. Leurs poids assignés décident comment la réputation évoluera dans le temps. Par exemple, si on décide d'attribuer un poids plus important au comportement récent, la réputation d'un nœud chutera rapidement après juste quelques expériences négatives. Nous assignons donc plus de poids au comportement récent.

A la réception d'une nouvelle observation (p,n), le GR mis à jour la RSL comme suit :

$$\begin{cases} \alpha^{new} = \phi_e \alpha + p \\ \beta^{new} = \varphi_e \beta + n \end{cases} \quad où \;\; \alpha^{new} et\; \beta^{new} \geq 1,\; 0 \leq \phi_e \leq 1 \text{ et } 0 \leq \varphi_e \leq 1$$

Où α^{new} et β^{new} désignent les nouvelles valeurs de α et β, respectivement. ϕ_e et φ_e représentent le taux d'affaiblissement des expériences positives et négatives, respectivement.

Le taux d'affaiblissement détermine la vitesse avec laquelle l'historique est ignoré. Si le taux d'affaiblissement est égal à 0 alors l'historique sera immédiatement oublié ; tandis que quand le taux est égal à 1, l'historique sera gardé pour toujours et les expériences seront considérées de même importance indépendamment de leurs âges.

b) Mise à jour périodique

En plus de la mise à jour suite à une nouvelle observation, nous mettons à jour la RSL périodiquement. Chaque nœud mis à jour périodiquement les RSL stockées dans sa TRS, ceci permet une sorte de rachat même en l'absence d'observations et empêche en même temps un nœud de se conduire mal en profitant d'une bonne réputation acquise dans le passé.

$$\begin{cases} \alpha^{new} = \phi_p \alpha \\ \beta^{new} = \varphi_p \beta \end{cases} \quad où \;\; \alpha^{new} et\; \beta^{new} \geq 1,\; 0 \leq \phi_p \leq 1 \text{ et } 0 \leq \varphi_p \leq 1$$

Où α^{new} et β^{new} désignent les nouvelles valeurs de α et β, respectivement. ϕ_p et φ_p représentent le taux d'affaiblissement des expériences positives et négatives, respectivement.

Les taux d'affaiblissement définis permettent de définir trois stratégies, à savoir :

- Une stratégie pessimiste : dans cette stratégie un nœud oublie l'historique positif (historique des expériences positives) plus rapidement que l'historique négatif. Cette stratégie est définie par : $\phi_e < \varphi_e$ et $\phi_p < \varphi_p$
- Une stratégie optimiste : dans cette stratégie un nœud oublie l'historique négatif (historique des expériences négatives) plus rapidement que l'historique positif. Cette stratégie est définie par : $\phi_e > \varphi_e$ et $\phi_p > \varphi_p$
- Une stratégie impartiale : dans cette stratégie un nœud oublie les deux historiques négatif et positif avec la même vitesse. Cette stratégie est définie par : $\phi_e = \varphi_e$ et $\phi_p = \varphi_p$

4.3.3.6. Classification

Notre module de décision utilise une politique de classification simple basée sur une classification binaire à base d'un seuil de coopération prédéfinit (voir figure 4.9). Dans la classification, nous utilisons la RSL si les expériences directes accumulées sont significatives. Dans le cas contraire nous utilisons la RSG.

La politique de classification proposée est optimiste dans la mesure où un nœud n'est considéré non coopératif que si nous avons suffisamment de preuves (les expériences directes accumulées sont significatives) qui indiquent un comportement non coopératif. Dans le cas contraire, le nœud est considéré coopératif.

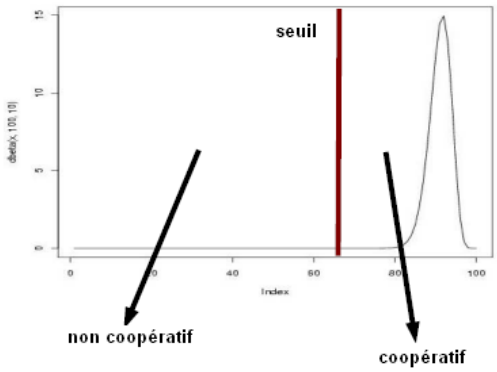


Figure 4.9. Le processus de classification

Le seuil en question est un paramètre local défini par chaque nœud localement. Il n'est pas fixe ; il dépend des exigences du nœud en matière de sécurité (par exemple, pour un seuil égal à 0.65, le nœud exige un niveau de coopération supérieur ou égal à 65%) ; il peut être ajusté, par exemple, en fonction de l'importance des données à répliquer.

Un nœud *j* classifie un nœud *i* en utilisant l'algorithme ci-dessous.

Algorithme 3 Classification

```
1:  SI (les expériences directes accumulées avec le nœud i sont significatives) Alors
2 :       SI RSL(i) < seuil de coopération Alors
3 :             i est non coopératif
4 :       FSI
5 :  SINON
6 :             i est coopératif
7 :  FSI
```

4.3.4. Le gestionnaire de recommandation

4.3.4.1. La collecte des recommandations

Si le GR ne dispose pas de suffisamment d'information, il sollicite le gestionnaire des recommandations (GC) pour avoir des informations de réputation globales. Le GC envoie une requête de demande d'avis à propos d'un nœud ou d'un ensemble de nœud (requête de recommandation) à un ensemble de nœuds.

Le premier problème qui se pose est : comment atteindre les nœuds qui ont déjà eu des expériences avec un nœud donné ? C'est-à-dire, comment diffuser la requête de recommandation ?

Un service de réplication à généralement la particularité d'être *indépendant de la proximité géographique*[10], c'est-à-dire, deux nœuds voisins n'ont pas nécessairement une relation client-serveur entre eux ou avec le même nœud serveur. Au contraire, les protocoles de réplication essayent souvent de répliquer les données dans des régions différentes dans le réseau pour faire

[10] C'est un concept que nous définissons nous-mêmes.

face au phénomène de partitionnement réseau. Donc on a souvent des relations client-serveur entre des nœuds non voisins.

Nous avons identifié trois solutions possibles à notre premier problème :

- ***solution localisée*** : la solution localisée consiste à diffuser la requête de recommandation dans le voisinage direct du nœud. Cette solution n'est pas efficace dans notre cas, car nous avons vu que les nœuds voisins non pas nécessairement une relation client-serveur avec les même nœuds serveurs.
- ***solution itérative*** : la solution itérative consiste à diffuser la requête de recommandation de manière itérative. C'est-à-dire diffuser la requête de recommandation dans le voisinage direct du nœud (les nœuds voisins à un saut), et si aucune information de réputation n'est disponible (aucune réponse ou réponse insuffisante) la requête de recommandation est rediffusée dans le voisinage à deux sauts (les nœuds voisins à deux sauts) et ainsi de suite. Mais, telle que prouvé par l'expérience [Lee03], une solution itérative est souvent plus coûteuse qu'une solution globale en terme de ressources (consommation en énergie, bande passante utilisée et une latence plus importante).
- ***solution globale*** : la solution globale consiste à diffuser la requête de recommandation dans tout le réseau. Si une relation client-serveur entre un nœud client quelconque (ou plusieurs nœuds client) et notre nœud serveur existe alors nous sommes certain d'atteindre ce nœud client pour récupérer des informations de réputation sur notre nœud serveur.

Parmi les trois solutions que nous avons discutées, concentrons-nous maintenant sur la solution globale puisque elle est la plus pratique. Nous distinguons deux approches possibles pour l'emploi de cette solution :

- Approche proactive : lors d'une interaction entre deux nœuds (un serveur et un client) le nœud client signale aux autres nœuds dans le réseau qu'il dispose d'une information sur la réputation du nœud serveur. Le nœud client peut diffuser la réputation du nœud serveur ou signaler simplement la disponibilité de cette information et répond par la suite aux requêtes de recommandations des nœuds intéressés.
- Approche réactive : dans cette approche, lorsqu'un client décide d'interagir avec un serveur et il ne dispose pas de suffisamment d'information sur la réputation de ce dernier, il diffuse une requête de recommandation dans le réseau pour avoir l'avis des autres nœuds à propos du nœud serveur. Le client collecte par la suite les réponses reçues pour mettre à jour ses informations de réputation concernant le nœud serveur.

L'approche proactive génère un trafic supplémentaire (overhead) important, et l'information diffusée n'est pas nécessairement utile pour les autres nœuds. L'approche réactive se révèle plus adaptée dans notre cas. Le fait que l'approche réactive découvre l'information de réputation quand le besoin se fait ressentir permet d'éviter de réserver constamment une partie des communications pour diffuser des informations de réputation qui ne sont pas toujours utiles.

La solution globale réactive est intéressante, mais elle reste coûteuse en terme de ressources. L'idéal serait d'avoir une solution équivalente en terme d'efficacité et moins coûteuse en terme de ressources. Pratiquement, nous essayons de trouver un compromis entre :

- Une diffusion totale d'une requête dans le réseau, ce qui est coûteux en terme de ressources et pose le problème du passage à l'échelle. D'un autre côté, cela permet d'atteindre tous les nœuds qui ont déjà une expérience avec le(s) nœud(s) ciblé(s), si ces nœuds existent, bien sûr.
- Une diffusion partielle d'une requête à un ensemble restreint de nœuds, ce qui n'est pas coûteux en termes de ressources, mais peut ne pas atteindre les nœuds qui ont déjà une expérience avec le(s) nœud(s) ciblé(s).

Nous remarquons que, nous n'avons pas besoin d'atteindre tous les nœuds du réseau mais plutôt la majorité des nœuds. Donc, une diffusion contrôlée en utilisant, par exemple, le TTL est suffisante. Cependant, cette solution n'est pas très efficace dans notre cas car elle est liée à la proximité géographique (la requête diffusée couvre une zone géographique limitée à proximité du nœud demandeur).

Le compromis que nous proposons repose sur l'utilisation d'un *algorithme de dissémination épidémique.* Celui-ci est peu coûteux en ressources et couvre une très grande partie du réseau de manière aléatoire, ce qui permet d'éviter le problème de proximité géographique.

a) La diffusion dans les MANETs

Un algorithme de diffusion a deux paramètres : permettre de joindre l'ensemble des nœuds du réseau tout en minimisant le nombre d'émissions nécessaires pour une telle opération. Ces deux paramètres sont liés et un bon algorithme doit être capable de trouver le juste équilibre entre les deux en fonction de l'environnement et de l'objectif à atteindre.

L'algorithme le plus trivial pour diffuser un message est le protocole d'inondation (*blind flooding*). Son fonctionnement est simple : chaque nœud qui reçoit le message d'inondation pour la première fois le réémet à ses voisins.

Bien qu'il ne nécessite qu'une table pour mémoriser les identifiants des messages d'inondation, l'algorithme d'inondation reste un algorithme coûteux. Malgré sa simplicité, il impose une charge énorme au réseau, car chaque voisin se doit de renvoyer le message. Ce grand nombre de mobiles essayant d'accéder en même temps, entraîne une probabilité importante de collisions parmi les rediffusions. Lorsque, deux mobiles très proches se décident à réémettre le même message, une grande partie des voisins de ces deux mobiles vont recevoir exactement la même information. Cette utilisation doublon est pourtant inutile. De plus, tous les voisins vont essayer de réémettre le message, entraînant ainsi des contentions pour l'accès au médium.

Ce problème, dit de la tempête de messages de diffusion (Broadcast Storm Problem), a été traité en profondeur par Ni et al. [Ni99]. Une des solutions proposée est la diffusion probabiliste (*Probabilistic Scheme*). Lorsqu'un mobile reçoit un message pour la première fois, il le réémet avec une probabilité P, fixée au départ [Ni99].

D'autres solutions combinent la réémission probabiliste avec un mécanisme additionnel, tel que :

- *le schéma fondé sur le comptage (Counter-Based Scheme)* [Cartigny03] : un mobile ne réémet pas un message s'il l'a déjà entendu plus de *C* fois.
- *le schéma fondé sur la distance (Distance-Based Scheme)* [Cartigny03] : un mobile ne réémet pas un message s'il le reçoit d'un mobile situé à une distance inférieure à *D*.
- *le schéma fondé sur la position (Location-Based Scheme)* [Cartigny03] : Un mobile ne réémet pas un message si la couverture supplémentaire par l'envoi de son message est inférieure à un seuil *A*.

D'autres protocoles, utilisent des combinaisons de ces schémas pour déterminer s'ils devraient rediffuser le message ou non ; toutes ces solutions souffrent d'une latence considérable.

Le mode probabiliste n'exploite aucune information sur la topologie du réseau, car la probabilité est fixée et ne varie pas en fonction de la configuration du voisinage, des changements dans le réseau et des messages déjà reçus. Pourtant, il est logique de faire varier la probabilité en fonction de la densité du réseau. En effet, la probabilité pour joindre un certain pourcentage de nœuds dépend de la densité moyenne du réseau. Plus exactement, le seuil de probabilité pour contacter une partie des nœuds est inversement proportionnel à la densité globale du réseau. En développant cette idée, un protocole performant a été proposé dans [Drabkin07].

Nous proposons un nouveau protocole, inspiré de [Drabkin07], pour la diffusion des requêtes de recommandations.

b) L'algorithme de diffusion des requêtes de recommandations

Nous utilisons la primitive *prob_bcast* qui permet de diffuser un message au voisinage direct de l'expéditeur avec une probabilité donnée. Nous employons également une file d'attente *cast_queue*, avec la méthode *add* qui accepte comme paramètres la probabilité de diffusion, le temps, le message lui-même et le type du message.

L'algorithme 4 représente le pseudo code de notre protocole pour chaque processus (nœud) P_i.

Description de l'algorithme :

La diffusion d'une requête de recommandation dans notre protocole se résume aux étapes suivantes :

1. Etape 1 (lignes 1-3) : l'émetteur P_i du message m, diffuse le message *m||header(m)* dans le voisinage direct (l'ensemble N(p)) avec une probabilité égale à 1. La partie header(m) constitue l'entête du message m, elle contient l'identifiant de la source et le numéro de séquence associé.
2. Etape 2 (lignes 4-10) : lors de la réception d'un message *m* pour la première fois, P_i accepte *m.* Si P_i est un voisin direct de l'émetteur alors il planifie une rediffusion de m (ligne 7) avec une probabilité égale à 1 et une durée d'attente aléatoire. Cette mesure permet d'éviter le problème de l'extension prématurée [Haas02] de la diffusion si la source

a relativement peu de voisins. Si P_i n'est pas un voisin direct de l'émetteur, il planifie une rediffusion de m (ligne 9) avec une probabilité *p* et une durée d'attente aléatoire.

$$p = \min\left(1, \frac{\beta}{|N(p_i)|}\right)$$

Où : $|N(p_i)|$ est le nombre de nœuds voisins directs du nœud p_i, et β est le facteur de fiabilité [Drabkin07] de l'algorithme.

3. Etape 3 (lignes 11-13) : si P_i reçoit un message *m* qu'il à déjà reçu, le message est ignoré et P_i annule sa retransmission.
4. Etape 4 (lignes 14-17) : lors de l'expiration d'un timer d'un message, le message est rediffusé.

Algorithme 4 La dissémination épidémique

```
Const : short_jitter
Var :  cast_queue

Lors de broadcast(msg) par l'application
    Faire
        (1)   header := msg_id||node_id; /* l'entête du msg=id-nœud (source) + n° séquentiel */
        (2)   data_msg := header||msg;
        (3)   prob_bcast(prob = 1, data_msg, DATA); /* broadcast local */
    Fait

Lors de receive(msg, DATA) depuis Pj
    Faire
        (4)   si (msg non reçu auparavant) alors
        (5)   accept(Pi, msg);       /*envoyer le msg à l'application*/
        (6)     si (je suis un voisin direct de l'émetteur) alors
        (7)       cast_queue.add(prob = 1,time=random(0, short_jitter), msg, DATA);
        (8)     sinon
        (9)       cast_queue.add(prob = min(1, β/|N(pi)|),time=random(0, short_jitter), msg, DATA);
        (10)    fsi
        (11)  sinon /*  msg déja recu  */
        (12)  cast_queue.remove(msg);
        (13)  fsi
    Fait

Lors d'expiration d'un timer d'un msg dans la file
    Faire
        (14)  cast_queue.remove(msg);
        (15)  pr = la probabilité attachée au msg;
        (16)  type = le type de msg;
        (17)  prob_bcast(prob = pr, msg, type);
    Fait
```

Un problème qui doit être considéré est la gestion des messages reçus, nous utilisons une approche simple pour la gestion de l'espace de stockage utilisé pour les messages reçus. L'approche est basée sur la suppression des messages selon leurs temps d'expiration (timer-based). Dans ce cas, il faut trouver un compromis pour déterminer le temps d'expiration des

messages : un temps important augmente la fiabilité, mais augmente également la consommation de mémoire. Il s'avère que même avec un temps d'expiration relativement courts nous pouvons atteindre une fiabilité satisfaisante [Drabkin07].

4.3.4.2. Le filtrage des recommandations

La qualité d'un système de réputation dépend de l'intégrité des recommandations qu'il reçoit. Un problème fondamental est qu'un nœud peut surestimer ou sous-estimer la réputation d'un autre nœud. Quand les recommandations sont diffusées, il est a priori impossible de savoir qu'un nœud fournit des recommandations injustes. La protection efficace contre des recommandations injustes est une condition de base pour un système de réputation robuste.

a) Les méthodes de filtrage des recommandations

Les méthodes de filtrage des recommandations peuvent être classées en deux catégories : endogènes et exogènes [Whitby04].

- Méthodes endogènes : cette catégorie couvre les méthodes qui excluent ou donnent un poids faible aux recommandations injustes en se basant sur l'analyse et la comparaison des recommandations elles-mêmes. L'hypothèse est que les recommandations injustes peuvent être identifiées par leurs propriétés statistiques seulement. Les travaux dans cette catégorie incluent Dellarocas [Dellarocas00] et Chen et Singh [Chen01].
- Méthodes exogènes : cette catégorie couvre les méthodes où des facteurs externes, tels que la réputation du recommandeur, sont employés pour déterminer le poids donné aux recommandations. L'hypothèse est que les recommandeurs avec de basses réputations donnent probablement des recommandations injustes et vice-versa. Cette méthode exige que la réputation du recommandeur puisse être déterminée en incluant des facteurs externes autres que les recommandations elles-mêmes. Les propositions dans cette catégorie incluent Buchegger et Le Boudec [Buchegger04], Cornelli et al. [Cornelli02], Ekstrmm et Bjornson [Ekstrom02], Liu et Issarny [Liu06] et Yu et Singh [Yu03].

b) Discussion

Les méthodes exogènes peuvent être classées[11] en deux catégories : avec filtrage a priori et avec filtrage a posteriori.

- Avec filtrage a priori : dans ce cas, la réputation du recommandeur est déterminée durant le filtrage et avant l'utilisation des informations de réputation. Un exemple typique est l'algorithme proposé par Buchegger et Le Boudec [Buchegger04]. Ce type de filtrage est utilisé dans des systèmes proactifs[12] et repose sur la disponibilité de l'information de réputation locale qui sera utilisée comme valeur de référence dans le filtrage. Ce type d'algorithme n'est pas utilisable dans notre architecture ;
- Avec filtrage a posteriori : dans ce cas, la réputation du recommandeur est déterminée après l'utilisation des informations de réputation. Liu et Issarny [Liu06] proposent d'utiliser d'abord l'information de réputation reçue, pour ensuite juger l'honnêteté des

[11] C'est une classification qui nous est propre et qui est liée à notre compréhension.
[12] Diffusion proactive des informations de réputation.

recommandeurs. Ce type d'algorithme oblige le nœud récepteur des informations de réputation à utiliser cette information pour ensuite juger l'honnêteté des recommandeurs. Nous pensons que ce type d'algorithme est peu pratique de manière générale ;

Concernant les méthodes endogènes, une proposition endogène intéressante dans [Whitby04] repose sur l'hypothèse que les recommandations fournies par les différents recommandeurs honnêtes à propos d'un nœud donné suivront plus ou moins la même distribution de probabilité. Quand un nœud change son comportement, tous les recommandeurs honnêtes qui interagissent avec ce nœud changeront leurs recommandations en conséquence. Un algorithme centralisé basé sur cette idée a été proposé dans [Whitby04].

Avant de discuter notre proposition, intéressons nous à l'algorithme proposé dans [Whitby04] qui est l'un des algorithmes endogènes les plus référencés dans la littérature. L'algorithme repose sur l'hypothèse que la majorité des recommandations reçues provient des nœuds honnêtes.

Un nœud *i* collecte un ensemble de recommandations à propos d'un nœud cible *j*. Le principe est de vérifier que la réputation du nœud *j* (calculée à partir de la distribution beta globale) est comprise entre le *q* quantile (borne inférieure) et le (1 - *q*) quantile (borne supérieure) de la distribution beta(*r*(*x*)) pour chaque recommandeur *x*. Par quantile, nous entendons la fraction de points au-dessous d'une valeur donnée. C'est-à-dire, le 0.01 (1%) quantile est le point où 1% des données sont au-dessous et 99% des données sont au-dessus. La fdp de la figure 4.10 a un 0.01 quantile de 0.456 et un 0.99 quantile de 0.983.

Pour être significatif, le paramètre *q* (quantile) doit être dans l'intervalle [0.0, 0.5]. Plus la valeur de *q* est petite, moins il y aura de faux positifs (c.-à-d. moins de recommandeurs honnêtes seront exclus) et plus il y aura de faux négatifs (c.-à-d. plus de recommandeurs malhonnêtes seront inclus). La sensibilité de l'algorithme peut être augmentée ou diminuée en modifiant le paramètre *q*.

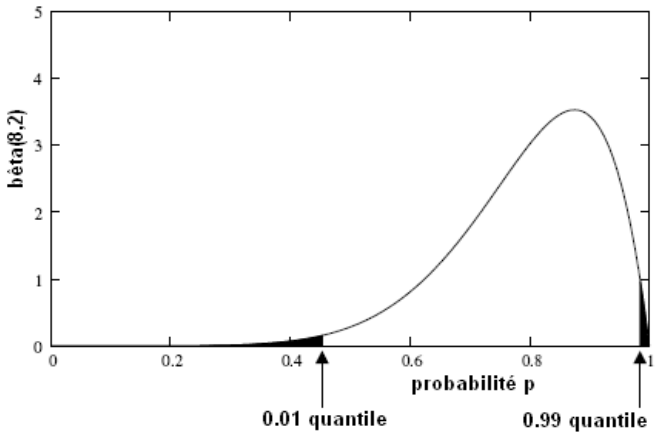


Figure 4.10. Les 1% et 99% quantiles de beta(8,2)

D'après notre analyse, cet algorithme pose plusieurs problèmes qui sont :

1) L'algorithme est gourmant en ressources CPU, ce qui le rend peu pratique dans les MANETs (le calcul des quantiles repose sur le calcul de la fonction de distribution cumulative inverse et emploi l'algorithme de Newton) ;

2) L'algorithme reste vulnérable à une attaque évidente : si un nœud *i* diffuse une recommandation avec des valeurs relativement grandes, la distribution globale calculée suivra automatiquement la distribution diffusée par le nœud *i* et toute recommandation qui ne suit pas celle de *i* sera considérée injuste.

3) Finalement, l'algorithme ne donne pas de bons résultats[13] même en l'absence de l'attaque décrite dans 2 ;

c) L'algorithme de filtrage proposé

Nous proposons un algorithme de filtrage hybride (approche endogène combinée avec une approche exogène). Nous présentons un algorithme distribué, qui combine l'idée discutée dans [Whitby04] avec une approche exogène pour exploiter l'historique des expériences de recommandations en vue d'améliorer la qualité du filtrage.

Suite à la diffusion d'une requête de recommandation par un nœud *i*, le nœud *i* collecte un ensemble de recommandations à propos d'un nœud cible *j*. Dans le cas général, quelques recommandations indiquent que *j* est un nœud coopératif, d'autres indiquent qu'il est non coopératif. Le nœud *i* calcule les deux scores associés aux deux types de recommandations en utilisant les réputations de recommandations (notée RR, voir 4.3.5.3) des nœuds recommandeurs. Le score le plus grand indique la recommandation majoritaire et celles incompatibles avec la recommandation majoritaire seront considérées injustes.

Description de l'algorithme :

Dans PARDON une recommandation prend la forme d'un vecteur

$$r = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \text{où } \alpha \geq 1 et \beta \geq 1$$

Notre algorithme de filtrage hybride est constitué des étapes suivantes :

- Etape 1 (lignes 1-2) : les nœuds considérés actif-malhonnête (voir section suivante) sont exclus, nous supposons que l'ensemble des recommandeurs considérés honnêtes est l'ensemble des recommandeurs D ;
- Etape 2 (lignes 3-8) : calcul des scores associés aux recommandations positives et négatives. Nous employons la réputation de recommandation RR (voir section suivante) ;
- Etape 3 (lignes 9-12) : si la recommandation du nœud *x* est incompatible avec la recommandation majoritaire, la recommandation sera considérée injuste. Donc *x* sera exclu de l'ensemble H ;
- Etape 4 (lignes 13-14) : les réputations de recommandations des nœuds recommandeurs sont mises à jour.

[13] Nous avons implémenté et testé l'algorithme avec des quantiles dans [0.2, 0.5].

Algorithme 5 Le filtrage hybride

Var : D : l'ensemble des recommandeurs
H : l'ensemble des recommandeurs considérés honnêtes
j : le nœud cible
sc = sn = 0
seuil, rsl

Initialisation
(1) D = D – {les nœuds considérés actif-malhonnête}
(2) H = D

Faire
Pour chaque recommandeur x dans H
(3) $rsl(x) = E(\text{beta}(\text{R}(\text{x}, \text{j}))$
(4) **si** $rsl(x) \geq seuil$ **alors**
(5) sc = sc + RR(x)
(6) **sinon**
(7) sn = sn + RR(x)
(8) **fsi**
Fait

Faire
Pour chaque recommandeur x dans H
(9) $rsl(x) = E(\text{beta}(\text{R}(\text{x}, \text{j}))$
(10) **si** ([sc > sn et $rsl(x) < seuil$] ou [sc ≤ sn et $rsl(x) \geq seuil$]) **alors**
(11) H = H – {x}
(12) **fsi**
Fait

(13) Mettre à jour négativement la réputation des nœuds {D – H}
(14) Mettre à jour positivement la réputation des nœuds {H}

4.3.4.3. Gestion de la réputation de recommandation

a) La réputation de recommandation

Définition : *La réputation de recommandation (RR) représente l'estimation de l'honnêteté qu'un nœud i attribue à un nœud j en tant qu'acteur dans le système de réputation (c.-à-d. si l'information locale publiée par le nœud j est susceptible d'être vraie).*

b) Modélisation

La RR est modélisée de la même façon que la RS. Nous modélisons la réputation de recommandation d'un nœud avec une distribution beta(α,β). Initialement α = 1 et β = 1. Chaque nœud stocke la RR des nœuds avec lesquels il est en interaction ou a eu une interaction. Les RRs des nœuds sont stockées dans une table de réputation de recommandation (TRR) gérée par le GC. La figure 4.11 représente une entrée de cette table.

ID nœud	RR		
unID	R_p	R_n	T_r

Figure 4.11. Une entrée dans la TRR

Où R_p (représente α) : le nombre d'expériences positives

R_n (représente β) : le nombre d'expériences négatives

T_r : le temps de la dernière mise à jour

Si la table est trop petite pour stocker toutes les RRs des entités que le nœud a sollicité, une certaine politique de remplacement sera appliquée, par exemple, les entrées seront supprimées selon leurs âges.

c) Evolution de la réputation

Le GC utilise la même approche bayesienne modifiée utilisée dans le GR. Nous utilisons les deux types de mises à jour : (i) mise à jour suite à une recommandation et (ii) mise à jour périodique.

Lorsque le gestionnaire de recommandation (GC) filtre les recommandations reçues, il met à jour la RR des nœuds recommandeurs comme suit :

(1) Si le nœud appartient à l'ensemble H, dans ce cas $R_p = R_p + 1$

(2) Si le nœud appartient à l'ensemble D-H, dans ce cas $R_n = R_n + 1$

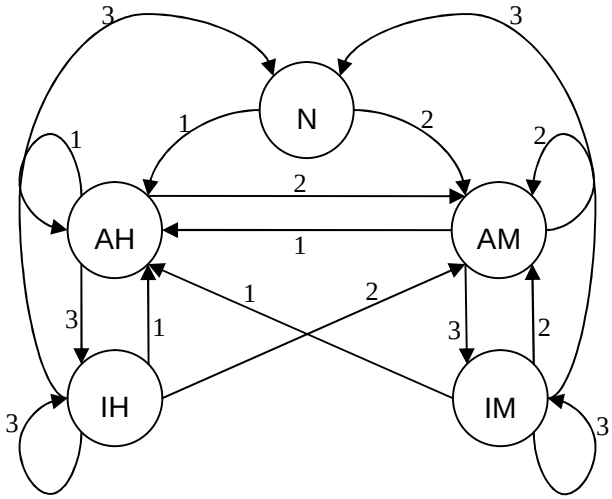
Un nœud *i* calcul la réputation de recommandation d'un nœud *j* (notée $RR_i(j)$) en se basant exclusivement sur ses expériences directes avec le nœud *j*.

$$\mathrm{RR_i(j)} = \frac{Rp_i(j)}{Rp_i(j) + Rn_i(j)}$$

Dans l'algorithme de filtrage, nous avons mentionné le poids d'une recommandation. La RR calculée pour un nœud *j* est le poids *w* qui sera attribué aux recommandations fournies par le nœud *j*.

d) Traitement des requêtes des recommandations

Basée sur les caractéristiques de la réputation beta, la valeur $R_p(j) + R_n(j) - 2$ sera grande si le nœud *j* est *actif* en fournissant des recommandations ; l'espérance de $f(R_p(j), R_n(j))$ sera grande si le nœud *j* est *honnête* en fournissant des recommandations justes. Ceci peut être employé pour identifier si un nœud est actif et honnête. Avec deux valeurs seuils λ_a (seuil d'activité) et λ_h (seuil d'honnêteté), un nœud recommandeur *j* sera considéré actif si $R_p(j) + R_n(j) - 2 > \lambda_a$ et inactif autrement. Il sera considéré honnête si $RR(j) > \lambda_h$ et malhonnête autrement. Nous pouvons donc définir 5 états possibles pour un nœud recommandeur : actif-honnête (AH), inactif-honnête (IH), actif-malhonnête (AM), inactif-malhonnête (IM) et nouveau (N). La figure 4.12 représente les différents états d'un recommandeur et les transitions possibles entre elles.



1 : fournit des recommandations justes

2 : fournit des recommandations injustes

3 : ne fournit pas de recommandations

Figure 4.12. Les états d'un recommandeur

Notons que l'état d'un recommandeur est un état local, un recommandeur actif et honnête peut être considéré inactif par un nœud *j* car il n'a pas eu une expérience directe avec le demandeur de recommandation (nœud *j*). Bien que l'inactivité puisse résulter du refus de recommander, elle n'implique pas nécessairement un comportement non coopératif, par exemple un nœud peut ne pas recevoir la requête de recommandation.

Du point de vue contribution dans PARDON, l'ordre d'utilité des différents recommandeurs est AH > IH > N > IM > AM. Afin de motivé un nœud pour devenir un AH, les nœuds qui sont plus utiles devraient également bénéficier plus des autres. Plus spécifiquement, les nœuds AH devraient avoir plus de chance d'identifier les nœuds non coopératifs en utilisant PARDON. Ceci est réalisé pendant la collecte des recommandations, où les requêtes de recommandations sont traitées différemment en termes d'accessibilité aux recommandations utiles.

Un nœud obtient des recommandations aléatoirement mais accepte seulement ceux des HA, HI, N et MI mais refuse ceux des MA. C'est pour renforcer le système contre la diffusion des rumeurs tout en ayant la possibilité d'identifier de nouveaux recommandeurs et de mettre à jour la RR des autres nœuds.

Quand un recommandeur *i* reçoit une requête de recommandation concernant un nœud *j*, *i* vérifie d'abord si ses expériences directes avec *j* sont assez significatives pour une recommandation. Si ce n'est pas le cas, *i* ignore tout simplement la requête de recommandation. Certains travaux (par exemple [Yu02] et [Mui02]) permettent une chaîne de recommandations, c.-à-d., un nœud peut recommander un recommandeur, qui peut renvoyer des recommandations ou en outre recommander un autre recommandeur, jusqu'à ce qu'une certaine limite soit atteinte. Cette pratique n'est pas incorporée dans PARDON parce qu'elle exige d'introduire la réputation de recommander un recommandeur (RRR) afin d'évaluer qu'une recommandation au sujet d'un

recommandeur est juste ou pas. La RRR est différente de la RR, parce qu'une entité peut être honnête en recommandant un recommandeur en donnant son RR dans la recommandation, mais peut être malhonnête en recommandant un fournisseur de service en ne donnant pas exactement son RS. Ceci nécessite la modélisation de la RRR, ce qui augmente considérablement la complexité du mécanisme de réputation. Par conséquent, un nœud dans PARDON ne recommande pas des recommandeurs.

Autrement, si *i* a assez d'expériences directes pour recommander, il traitera la requête de recommandation selon l'état du demandeur de recommandation (nœud *k*) :

- Si *k* est considéré AH, *i* répond immédiatement à la requête (la réponse est la $RSL_i(j)$) ;
- Si *k* est considéré AM, *i* ignore simplement la requête ;
- Si *k* est considéré IH ou N ou IM, *i* répond à la requête avec une probabilité *p* calculée comme suit :

$$p = \begin{cases} \dfrac{(R_p + R_n - 2)}{\lambda_a} + \varepsilon & Si \quad \text{k est considéré IH} \\ \max\left(prob\_mi, \dfrac{(R_p + R_n - 2)}{\lambda_a} \right) & Si \quad \text{k est considéré N} \\ \dfrac{(R_p + R_n - 2)}{\lambda_a} - \varepsilon & Si \quad \text{k est considéré IM} \end{cases}$$

Où ε est une petite valeur qui permet de différencier entre les IH et les IM. prob_mi est la probabilité minimale de répondre à un nœud considéré nouveau.

4.3.4.4. Attaques possibles

Un problème fondamental se pose dans la collecte des recommandations. Il s'agit de la protection des messages de recommandation. Un message réponse (réponse à une requête de recommandation) peut être modifié durant son transit entre les nœuds intermédiaires.

Comme première solution, nous supposons l'existence d'un protocole de routage sécurisé. Donc, le nœud demandeur et le nœud cible (celui qui reçoit la requête) ont nécessairement déjà une paire de clés partagée qui sera utilisée pour crypter les messages. Cette solution "lourde" permet d'assurer l'intégrité, la confidentialité des messages et l'authentification de l'émetteur, mais elle repose sur l'hypothèse de l'existence d'un protocole de routage sécurisé.

Nous proposons une solution plus légère où chaque nœud détient une paire de clés (Pk,Sk). La solution consiste à ce que le nœud demandeur envoie sa clé publique (Pk) avec le message de requête. Cette clé sera utilisée pour crypter les réponses. Seul le nœud demandeur peut décrypter les réponses puisque c'est lui seul qui détient la clé privée associée. Il faut noter que cette solution ne nécessite pas une autorité de certification (CA). Cette solution est simple et permet d'assurer l'intégrité et la confidentialité des messages mais pas l'authentification de l'émetteur.

4.3.5. Le gestionnaire de l'espace de stockage

Le gestionnaire de l'espace de stockage (GE) joue deux rôles qui sont :

- Implémenter les décisions prises par notre mécanisme ;
- Faire abstraction de la gestion de l'espace de stockage dédié au protocole de réplication, ce qui apporte une flexibilité à notre architecture.

Si un nœud est classé non coopératif, le mécanisme prendra certaines mesures. Nous classifions ces mesures en trois catégories :

- Réponses disciplinaires : cette catégorie comprend les mesures prises par le système pour punir d'une manière directe le nœud classé comme non coopératif ;
- Réponses informatives : cette catégorie englobe les mesures prises par le système pour prévenir les autres nœuds sur le nœud classé comme non coopératif ;
- Réponses correctives : cette catégorie renferme les mesures prises par le système pour corriger les dégâts qui peuvent être causés par un nœud classé comme non coopératif.

a) Réponses disciplinaires

Si un nœud *i* classe un nœud *j* comme étant non coopératif, le nœud *i* prendra les mesures suivantes :

- Ignore les requêtes de lecture/écriture provenant du nœud *j* ;
- Ignore les demandes de sauvegarde provenant du nœud *j* ;
- Supprime les données stockées du nœud *j*.

b) Réponses informatives

A la réception d'une requête de recommandation à propos d'un nœud *j* par un nœud *i*, le nœud *i* signale dans sa réponse la RSL du nœud *j.* Cette réponse constitue une sorte de punition indirecte car la réponse publiée signale le comportement non coopératif du nœud *j*.

c) Réponses correctives

Si un nœud *i* classe un nœud *j* comme étant non coopératif, et si le nœud *i* stocke des données relativement sensibles chez le nœud *j* alors *i* considèrera que ses données chez *j* sont perdues et peut donc les répliquer de nouveau si nécessaire.

4.4. Conclusion

Dans ce chapitre, nous avons détaillé l'architecture de notre mécanisme de coopération PARDON. Le mécanisme PARDON est un système de réputation globale, qui incorpore la notion de temps et de confiance, et gère les deux types de réputation.

Durant la conception de notre solution, nous avons adopté une démarche scientifique, où la problématique de départ a été divisée en plusieurs sous-problèmes. Pour chaque sous-problème, nous avons présenté et discuté plusieurs solutions possibles pour ensuite choisir la mieux adaptée à notre contexte de travail.

Premièrement, nous avons discuté le problème de monitoring dans les MANETs. La solution retenue consiste en un monitoring probabiliste et adaptatif qui permet un gain considérable en énergie, limite le trafic supplémentaire et réduit les faux positifs. Nous avons discuté les attaques possibles et les solutions envisageables et présenté notre propre solution.

Deuxièmement, nous avons discuté l'estimateur réseau qui permet principalement d'adapter le moniteur aux conditions réseau pour réduire les faux positifs.

Troisièmement, nous avons développé un gestionnaire de réputation basé sur une approche bayesienne modifiée et qui utilise comme distribution a priori la distribution beta. Nous avons présenté également notre module de classification.

Quatrièmement, nous avons proposé un gestionnaire de recommandation qui intègre des composants que nous avons conçu ou adapté, à savoir :

- Un protocole de dissémination épidémique pour la diffusion des requêtes de recommandation qui permet de garantir le passage à l'échelle de notre mécanisme.
- Un algorithme de filtrage hybride qui permet d'utiliser toute l'information de réputation disponible, qu'elle soit négative ou positive.
- Un module de traitement des requêtes de recommandation, qui permet d'inciter les nœuds à recommander.
- Un gestionnaire de confiance basé sur une approche bayesienne modifiée.

Notre gestionnaire de recommandation fournit des mesures de défense pour faire face aux attaques classiques rencontrées dans les systèmes de coopération, à savoir : l'inactivité, la diffamation et la collusion. La diffusion des informations de réputation se fait d'une manière réactive ce qui limite les attaques possibles et permet d'économiser de la bande passante. La protection des messages de réputation est assurée par un mécanisme simple, ceci permet d'avoir une architecture simple qui peut être déployée facilement.

Finalement, nous avons discuté les réponses disciplinaires, informatives et correctives prises par notre mécanisme lors de la détection d'un nœud non coopératif.

Le mécanisme de coopération PARDON détaillé dans ce chapitre constitue une solution complète pour le déploiement des protocoles de réplication dans les environnements ad-hoc non coopératifs. Dans le chapitre suivant, nous évaluons les performances du mécanisme PARDON.

Chapitre 5. Evaluation des performances du mécanisme PARDON

5.1. Introduction

L'objectif principal de notre mécanisme de coopération PARDON est la détection et l'isolation des nœuds non coopératifs dans le réseau. Les paramètres que nous choisissons comme facteurs ayant une influence sur PARDON ainsi que les métriques que nous mesurons sont détaillés dans le présent chapitre. Les résultats de simulation que nous obtenons sont présentés sous forme de graphes accompagnés des discussions et interprétations.

5.2. Plan d'évaluation des performances

Avant d'entamer l'évaluation des performances de notre système, nous avons établi un plan d'évaluation des performances, tel que suggéré dans [Raj91], que nous définissons dans cette section.

5.2.1. Définition du système

L'objectif de cette évaluation des performances est d'étudier l'apport du mécanisme de coopération PARDON lorsqu'il est déployé avec un protocole de réplication push-based dans un environnement non coopératif. Le système étudié consiste en un protocole de réplication push-based générique déployé dans un réseau ad-hoc non coopératif. Chaque nœud du système peut jouer le rôle d'un client et/ou celui d'un serveur de réplication.

5.2.2. Les services

PARDON offre plusieurs services à savoir : le choix final des serveurs de réplication, les traitements des différentes requêtes lecture/écriture/sauvegarde, ainsi que la détection et l'isolation des nœuds non coopératifs.

5.2.3. Les métriques

Les métriques qui seront étudiées durant cette évaluation de performance sont les métriques d'efficacité qui permettent de mesurer l'exactitude de la détection des nœuds malveillants et la réduction de l'effet négatif des nœuds malveillants. Nous redéfinissons ces métriques dans notre cas comme suit :

a) Qualité de détection :

i) Temps de détection : cette métrique mesure le temps moyen de détection des nœuds non coopératifs [Refaei07] ;

ii) Taux de détection : cette métrique mesure le pourcentage des nœuds détectés avec succès par le système [Refaei07] ;

iii) Faux positives : lors de la détection des nœuds non coopératifs, des nœuds coopératifs peuvent être faussement identifiés comme non coopératifs. Cette métrique mesure combien de fois de tels cas se produisent [Refaei07] ;

iv) Faux négatives : lors de la détection des nœuds non coopératifs, des nœuds non coopératifs peuvent être faussement identifiés comme coopératifs. Cette métrique mesure combien de fois de tels cas se produisent [Refaei07] ;

v) Robustesses contre les attaques : cette métrique mesure la robustesse du système contre des attaques telles que la diffamation, l'inactivité et la collusion [Refaei07].

b) Qualité de réaction : cette métrique mesure la qualité de réaction du système de réputation en terme de bénéfice obtenu lorsque le nœud adopte un comportement coopératif [Refaei07]. Dans notre cas, cela revient à mesurer la disponibilité des données des nœuds coopératifs par rapport à celle des données des nœuds non coopératifs.

c) Réduction de l'effet négatif des nœuds malveillants : cette métrique mesure la capacité du système de réputation à réduire l'impact négatif des nœuds non coopératifs [Refaei07]. Dans notre cas, cela revient à mesurer la disponibilité des données avec et sans le système de gestion de réputation.

5.2.4. Les paramètres

Les paramètres qui affectent les performances de notre système sont les suivants.

1) Les paramètres réseau :
    a. Le nombre de nœuds formant le réseau
    b. Le nombre de nœuds non coopératifs
    c. La mobilité des nœuds
    d. Le protocole de routage
2) Les paramètres de réplication : la charge de réplication
    a. Le nombre de serveurs réplicas
    b. La fréquence de réplication : la fréquence des requêtes de réplication
    c. La fréquence des requêtes de lecture
3) Les paramètres de PARDON : tous les paramètres seuils de notre mécanisme.

5.2.5. La technique d'évaluation

Etant donné le temps et les ressources disponibles, nous avons choisi la simulation comme technique d'évaluation. Pour cela, nous utilisons le simulateur NS-2 [NS2].

5.2.6. La charge de travail

La charge de réplication sera générée automatiquement. Trois paramètres sont utilisés : (i) la fréquence de réplication, (ii) la fréquence des requêtes de lecture et (iii) le nombre de serveurs réplicas.

5.3. Méthodologie de simulation

Pour évaluer les performances de notre mécanisme PARDON, plusieurs simulations ont été faites dans des conditions différentes. Sachant que PARDON a pour but de détecter et d'isoler les nœuds non coopératifs pour améliorer la disponibilité des données, nous avons effectué quatre batteries principales de simulation.

La première batterie constitue le cas de référence. Elle permet d'étudier le taux de disponibilité des données dans un réseau ad-hoc coopératif ainsi que le lien entre ce taux et les différents paramètres réseau et réplication. Le cas de référence permet d'avoir une idée claire sur la variation du taux de disponibilité en fonction des paramètres réseau et réplication, ce qui nous facilite l'interprétation des résultats des autres batteries de simulation.

Ensuite, nous introduisons des nœuds non coopératifs qui ne stockent pas localement les données et qui seront donc incapables de répondre par la suite aux différentes requêtes. Nous étudions le taux de disponibilité des données dans un réseau ad-hoc non coopératif et le lien entre ce taux et les différents paramètres réseau et réplication. Nous appelons ce cas "sans-défense".

Finalement, nous étudions le taux de disponibilité des données dans un réseau ad-hoc non coopératif où le protocole de réplication est renforcé par notre mécanisme de coopération PARDON. Nous appelons ce cas "PARDON". Nous analysons en détail PARDON en évaluant les métriques discutées dans la section 5.2.3. Nous avons effectué deux batteries de simulation : la première permet d'étudier PARDON en mode local "PARDON-Local" et la deuxième ce concentre sur PARDON en mode global "PARDON-Global".

Nos simulations ont été faites sur une machine Intel Core 2 1.99 GHz avec 2 Go de RAM, en utilisant Cygwin et le simulateur réseau NS-2 [NS2]. Nous avons répété dix fois chaque simulation avec variation du paramètre aléatoire puis nous avons utilisé une moyenne arithmétique pour déduire les valeurs à reporter sur les graphes. Le niveau de confiance des intervalles de confiance est de 95%. Les paramètres fixes, utilisés dans la majorité des simulations, sont donnés dans le tableau 5.1. Les paramètres radio, débit et couche MAC sont sélectionnés pour représenter un équipement standard. La vitesse est uniformément distribuée entre 0 et 20 m/s pour offrir une gamme d'utilisateurs qui sont dans un endroit fixe, marchant, ou conduisant une voiture. La scène choisie représente approximativement le centre d'une ville. Le modèle de mobilité choisi est le modèle *Random waypoint* [Yoon03] dans lequel les nœuds se déplacent vers une destination aléatoire à une vitesse uniformément distribuée entre 0 et la vitesse maximale indiquée. Lorsqu'un nœud du réseau atteint sa destination, il reste stationnaire pour un temps uniformément distribué entre 0 et le temps de pause indiqué. Notre choix de ces paramètres a pour but de refléter le comportement réaliste d'un utilisateur.

Paramètres	**Valeurs**
Scène de la simulation	1000 *m* ×1000 *m*
Durée de la simulation	15 minutes
Vitesse de déplacement	uniformément distribuée dans [0, 20] *m/s*
Modèle de mobilité	Random waypoint
Modèle de propagation	Two-ray ground
MAC	802.11 DCF
Portée radio	250 *m*

Tableau 5.1. Les paramètres fixes

5.4. Les simulations et les résultats

5.4.1. Batterie 1 : "Référence"

Le système dans cette batterie de simulation est un réseau ad-hoc coopératif avec un protocole de réplication push-based générique. La charge de réplication est définie comme suit : au début, chaque nœud client sélectionne RS_NUMBER (RSN) serveurs de réplication. Ensuite, toute les PERIODIC_SELECTION_TIME (PST), il ajoute à la sélection initiale RSN serveurs de réplication et envoie une requête de lecture toute les PERIODIC_CHECK_TIME (PCT). Pour une évaluation significative, les requêtes de lecture sont envoyées par chaque nœud client vers tous ses serveurs de réplication. Chaque nœud du réseau joue le rôle d'un client et peut jouer le rôle d'un serveur. Le taux de disponibilité calculé est le taux de disponibilité global défini dans [Derhab07] comme étant le nombre de requêtes de lecture envoyées par tous les nœuds divisé par le nombre de réponses reçues par tous les nœuds.

Nous étudions le taux de disponibilité ainsi que le lien entre ce taux et les différents paramètres réseau et charge de travail. Le tableau 5.2 résume les différents scénarios simulés.

Paramètres	**Valeurs**
Nombre de nœuds (nn)	20, 30, 40, et 50
Temps de pause (*s*)	0, uniformément distribué dans [0, 100], [0, 300], [0, 600] et [0, 900]
Protocole de routage	DSR, AODV et DSDV
RS_NUMBER	1, 2 et 3
PERIODIC_SELECTION_TIME (*s*)	100
PERIODIC_CHECK_TIME (*s*)	20
Moniteur	Périodique

Tableau 5.2. Les paramètres de la batterie 1

La figure 5.1 représente le taux de disponibilité en fonction des paramètres réseau ((nn = 20, RSN = 1), ((nn = 30, RSN = 2), ((nn = 40/50, RSN = 3), PCT = 20 et PST = 100) et la figure 5.2 représente le taux de disponibilité en fonction du protocole de routage (nn = 30).

Dans une scène de 1000 *m* x 1000 *m*, quand le nombre de nœuds augmente, ceci améliore la connectivité du réseau qui devient de plus en plus dense et par conséquent les données demandées sont susceptibles d'être disponibles. Pour un nombre de nœuds inférieur à 40, la disponibilité des données augmente avec l'augmentation de la mobilité des nœuds (diminution du temps de pause), parce qu'en raison de la mobilité des nœuds beaucoup de nœuds se déplace les uns envers les autres pour former une grande partition (figure 5.1). Bien sûr, le trafic généré dans le réseau augmente mais l'accès au medium reste acceptable ce qui n'influence pas négativement le taux de délivrance des paquets.

Pour 50 nœuds, le réseau est considéré très dense. Pour un temps de pause de 900s (réseau statique), la plupart des nœuds appartiennent à une seul partition, ceci mène à la disponibilité de données la plus élevée. Lorsque la mobilité des nœuds augmente, la grande partition se fractionne dans des partitions plus petites chacune avec quelques nœuds seulement et par conséquent les chances d'accéder à une donnée diminue (figure 5.1).

Nous remarquons sur la figure 5.2 que DSR dépasse AODV et DSDV en terme de taux de disponibilité. Ceci était prévisible puisqu'il est évident que le taux de disponibilité des données est directement lié au taux de délivrance de paquets. Dans les études de simulations faites sur les protocoles de routage [Das00, Broch98], il a été démontré que DSR dépasse AODV et DSDV en terme de taux de délivrance de paquets. Pour un temps de pause égal à 0, DSDV ne dépasse pas un taux de disponibilité de 50%. Ceci est dû au fait que DSDV est un protocole proactif qui présente des performances inférieures par rapport à celles des protocoles réactifs [Das00, Broch98].

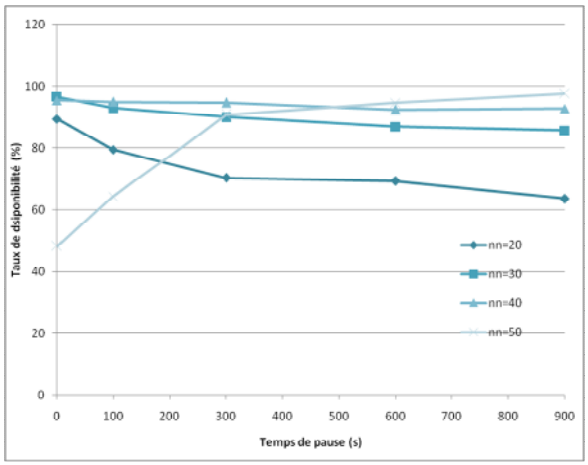


Figure 5.1. Taux de disponibilité en fonction de la mobilité

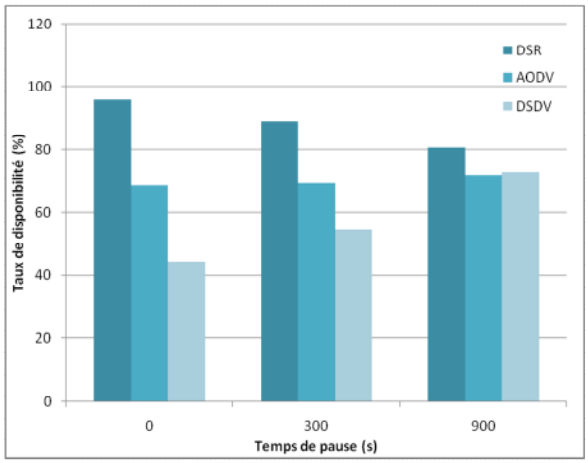


Figure 5.2. Taux de disponibilité en fonction du protocole de routage

5.4.2. Batterie 2 : "sans-défense"

Dans ce cas, le système est un réseau ad-hoc non coopératif avec un protocole de réplication push-based générique, et les nœuds non coopératifs sont sélectionnés aléatoirement. Chaque nœud client sélectionne RSN serveurs de réplication au début. Ensuite, toute les PST, il ajoute à la sélection initiale RSN serveurs de réplication et envoie une requête de lecture toute les PCT.

Nous définissons trois types de taux de disponibilité :

- Le taux de disponibilité global (TDG), défini comme étant le nombre de requêtes de lecture envoyées par tous les nœuds divisé par le nombre de réponses positives reçues par tous les nœuds ;
- Le taux de disponibilité des nœuds coopératifs (TDC), défini comme étant le nombre de requêtes de lecture envoyées par les nœuds coopératifs divisé par le nombre de réponses positives reçues par les nœuds coopératifs ;
- Le taux de disponibilité des nœuds non coopératifs (TDNC), défini comme étant le nombre de requêtes de lecture envoyées par les nœuds non coopératifs divisé par le nombre de réponses positives reçues par les nœuds non coopératifs ;

Nous étudions le taux de disponibilité ainsi que le lien entre ce taux et le nombre de nœuds non coopératifs présents dans le réseau. Le tableau 5.3 résume les différents scénarios simulés.

Paramètres	Valeurs
Nombre de nœuds	40
Nombre de nœuds non coopératifs	0 (0%), 4 (10%), 8 (20%), 12 (30%), 16 (40%), 20 (50%), 24 (60%), 28 (70%), 32 (80%), 36 (90%), 40 (100%)
Protocole de routage	DSR
Temps de pause (*s*)	0, uniformément distribué dans [0, 300], [0, 900]
RS_NUMBER	3
PERIODIC_SELECTION_TIME	100
PERIODIC_CHECK_TIME	20
Moniteur	Périodique

Tableau 5.3. Les paramètres de la batterie 2

Comme le montre la figure 5.3, tel que prévu, quand le nombre de nœuds non coopératifs augmente, le taux de disponibilité diminue jusqu'à atteindre la valeur 0. Lorsque tous les nœuds du réseau sont non coopératifs ceci représente un cas extrême de non coopération qui engendre un réseau tout simplement non fonctionnel.

La courbe de la figure 5.3 est pratiquement linéaire. La courbe de tendance associée (avec un coefficient de détermination $R^2 = 0,998$) est $y = -0,954x + 93,46$, où y représente le taux de disponibilité et x le pourcentage des nœuds non coopératifs. Il est à noter que cette relation n'est pas générale mais plutôt spécifique à nos expériences.

Pour les trois types de mobilité (forte, moyenne et quasi statique), les courbes obtenues sont pratiquement les mêmes. Ceci signifie, dans notre cas, que le taux de diminution du taux de disponibilité est directement lié au nombre de nœuds non coopératifs dans le réseau.

Nous remarquons bien que l'effet négatif de la non-coopération dans le protocole de réplication est indépendant de la mobilité des nœuds ; par opposition à l'effet négatif de la non-coopération dans les protocoles de routage qui dépend de la mobilité des nœuds (à titre d'exemple, le lecteur est invité à examiner les résultats des simulations dans [Buchegger02 et Bansal03]).

Soit un réseau composé de N nœuds dont n nœuds sont non coopératifs. Supposons que la sélection des nœuds ce fait de manière uniforme, c'est-à-dire que la sélection des nœuds est équiprobable. La probabilité qu'un nœud coopératif sélectionne comme serveur de réplication un nœud coopératif est notée *P(c/c)* et la probabilité qu'un nœud non coopératif sélectionne comme serveur de réplication un nœud coopératif est notée *P(nc/c)*. Nous avons :

$$\begin{cases} P(nc/c) = \dfrac{N-n+1}{N-1} \\ P(c/c) = \dfrac{N-n-1}{N-1} \end{cases}$$

Nous voyons bien que, théoriquement, *P(nc/c)* est supérieure à *P(c/c).* Pratiquement, sur la figure 5.4 nous remarquons que les nœuds non coopératifs ont un taux de disponibilité souvent supérieur à celui des nœuds coopératifs. Ceci permet non seulement de constater l'effet négatif du comportement non coopératif mais pire encore, qu'un nœud non coopératif est avantagé par rapport à un nœud coopératif, et ce, sans considérer le fait qu'il préserve aussi ses ressources.

Ce qui motive encore plus le comportement non coopératif et rend inévitablement le réseau non fonctionnel.

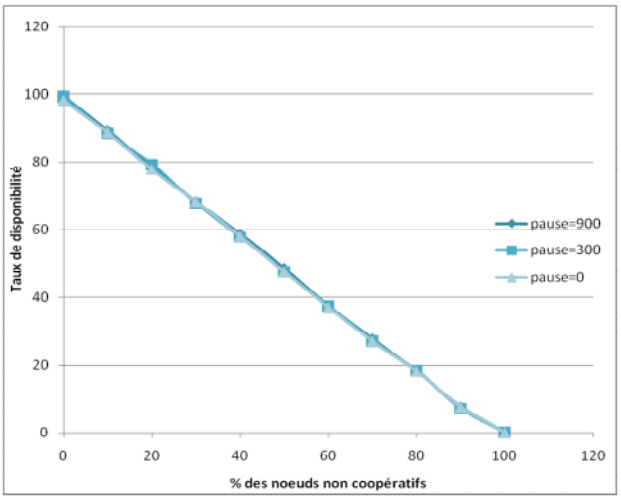


Figure 5.3. TDG en fonction du nombre de nœuds non coopératifs

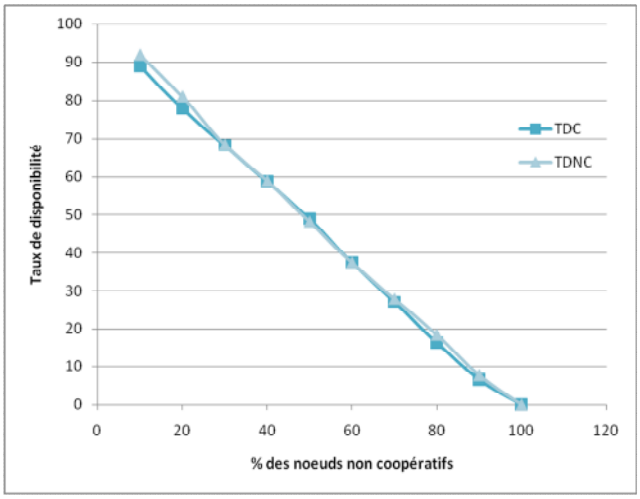


Figure 5.4. TDC et TDNC en fonction du nombre de nœuds non coopératifs

5.4.3. Batterie 3 : "PARDON-Local"

Le système dans ce cas est un réseau ad-hoc non coopératif avec un protocole de réplication push-based générique et notre système de réputation PARDON. La charge de réplication est similaire à celle de la batterie 2. Pour une évaluation objective, les nœuds non coopératifs utilisent aussi PARDON pour améliorer leur taux de disponibilité.

5.4.3.1. Sous batterie 3.1 : le GR

Nous étudions les métriques discutées dans 5.1 ainsi que le lien entre ces métriques et les paramètres réseau. Dans cette sous batterie, chaque nœud utilise seulement l'information directe collectée pour prendre des décisions. Le tableau 5.4 résume les différents scénarios simulés.

Paramètres	**Valeurs**
Nombre de nœuds	40
Nombre de nœuds non coopératifs	0 (0%), 4 (10%), 8 (20%), 12 (30%), 16 (40%), 20 (50%), 24 (60%), 28 (70%), 32 (80%), 36 (90%)
Protocole de routage	DSR
RS_NUMBER	3
PERIODIC_SELECTION_TIME	100
PERIODIC_CHECK_TIME	20
ϕ_e , φ_e	0.999
Seuil de coopération	0.25
Seuil d'expériences	4
Mise à jour périodique de la RSL	Désactivée
Moniteur	Périodique

Tableau 5.4. Les paramètres de la sous batterie 3.1

La figure 5.5 représente la réduction de l'effet négatif des nœuds non coopératifs. PARDON-Local permet de garder un taux de disponibilité supérieur dans tous les cas. De plus, nous

remarquons que souvent PARDON-Local permet d'avoir un taux de disponibilité deux fois supérieur au cas sans-défense (par exemple à 60% de nœuds non coopératifs, le cas sans-défense à un TD = 33,96%, alors qu'avec PARDON-Local nous avons un TD = 72,29%). Il faut noter que le taux de disponibilité dépend également du temps de simulation. Dans ce cas, si nous augmentons le temps de simulation (plus de 900 *s*) la différence entre le taux de disponibilité de "PARDON-Local" et celui de "sans-défense" sera encore plus grande.

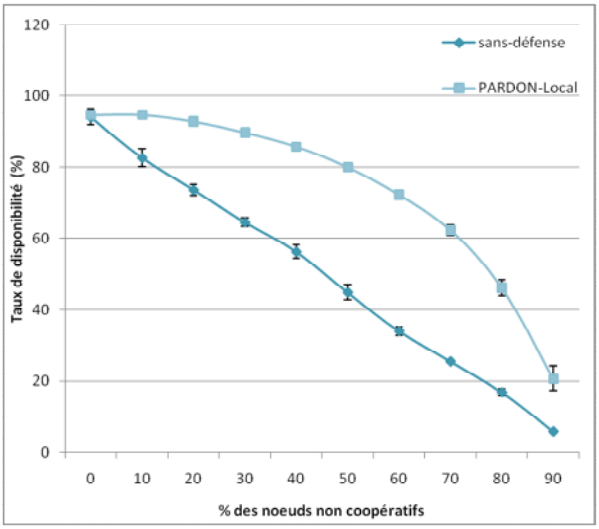


Figure 5.5. Réduction de l'effet négatif des nœuds non coopératifs

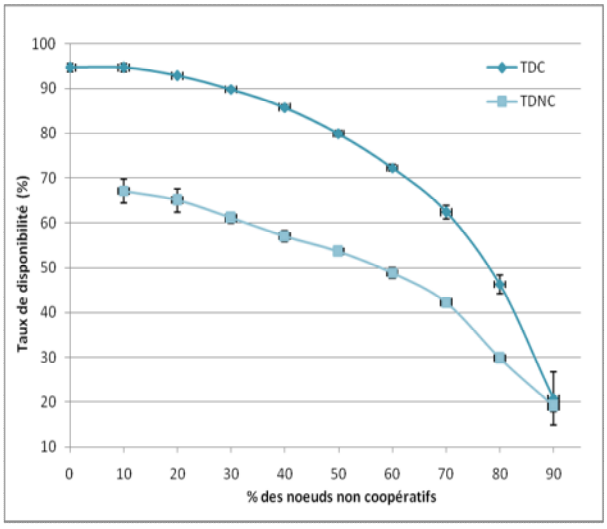


Figure 5.6. Taux de disponibilité des nœuds coopératifs et non coopératifs

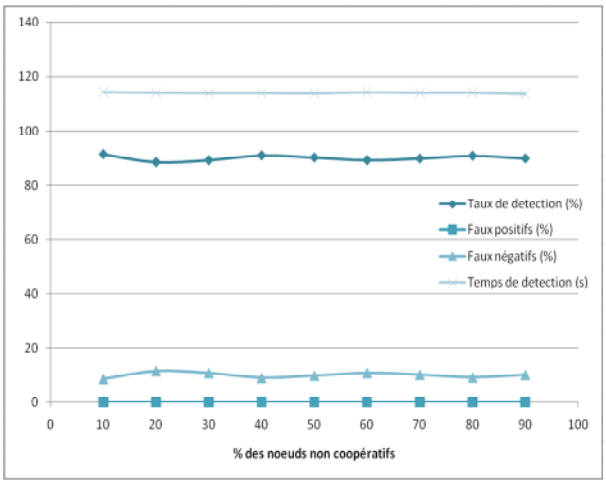


Figure 5.7. Qualité de détection

Dans nos simulations, même les nœuds non coopératifs utilisent PARDON pour éviter eux même les autres nœuds non coopératifs et améliorer ainsi leur taux de disponibilité. Ceci représente un cas extrême de non coopération, car les nœuds non coopératifs essayent d'exploiter PARDON à leur profit. Nous voyons sur la figure 5.6 que la qualité de réaction de PARDON est très acceptable car les nœuds coopératifs gardent toujours un taux de disponibilité nettement supérieur à celui des nœuds non coopératifs.

Pour la qualité de détection, nous voyons sur la figure 5.7 que le taux de détection est supérieur à 90%. La non détections de certains nœuds est due au facteur temps, c'est-à-dire certains nœuds coopératifs sélectionnent des nœuds non coopératifs juste avant la fin de la simulation et n'aurons donc pas suffisamment de temps pour les détecter. Si nous ignorons ce cas, le taux de détection est 100%. Quant au taux de faux négatifs, il est le complément du taux de détection.

Les faux positifs dépendent essentiellement des conditions réseau. Nous savons d'après [Gupta98] que la connectivité dans un réseau ad-hoc est forte si la condition suivante est vérifiée :

$$r \geq a\sqrt{\frac{C\ln(n)}{n}}, \quad C > \frac{1}{\pi}$$

Où *r* est la portée de transmission, *a* est la longueur du réseau, *C* est une constante et *n* est le nombre de nœud.

Dans notre cas, un réseau de 1000 *m* x 1000 *m* et un nombre total de nœud égal à 40 avec une portée de transmission de 250 *m* représente un réseau avec une bonne connectivité, en ajoutant l'absence d'un trafic applicatif ; ceci explique le 0% de faux positifs.

Le temps de détection moyen par nœud dépend de deux facteurs qui sont la PCT et le seuil de classification du GR. Avec une PCT uniformément distribuée dans [20 *s*, 30 *s*] et un seuil de coopération de 0.25, nous avons un temps de détection de l'ordre de 114 *s*.

5.4.3.2. Sous batterie 3.2 : le moniteur

Nous étudions dans cette sous batterie le problème des faux positifs et le lien entre ce problème et les différents types de moniteurs que nous avons proposé. Le moniteur adaptatif assure une adaptation par rapport à la réputation des nœuds. L'adaptation par rapport aux conditions réseau n'est pas prise en charge dans cette simulation. Par rapport aux expériences précédentes, nous introduisons dans cette sous batterie un trafic applicatif. Le tableau 5.5 résume les différents scénarios simulés.

Paramètres	**Valeurs**
Nombre de nœuds	20
Nombre de nœuds non coopératifs	6 (30%)
Type de moniteur	Périodique, Probabiliste (P_{on} de 0.2 et 0.5) et Adaptatif
Protocole de routage	DSR
RS_NUMBER	2
PERIODIC_SELECTION_TIME	100
PERIODIC_CHECK_TIME	20
Nombre de connexions	5, 10, 15, 20, 25 et 30

Tableau 5.5. Les paramètres de la sous batterie 3.2

Nous voyons clairement sur la figure 5.9 que le moniteur périodique assure le temps de détection minimum par rapport aux autres moniteurs. D'un autre coté, nous voyons sur la figure 5.8 qu'il engendre un taux de faux positifs important par rapport aux autres moniteurs.

Un moniteur périodique avec une P_{On} de 0.5 permet de diminuer le taux des faux positifs, mais en contrepartie, il augmente le temps de détection. Utiliser une P_{On} de 0.2 permet de gagner en terme de taux de faux positifs, mais nous perdons en termes de temps de détection. En conclusion, en fixant la valeur de P_{On}, nous ne pouvons pas améliorer en même temps les deux paramètres (faux positifs et temps de détection) mais nous pouvons plutôt trouver un compromis.

Le moniteur adaptatif permet d'avoir une P_{On} variable pour chaque nœud. La variation de la P_{On} dépend de la réputation du nœud en question. Ceci permet justement de trouver un bon compromis avec un temps de détection proche de celui du moniteur périodique (de l'ordre de 159 s) et un taux de faux positifs proche de celui du moniteur probabiliste avec une P_{On} de 0.2. Ce qui confirme notre analyse théorique.

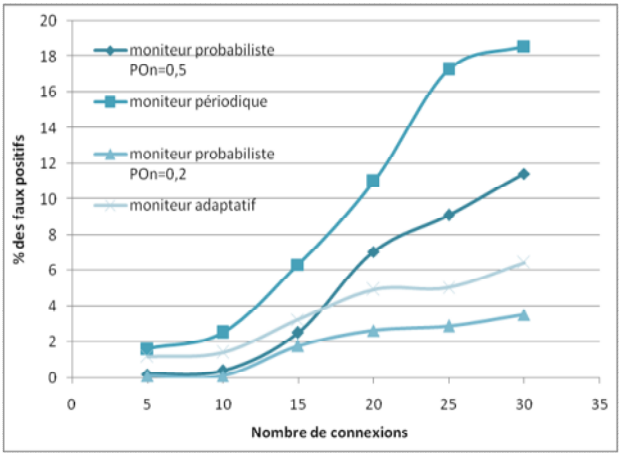


Figure 5.8. Pourcentage des faux positifs

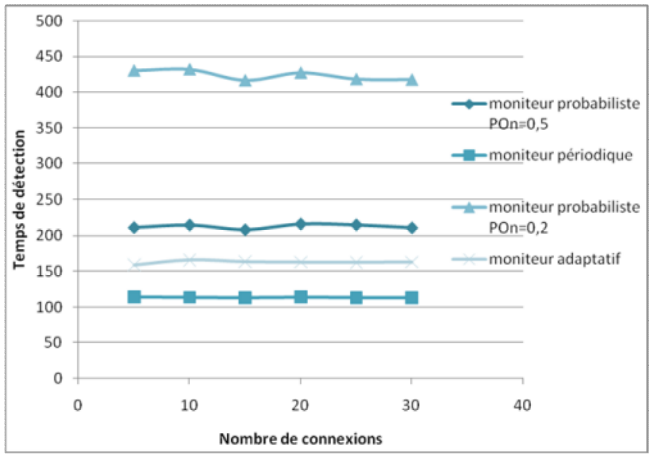


Figure 5.9. Temps de détection

5.4.4. Batterie 4 : "PARDON-Global"

Le système dans ce cas est un réseau ad-hoc non coopératif avec un protocole de réplication push-based générique et notre système de réputation PARDON en mode global. Nous étudions dans cette batterie trois aspects : (i) le protocole de dissémination probabiliste utilisé pour la diffusion des requêtes de recommandations, (ii) les performances de PARDON-Global par rapport à PARDON-Local où le réseau ne contient pas de nœuds malhonnêtes et (iii) les performances de PARDON-Global dans un réseau contenant des nœuds malhonnêtes actifs.

5.4.4.1. Sous batterie 4.1 : la dissémination probabiliste

Nous étudions dans cette sous batterie le protocole de dissémination probabiliste à travers deux cas. Dans le premier, nous utilisons une probabilité de retransmission fixe. Dans le deuxième, une probabilité de retransmission adaptative est utilisée telle que décrite dans notre algorithme de dissémination probabiliste (algorithme 4). Concernant les métriques mesurées, le taux de délivrance calculé est la fraction des nœuds qui ont reçus l'ensemble des paquets diffusés [Drabkin07]. Le nombre de retransmissions indique la charge induite par le protocole en terme de retransmission des paquets. Dans nos simulations, le nombre de retransmissions indique le nombre de retransmissions effectif. Les retransmissions annulées ne sont pas comptabilisées.

Nous faisons varier le nombre de nœuds diffuseurs ; chaque nœud diffuseur diffuse dix messages à des instants aléatoires. Le tableau 5.6 résume les différents scénarios simulés.

Paramètres	**Valeurs**
Nombre de nœuds	40
Nombre de nœuds diffuseurs	1, 5, 10, 15 et 20
Probabilité de retransmission	Fixe (1, 0.8, 0.4 et 0.65) et Adaptative (β=6.5)
short_jitter	10 *ms*

Tableau 5.6. Les paramètres de la sous batterie 4.1

Le taux de délivrance d'un protocole de dissémination probabiliste dépend essentiellement de sa probabilité de retransmission. Sur la figure 5.10, nous voyons qu'un protocole de diffusion simple (*p* = 1, *blind flooding*) permet d'atteindre un taux de délivrance supérieur à 95% ; mais en contrepartie, il engendre un coût (nombre de retransmission) très important. Diminuer la probabilité de retransmission permet de réduire le coût du protocole ; mais au prix d'une réduction du taux de délivrance.

Lorsque le nombre de nœuds diffuseurs augmente le taux de délivrance diminue légèrement. Ceci est dû principalement aux collisions qui peuvent se produire, ce qui augmente le taux de pertes des paquets.

L'étude empirique faite dans [Haas02 et Drabkin07] montre que souvent une probabilité de retransmission de 0.65 permet d'avoir un bon compromis performances/coût. Dans notre cas, avec une probabilité de retransmission de 0.65 le taux de délivrance dépasse 80%. Ceci est très utile lorsque l'estimation du nombre de nœuds voisins est impossible ou coûteuse.

L'idée de [Drabkin07], stipule qu'un protocole de diffusion probabiliste adaptatif (emploi une probabilité de retransmission adaptative) permet d'obtenir des résultats meilleures par rapport à un protocole de diffusion probabiliste non adaptatif (emploi une probabilité de retransmission fixe). Nous avons obtenu le même résultat ; avec une probabilité de retransmission adaptative nous avons un taux de délivrance qui dépasse le cas d'une probabilité de retransmission de 0.65 (figure 5.10) avec un coût pratiquement similaire (figure 5.11).

Pour un taux de délivrance supérieur à 80% et un jitter de 10 *ms*, la latence est pratiquement la même dans tous les cas et elle est de l'ordre de 23 *ms*. La latence augmente légèrement lorsque le nombre de nœuds diffuseurs augmente ; ceci est dû au temps supplémentaire passé dans les files d'attentes des nœuds du réseau. L'algorithme que nous utilisons n'emploi aucune mesure corrective ceci permet de réduire considérablement la latence du protocole, sans influencer négativement ses performances.

Nous voyons bien que la dissémination probabiliste offre un avantage certain même dans des réseaux relativement petits.

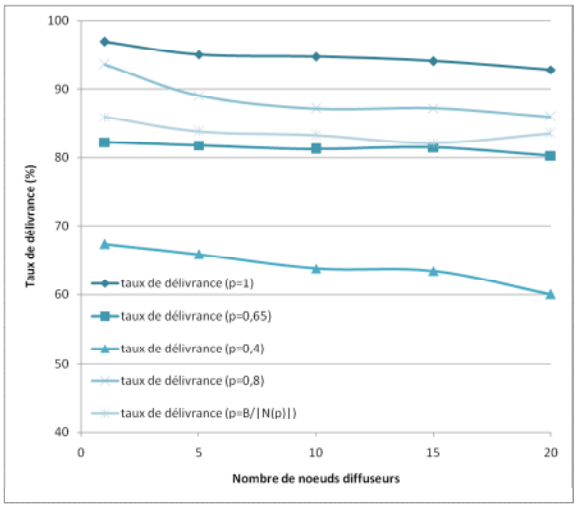


Figure 5.10. Taux de délivrance

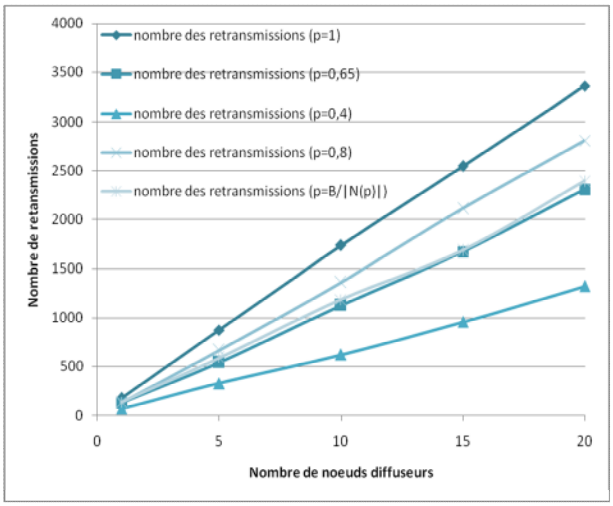


Figure 5.11. Nombre de retransmissions

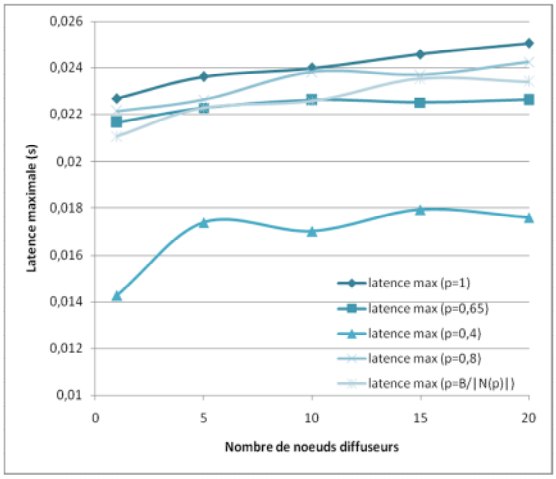


Figure 5.12. La latence maximale

5.4.4.2. Sous batterie 4.2 : "PARDON-Global" PARDON en mode global sans des nœuds malhonnêtes

Nous étudions dans cette sous batterie les performances de PARDON-Global par rapport à PARDON-Local où le réseau ne contient pas des nœuds malhonnêtes. Si un nœud client sélectionne un serveur considéré nouveau, le nœud client diffuse une requête de recommandation. Les réponses collectées sont filtrées à l'aide de notre algorithme de filtrage. Si aucune réponse n'est obtenue, le nœud adopte une stratégie optimiste qui consiste à choisir le nœud serveur considéré nouveau avec une probabilité égale à 1. Le tableau 5.7 résume les paramètres de simulation.

Sur la figure 5.13, nous remarquons clairement que PARDON-Global permet d'améliorer le taux de disponibilité rapidement par rapport à PARDON-Local. Ceci s'explique par le fait que l'information de réputation devient de plus en plus disponible avec le temps, ce qui permet d'éliminer rapidement les nœuds non coopératifs et améliorer ainsi rapidement le taux de disponibilité. Du point de vue temps de détection, nous constatons sur la figure 5.15 que PARDON-Local souffre d'un temps de détection relativement important par rapport à PARDON-

Global. De plus, ce temps a tendance à augmenter avec le temps car le nombre de nœuds serveurs augmente, ce qui réduit la probabilité de vérification. Tel qu'expliqué auparavant, dans PARDON-Global, l'information de réputation est de plus en plus disponible, ce qui permet de réduire significativement le temps de détection lequel atteint 50*s* après 2000*s* de temps de simulation alors qu'il était de 164*s* dans PARDON-Local.

Paramètres	**Valeurs**
Nombre de nœuds	40
Nombre de nœuds non coopératifs	13 (33%)
Protocole de routage	DSR
RS_NUMBER	1
PERIODIC_SELECTION_TIME	50
PERIODIC_CHECK_TIME	20
ϕ_e , φ_e	0.999
Seuil de coopération	0.25
Seuil d'expériences	4
Mise à jour périodique de la RSL	Désactivée
Mise à jour périodique de la RR	Désactivée
Prob_mi	0.3
Moniteur	Adaptatif

Tableau 5.7. Les paramètres de la sous batterie 4.2

La figure 5.14 montre que dans PARDON-Global les nœuds bénéficient d'un taux de détection meilleur par rapport à celui de PARDON-Local. Dans PARDON-Global, le taux de détection atteint 100% après seulement 1800*s* de temps de simulation.

PARDON-Global permet d'améliorer rapidement le taux de détection et, par conséquent, de réduire rapidement les faux négatifs pour atteindre 0% après 1800*s* de temps de simulation. Concernant les faux positifs, PARDON-Local permet de garder un taux inférieur à 1% (voir figure 5.16). Pratiquement, ce 1% de faux positifs représente une information fausse qui sera diffusée par les nœuds dans PARDON-Global de manière non intentionnelle, ce qui va augmenter le taux des faux positifs. L'algorithme de filtrage permet de réduire considérablement cette information fausse ; d'où un taux de faux positifs qui ne dépasse pas les 1,56% dans PARDON-Global.

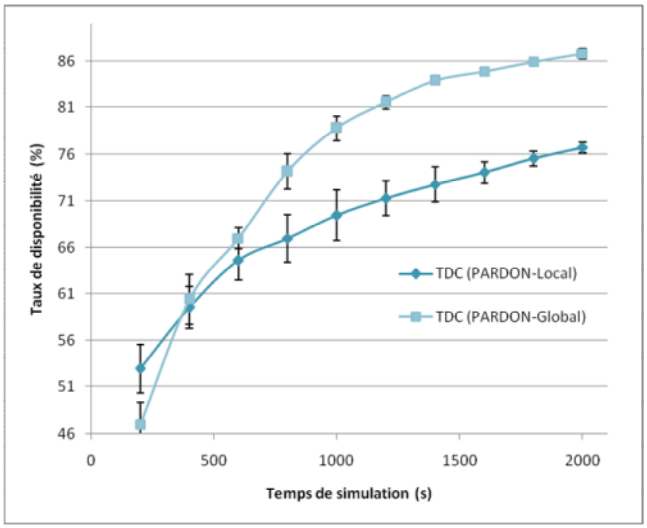


Figure 5.13. TD des nœuds coopératifs

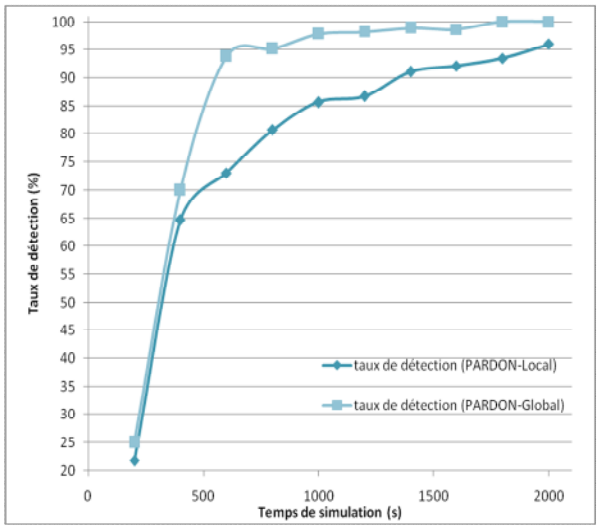


Figure 5.14. Taux de détection

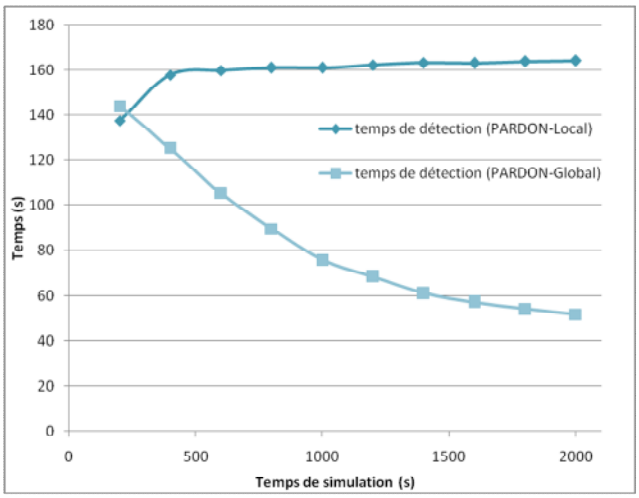


Figure 5.15. Temps de détection

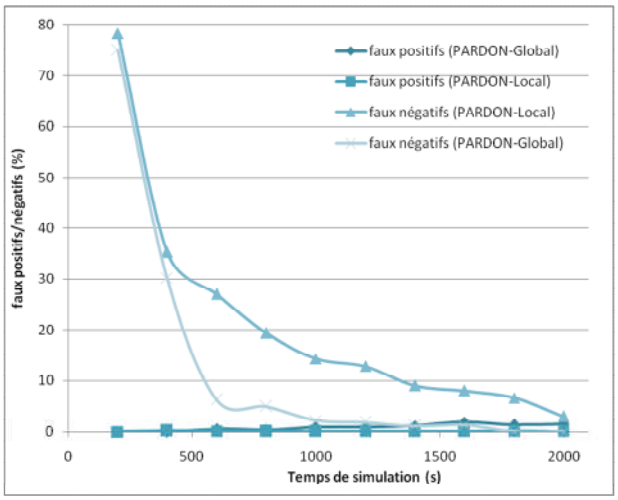


Figure 5.16. Les faux positifs et négatifs

5.4.4.3. Sous batterie 4.3 : "PARDON-Global-MA" PARDON en mode global avec des nœuds malhonnêtes actifs

Dans cette batterie, nous étudions l'influence de la présence des nœuds malhonnêtes actifs sur les performances de PARDON. Un nœud malhonnête actif (MA) qui répond activement aux requêtes de recommandations reçues produit certainement un effet négatif plus important qu'un nœud MI qui répond rarement aux requêtes de recommandations reçues. Pour cette raison, nous employons des nœuds MA dans nos simulations. Un nœud MA recommande avec beta(S_n,S_p) au lieu de beta(S_p,S_n). Les nœuds MA sont sélectionnés aléatoirement. Le tableau 5.8 résume les différents scénarios. Les paramètres non spécifiés sont similaires à ceux de la batterie précédente.

Sur les figures 5.17 et 5.18, nous remarquons que pour un pourcentage de nœuds MA allant jusqu'à 30%, le taux de détection reste supérieur à 99,7% et le taux de faux positifs reste inférieur à 2%. Ceci permet d'affirmer que l'algorithme de filtrage rejette correctement les informations de recommandation injustes.

Paramètres	Valeurs
Nombre de nœuds	40
Nombre de nœuds non coopératifs	13 (33%)
Nombre de nœuds malhonnêtes actifs	De 4 (10%) à 40 (100%)

Tableau 5.8. Les paramètres de la sous batterie 4.3

Pour un pourcentage de nœuds MA supérieur à 60%, le taux de faux positifs dépasse 10%. Avec un taux pareil nous pouvons conclure que l'algorithme de filtrage a du mal à filtrer correctement les informations de recommandation injustes car l'hypothèse sur laquelle se tient notre algorithme n'est plus valable dans ce cas. Il faut noter, qu'un nœud MA recommande avec beta(Sn,Sp). Ceci permet dans le cas général de produire implicitement une attaque par collision, ce qui représente un cas extrême de malhonnêteté des nœuds. Dans un scénario de déploiement réel, lancer une telle attaque s'avère difficile et dans ce cas l'algorithme de filtrage résiste à un pourcentage de nœuds MA beaucoup plus important.

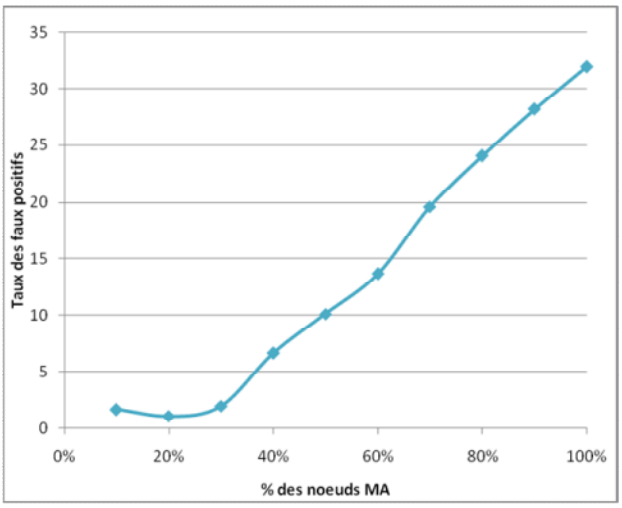


Figure 5.17. Les faux positifs en fonction du nombre de nœuds MA

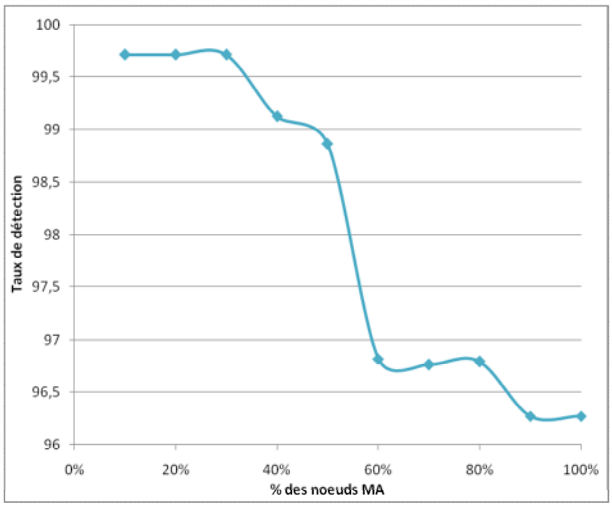


Figure 5.18. Taux de détection en fonction du nombre de nœuds MA

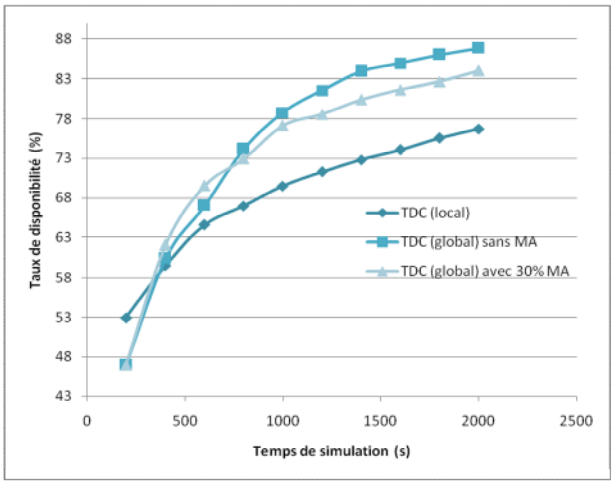


Figure 5.19. Taux de disponibilité dans différents scénarios

La figure 5.19 montre qu'avec un réseau dont 30% des nœuds sont MA, PARDON assure un taux de disponibilité relativement inférieur à celui d'un réseau sans des nœuds MA. Mais ce taux reste nettement supérieur à celui obtenu avec PARDON en mode local. Nous pouvons conclure que le mode global est toujours bénéfique même s'il y a un certain nombre de nœuds MA dans le réseau.

5.4.4.4. Sous batterie 4.4 : comparaison entre notre algorithme de filtrage et l'algorithme de filtrage proposé dans [Whitby04]

Dans cette sous batterie nous effectuons une comparaison quantitative entre notre algorithme de filtrage et l'algorithme de filtrage proposé dans [Whitby04]. Les deux algorithmes sont étudiés sous les mêmes paramètres de simulation, un quantile de 0,25 est fixé pour l'algorithme de [Whitby04]. Le tableau 5.9 résume les paramètres de simulation. Les paramètres non spécifiés sont similaires à ceux de la batterie précédente.

Paramètres	**Valeurs**
Nombre de nœuds	40
Nombre de nœuds non coopératifs	13 (33%)
Nombre de nœuds malhonnêtes	12 (30%)

Tableau 5.9. Les paramètres de la sous batterie 4.4

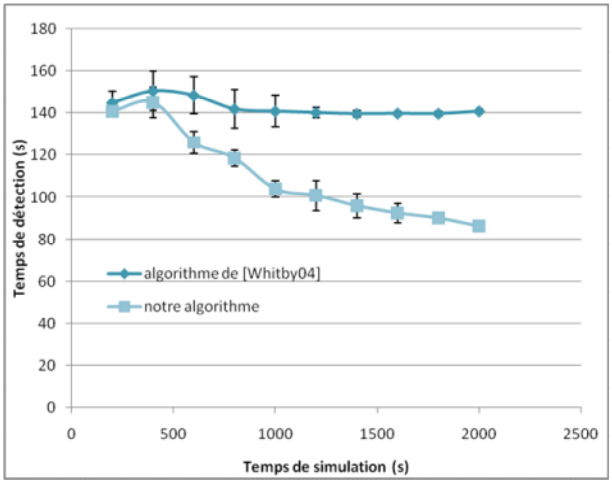


Figure 5.20. Temps de détection

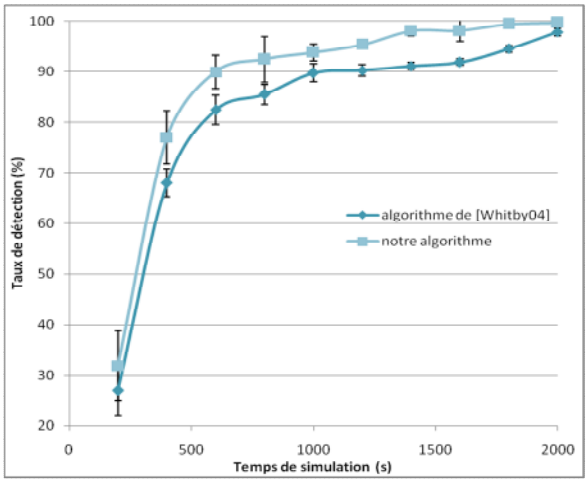


Figure 5.21. Taux de détection

Suite à notre analyse approfondie de l'algorithme de filtrage proposé dans [Whitby04], nous avons constaté que cet algorithme rejette souvent l'ensemble des recommandations reçues[14]. Ce qui explique, dans ce cas, le fait que PARDON ne bénéficie pas vraiment du mode global d'où un temps de détection très proche de celui du mode local (voir figure 5.20). Par contre, l'algorithme de filtrage hybride que nous proposons assure un temps de détection nettement inférieur à celui du mode local. De ce fait, notre algorithme enregistre un taux de détection meilleur durant tout le

[14] Ceci reste vrai pour différentes valeurs des quantiles dans l'intervalle [0.2, 0.5].

temps de simulation (voir figure 5.21). Avec les deux algorithmes ci-dessus, le taux des faux positifs ne dépasse pas 2,5%.

5.5. Conclusion

Dans ce chapitre, nous avons présenté l'évaluation des performances de notre mécanisme de coopération PARDON. Nous avons adopté une démarche claire durant cette étude. Nous avons commencé par élaborer un plan d'évaluation de performances, tel que suggéré dans [Raj91]. Ensuite, nous avons défini quatre batteries principales de simulations. Toutes nos simulations ont été répétées au moins une dizaine de fois afin augmenter la fiabilité des résultats obtenus. Nous avons renforcé les comparaisons des différents graphes par des intervalles de confiances et choisi les paramètres de simulation qui reflètent un comportement réaliste d'un utilisateur dans des scénarios de déploiement réels.

A travers cette évaluation des performances, nous avons validé notre mécanisme de coopération PARDON suivant plusieurs axes. Nous avons montré que PARDON en mode local détecte et isole les nœuds non coopératifs, ce qui permet d'améliorer la disponibilité des données dans un réseau ad-hoc non coopératif. Le comportement adaptatif du moniteur pour prendre en charge la perte des paquets et son aspect probabiliste pour économiser de l'énergie, constituent une solution intéressante qui permet effectivement de réduire le taux des faux positifs et offre un temps de détection acceptable.

Dans toutes nos simulations, les nœuds non coopératifs utilisent PARDON pour améliorer leur taux de disponibilité. Dans des conditions extrêmes, PARDON offre une bonne qualité de réaction qui se traduit par un taux de disponibilité des nœuds non coopératifs inférieur à celui des nœuds coopératifs. Pour améliorer son taux de disponibilité, un nœud n'a qu'une seule option c'est d'adopter un comportement coopératif et c'est l'objectif de notre système de coopération PARDON.

PARDON en mode global combiné avec l'algorithme de filtrage que nous proposons offre des performances encore meilleures en accélérant la détection des nœuds non coopératifs tout en réduisant l'effet négatif des recommandations injustes.

Conclusion générale et Perspectives

Dans ce mémoire, nous avons proposé un mécanisme de coopération basé-réputation pour le déploiement des protocoles de réplication dans les environnements ad-hoc non coopératifs. D'après notre recherche bibliographique, aucune proposition similaire n'existe dans le domaine.

Nous avons présenté un état de l'art des mécanismes de coopération pour les réseaux ad-hoc tout en effectuant une comparaison, discuté les résultats de cette comparaison et tirés des conclusions sous formes de recommandations pour la conception d'un "bon" système de coopération pour le déploiement des protocoles de réplication dans les réseaux ad-hoc non coopératifs.

Nous avons proposé un nouveau mécanisme de coopération, appelé PARDON, qui détecte et isole les nœuds non coopératifs. Ce mécanisme permet ainsi d'améliorer la disponibilité des données dans les réseaux ad-hoc non coopératifs. Le mécanisme PARDON comprend les modules suivants :

- Un moniteur probabiliste et adaptatif qui permet un gain considérable en énergie, limite le trafic supplémentaire et réduit les faux positifs.
- Un estimateur réseau qui permet principalement d'adapter le moniteur aux conditions réseau pour réduire les faux positifs.
- Un gestionnaire de réputation basé sur une approche bayesienne modifiée qui comprend un module de classification.
- Un gestionnaire de recommandation qui comprend :
    - Un gestionnaire de confiance basé sur une approche bayesienne modifiée.
    - Un protocole de dissémination épidémique pour la diffusion des requêtes de recommandation qui permet de garantir le passage à l'échelle de notre mécanisme.
    - Un algorithme de filtrage hybride qui permet d'utiliser toute l'information de réputation disponible qu'elle soit négative ou positive.
    - Un module de traitement des requêtes de recommandations, qui permet d'inciter les nœuds à recommander.
- Un gestionnaire d'espace de stockage qui implémente les réponses disciplinaires prises par notre mécanisme PARDON.

Nous avons validé notre solution suivant plusieurs axes à travers une évaluation des performances en effectuant des simulations extensives.

Dans une première batterie de simulation, nous avons étudié le taux de disponibilité des données dans un réseau ad-hoc coopératif ainsi que le lien entre ce taux et les différents paramètres réseau et réplication. Nous avons montré dans une seconde batterie de simulation l'effet négatif de la non-coopération qui se traduit par un taux de disponibilité faible.

Dans une troisième batterie de simulation, nous avons étudié les performances de notre mécanisme PARDON en mode local. Pour un temps de simulation de 900*s* seulement, PARDON-Local permet d'avoir un taux de disponibilité deux fois supérieur au cas sans-défense avec une qualité de réaction très acceptable, un taux de détection supérieur à 90% et un temps de détection de l'ordre de 114*s*. Notre moniteur adaptatif enregistre un temps de détection relativement court de l'ordre de 150*s* tout en gardant un taux de faux positifs faible.

Dans une quatrième batterie de simulation, nous avons étudié les performances de notre mécanisme PARDON en mode global. PARDON-Global permet de réduire significativement le temps de détection qui atteint 50s après 2000s de temps de simulation par rapport à 164s dans PARDON-Local. PARDON-Global permet aussi d'améliorer rapidement le taux de détection et, par conséquent, de réduire rapidement les faux négatifs pour atteindre 0% après 1800s de temps de simulation. L'algorithme de filtrage hybride que nous proposons permet de réduire considérablement l'information de recommandation considérée injuste, il enregistre un taux de faux positifs qui ne dépasse pas les 1,56% dans PARDON-Global.

Sur la base du travail réalisé, nous dressons plusieurs perspectives de recherche et de développement qui sont les suivantes :

- Le mécanisme PARDON peut être étendu à d'autres domaines d'application, tel que le partage de fichiers.
- L'aspect adaptation par rapport aux conditions réseau nécessite le mode conception *cross-layer*. Ce mode de conception peut être exploité non seulement pour développer un estimateur réseau efficace mais aussi pour développer tout un mécanisme de coopération cross-layer. L'architecture de ce nouveau mécanisme doit être vue avec un œil très attentif, du moment que plusieurs alternatives sont possibles, qui sont :
    - L'architecture peut être indépendante des services réseau. Dans ce cas, le mécanisme de coopération connaît les différentes menaces auxquelles chaque service (routage, réplication, etc.) doit faire face, et le traitement nécessaire à exécuter lors de la détection de ces menaces. L'ajout d'un service implique la reconfiguration du mécanisme de coopération. Pour la punition des nœuds non coopératifs, on devra décider sur une punition partielle par service (interdire l'utilisation d'un service particulier) ou une punition extrême (isolement total du nœud).
    - Une deuxième alternative est d'implémenter un mécanisme de coopération par service. Dans ce cas, si un nœud est coopératif dans le service routage et non coopératif dans le service réplication, il sera puni seulement par le service de réplication. L'inconvénient de cette approche est peut-être le coût élevé de son implémentation.
- Un système de coopération intégrant l'aspect contexte dans une architecture cross-layer permet d'utiliser, par exemple, des informations de réputation liées à la couche réseau (fonctionnalité de routage ou d'expédition de données) pour prendre des décisions dans les couches supérieures de la pile protocolaire (dans le cas la réplication, c'est la couche application), est une idée de recherche qui mérite d'être explorée.

Références

[Altman04] E. Altman, A. Kherani, P. Michiardi, and R. Molva. Non-cooperative forwarding in ad-hoc networks. Technical Report INRIA Report No. RR-5116, 2004.

[Bansal03] Bansal S, Baker M. Observation-based cooperation enforcement in ad-hoc networks. Techical Report, Stanford University, 2003.

[Broch98] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad-hoc Network Routing Protocols. In Proc. of the ACM/IEEE MobiCom, October 1998.

[Buchegger02] S. Buchegger and J.Y. Le Boudec. Performance analysis of the CONFIDANT protocol. In Proceedings of 3rd ACM International Symposium, on Mobile Ad-hoc Networking and Computing (MobiHOC), Lausanne, June 2002.

[Buchegger03] S. Buchegger, J.Y. Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In Proceedings of WiOpt'03: Modeling and Optimization in Mobile, Ad-hoc and Wireless Networks, March, 2003.

[Buchegger03-2] S. Buchegger, J.Y. Le Boudec. Coping with False Accusations in Misbehavior Reputation Systems for Mobile Ad-hoc Networks. EPFL Technical Report Number IC/2003/31, May 2003.

[Buchegger04] S. Buchegger, J.Y. Le Boudec. A robust reputation system for P2P and mobile ad-hoc networks. In Proceedings of P2PEcon2004, June 2004.

[Buttyan01] L. Buttyan, J.P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad-hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, 2001.

[Buttyan03] L. Buttyan, J.P. Hubaux. Stimulating cooperation in self-organizing mobile ad-hoc networks. ACM/Kluwer Mobile Networks and Applications, 8(5), Oct. 2003.

[Capkun03] Capkun S, Hubaux JP. BISS: building secure routing out of an incomplete set of security associations. In Proceedings of ACM WiSe2003, September 2003.

[Carle04] J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. IEEE Computer Magazine, 2004.

[Caronni03] Germano Caronni and Marcel Waldvogel. Establishing Trust in Distributed Storage Providers. Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03). 2003.

[Cartigny03] Julien Cartigny. Contributions à la diffusion dans les réseaux ad-hoc. Thèse de doctorat. Université des Sciences et Technologies de Lille. Décembre 2003.

[Cheambe04] Cheambe J, Tchouto JJ, Tittel C, et al. Security in wireless ad-hoc networks. In Proceedings of 13th IST mobile and wireless communications, June 2004.

[Chen01] M. Chen and J. Singh. Computing and Using Reputations for Internet Ratings. In Proceedings of the Third ACM Conference on Electronic Commerce (EC'01). ACM, October 2001.

[Conti04] M. Conti, E. Gregori, and G. Maselli. Cooperation Issues in Mobile Ad-hoc Networks. Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), Tokyo, Japan, Mar. 2004.

[Cornelli02] F. Cornelli et al. Choosing Reputable Servents in a P2P Network. In Proceedings of the eleventh international conference onWorldWide Web (WWW'02). ACM, May 2002.

[Corson99] S. Corson and J. Macker. Mobile Ad-hoc Networking: Routing Protocol Performance Issues and Evaluation considerations. Request for comments 2501, IETF, 1999.

[Cox02] Landon P. Cox, Christopher D. Murray, and Brian D. Noble. Pastiche: Making Backup Cheap and Easy. In the 5th Symposium on Operating Systems Design and Implementation (OSDI'02). 2002.

[Cox03] Landon P. Cox and Brian D. Noble. Samsara: Honor Among Thieves in Peer-to-Peer Storage. SOSP '03, Bolton Landing, New York, USA. October 19–22, 2003.

[Das00] S. Das, R. Castaneda, and J. Yan. Simulation based performance evaluation of mobile, ad-hoc network routing protocols. ACM/Baltzer Mobile Networks and Applications (MONET) Journal, pages 179-189, July 2000.

[Dellarocas00] C. Dellarocas. Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. In ACM Conference on Electronic Commerce, pages 150–157, 2000.

[Derhab07] DERHAB Abdelouahid. Localized Protocols for Data and Service Replication in Mobile Ad-hoc Networks. Thèse de doctorat, USTHB, 2007.

[Deutsch96] Peter Deutsch and Jean-Loup Gailly. ZLIB compressed data format specification version 3.3. RFC 1950, Internet Engineering Task Force, May 1996.

[Dingledine00] R. Dingledine, The Free Haven project: Design and deployment of an anonymous secure data haven. Master's thesis, MIT, June 2000.

[Drabkin07] V. Drabkin, R. Friedman, G. Kliot, and M. Segal. RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks. The IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, October 2007.

[Ekstrom02] M. Ekstrom and H. Bjornsson. A rating system for AEC e-bidding that accounts for rater credibility. In Proceedings of the CIB W65 Symposium, pages 753-766, September 2002.

[Friedman01] Friedman and Paul Resnick. The social cost of cheap pseudonyms. Journal of Economics and Management Strategy, 10(2):173–199, 2001.

[Gupta98] P. Gupta and P. Kumar. Critical Power for Asymptotic Connectivity in Wireless Networks. In Stochastic Analysis, Control, Optimization and Applications, Birkhauser, Boston, pages 547–566, 1998.

[Haas02] Z. Haas, J. Halpern, and L. Li. Gossip-Based Ad-hoc Routing. In INFOCOM, pages 1707–1716, June 2002.

[Hauspie03] Michael Hauspie, David Simplot, and Jean Carle. Partition detection in mobile ad−hoc networks. In Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Networks. Mahdia, Tunisia, June 2003.

[Hu03] Y. Hu, D. Johnson and D. Maltz. The dynamic source routing protocol for mobile ad-hoc networks (dsr). http://www.ietf.org/internetdrafts/draft-ietf-manet-dsr-09.txt, April 2003.

[Jain01] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. IEEE Personal Communications, pages 48-57, February 2001.

[Jøsang02] A. Jøsang and R. Ismail. The Beta Reputation System. In Proceedings of the 15th Bled Electronic Commerce Conference, Bled, Slovenia, June 2002.

[Kong01] Kong J, Zerfos P, Luo H, et al. Providing robust and ubiquitous security support for mobile ad-hoc networks. In Proceedings of International Conference on Network Protocols (ICNP), November 2001.

[Lee03] S.-J. Lee, E.M. Belding-Royer, and C.E. Perkins. Scalability Study of the Ad-hoc On-Demand Distance Vector Routing Protocol. In International Journal of Network Management, Volume 13, Issue 2, Pages: 97 – 114, March/April 2003.

[Liao01] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu. Grid: a fully location-aware routing protocol for mobile ad hoc networks. Telecommunication Systems,18:6184, 2001.

[Lillibridge03] M. Lillibridge, S. Elnikety, A. Birrell, M.Burrows, and M. Isard, A Cooperative Internet Backup Scheme, In Proceedings of the 2003 Usenix Annual Technical Conference (General Track), pp. 29-41, San Antonio, Texas, June 2003.

[Liu04] Liu J, Issarny V. Enhanced reputation mechanism for mobile ad-hoc networks. In Proceedings of 2nd International Conference on Trust Management, March 2004.

[Liu06] Jinshan Liu. Découverte de services sensible à la qualité de service dans les environnements de l'informatique diffuse. Thèse de doctorat. Université de VERSAILLES. Juillet 2006.

[Marias06] G. F. Marias, P. Georgiadis, D. Flitzanis and K. Mandalas. Cooperation enforcement schemes for MANETs: A survey. Wireless Communications & Mobile Computing, Volume 6, Issue 3, Wireless Network Security, Pages: 319 - 332. May 2006.

[Marti00] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad-hoc networks. In Proceedings of MOBICOM 2000, pages 255-265, 2000.

[Michiardi02] Michiardi P and Molva R. CORE: a COllaborative Reputation mechanism to enforce node cooperation in mobile ad-hoc networks. In Proceedings of 6th IFIP Communication and Multimedia Security Conference, September 2002.

[Montenegro02] G. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable (sucv) identifiers and addresses. NDSS'02, February 2002.

[Muhlethaler02] P. Muhlethaler. 802.11 et les réseaux sans fil. Edition Eyrolles, 2002.

[Mui02] Mui, L., Mohtashemi, M., and Halberstadt, A. A computational model of trust and reputation. In Proceedings of the 35th HICSS. 2002.

[Ni99] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad-hoc network. In Proc. MobiCom'99, pages 151–162, Seattle WA, August 1999.

[NS2] Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, The ns Manual. February 27, 2006. http://www.isi.edu/nsnam/ns/ns-documentation.

[Qi04] Qi He, Dapeng Wu and Pradeep Khosla. SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks. In IEEE Wireless Communications and Networking Conference (WCNC 2004), Atlanta, GA, USA, March 2004.

[Quercia06] D. Quercia, S. Hailes and L. Capra. B-trust: Bayesian Trust Framework for Pervasive Computing. In Proc. of the 4th International Conference on Trust Management (iTrust), pages 298-312. Pisa, Italy. May 2006.

[Raj91] Raj Jain. The Art of Computer Systems Performance Analysis. John Wiley & Sons, New York, edition 1991.

[Ratnasamy02] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage in sensor-nets. In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.

[Refaei07] Mohamed Tamer A. Refaei. Adaptation in Reputation Management Systems for Ad-hoc Networks. Thèse de doctorat. Arlington, Virginia. Février, 2007.

[Shamir79] Shamir A. How to share a secret. Communications of the ACM; 22(11):612–613. 1979.

[Srinivasan03] V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao. Cooperation in wireless ad-hoc networks. In Proceedings of IEEE Infocom, 2003.

[Thanedar04] Vineet Thanedar, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. A lightweight content replication scheme for mobile ad-hoc environments. In NETWORKING, pages 125–136, 2004.

[Tsudik92] Gene Tsudik. Message authentication with one-way hash functions. ACM Computer Communication Review, 22(5):29–38, 1992.

[Wang02] K. Wang and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. In Proceedings of IEEE INFOCOM. 2002.

[Weiser91] M. Weiser. The computer for the twenty-first century. In Scientific American, p 99-104, 1991.

[Whitby04] Andrew Whitby, Audun Jøsang, and Jadwiga Indulska. Filtering Out Unfair Ratings in Bayesian Reputation Systems. In the Proceedings of the Workshop on Trust in Agent Societies, at the Autonomous Agents and Multi Agent Systems Conference (AAMAS2004), New York, July 2004.

[Yi03] Yi S and Kravets R. MOCA: Mobile certificate authority for wireless ad-hoc networks. In Proceedings of 2nd Annual PKI Research Workshop (PKI03), April 2003.

[Yoon03] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. Infocom 2003, 2003.

[Yu02] Yu, B. and Singh, M. P. An evidential model of distributed reputation management. In Proceedings of ACM AAMAS. 2002.

[Yu03] B. Yu and M. Singh. Detecting Deception in Reputation Management. In Proceedings of the Second Int. Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS), pages 73–80. ACM, 2003.

[Yu05] Hao Yu, Hossam Hassanein and Patrick Martin. Cluster-based replication for large-scale mobile ad-hoc networks. In International Conference on Wireless Networks, Communications and Mobile Computing, pages 552–557, June 2005.

[Zhong03] Zhong S, Chen J, Yang R. Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks. In Proceedings of IEEE INFOCOM2003, April 2003.

[Zhou99] Zhou L, Haas Z. Securing ad-hoc networks. IEEE Network 13(6): 24–30. 1999.

Résumé

Les protocoles de réplication de données proposés dans la littérature pour les MANETs sont conçus pour des MANETs coopératifs. Pour un déploiement dans les MANETs non coopératifs, ces protocoles doivent être sécurisés. Dans ce travail de recherche, nous présentons un état de l'art des mécanismes de gestion de la confiance dans les services collaboratifs entre mobiles mutuellement suspicieux, suivi d'une étude comparative. Nous proposons ensuite un nouveau mécanisme de coopération basé-réputation pour les MANETs qui permet de détecter et d'isoler les nœuds non coopératifs de telle sorte que les nœuds malicieux sont ignorés et les nœuds égoïstes sont encouragés à participer à l'exécution du service de stockage s'ils veulent exploiter les ressources du réseau. Nous validons notre solution à l'aide d'une étude d'évaluation des performances.

Mots clés : *coopération, réputation, rémunération, égoïsme, MANETs, simulation, NS2*

Abstract

Data replication protocols for MANETs proposed in the literature are conceived for cooperative MANETs. For a deployment in no-cooperative MANETs, these protocols must be secured. In this research work, we present a state of the art of the cooperation enforcement schemes for MANETs, followed by a comparative study. We propose then a new reputation-based mechanism for MANETS which makes it possible to detect and isolate non-cooperatives nodes so that malicious nodes are ignored and selfish nodes are encouraged to take part in the execution of the replication service if they want to exploit the networks' resources. We validate our proposal with a performance evaluation study.

Key words: *cooperation, reputation, incentive, selfish, MANETs, simulation, NS2*