THÈSE DE DOCTORAT

DE L'UNIVERSITÉ PARIS-SUD 11

SPÉCIALITÉ INFORMATIQUE

Présentée et soutenue publiquement par

Ignacy GAWĘDZKI

le 29 septembre 2008

ALGORITHMES DISTRIBUÉS POUR LA SÉCURITÉ ET LA QUALITÉ DE SERVICE DANS LES RÉSEAUX AD HOC MOBILES

Sous la direction de Khaldoun AL AGHA

Jury

Rapporteurs :

Refik MOLVA	Professeur à EURÉCOM
Fabrice VALOIS	Professeur à l'INSA Lyon

Examinateurs :

Khaldoun AL AGHA	Professeur à l'université Paris-Sud 11
Joffroy BEAUQUIER	Professeur à l'université Paris-Sud 11
Walter GROTE	Professeur à l'UTFSM (Chili)
Serge HETHUIN	Responsable de Secured Wireless Products (SWP), Thales Communications
Philippe JACQUET	Directeur de Recherche à l'INRIA

Laboratoire de Recherche en Informatique, UMR CNRS 8623
Université Paris-Sud 11, bât. 490, F-91405 Orsay Cedex, France

Résumé

La question du routage dans les réseaux ad hoc mobiles est un domaine de recherche très actif ces dernières années. Les protocoles arrivant aujourd'hui à maturité ont été mis au point dans l'hypothèse qu'aucun des dispositifs prenant part au fonctionnement du réseau n'est malveillant. Pourtant il existe de nombreuses applications où une telle hypothèse n'est pas vérifiée et de nombreux moyens existent pour saper le bon fonctionnement du réseau.

Dans cette thèse nous nous sommes intéressés à la mise en place d'une solution ayant pour but de rendre les protocoles de routage proactifs résistants à l'action d'entités malveillantes. Il s'est agit d'enrichir le protocole de routage afin de permettre aux nœuds d'effectuer de façon distribuée une surveillance du réseau et de fournir une mesure de la menace représentée par chacun des autres nœuds. De ces mesures est ensuite extraite une métrique de qualité de service qui est prise en compte par le protocole de routage. L'effet recherché par l'incorporation de cette métrique est de pouvoir contourner les nœuds qui sont les plus suspects d'après les différentes méthodes de détection utilisées.

Nous proposons en particulier de détecter la perte, intentionnelle ou non, de paquets de données, qui demeure une façon très simple de nuire au bon fonctionnement du réseau, qu'elle soit motivée par l'égoïsme ou non. La détection est réalisée par une vérification distribuée du principe de conservation de flot, basée sur l'échange de compteurs de paquets entre les nœuds voisins. Nous proposons également une méthode de diffusion de ces valeurs qui permet un meilleur passage à l'échelle. Les résultats de cette détection ne pouvant pas servir directement pour incriminer un nœud, ils ne sont utilisés que pour maintenir un degré de suspicion local envers les autres. Les degrés de suspicion locaux sont ensuite diffusés dans le réseau et combinés entre eux pour former un degré de suspicion global envers chaque nœud, qui sert de métrique de qualité de service.

L'application de la solution au protocole OLSR est décrite et ses performances sont évaluées par simulation. Nous observons notamment que la solution permet de diminuer l'impact des nœuds malveillants de façon notable et que malgré le surcoût de contrôle considérable par rapport à l'utilisation d'OLSR classique, la part supplémentaire de capacité du médium utilisée reste faible.

Nous présentons enfin la plateforme de mise en œuvre expérimentale du routage OLSR avec qualité de service et sécurité. Notre objectif est de pouvoir à la fois faire fonctionner nos solutions dans des mises en place réelles et de pouvoir rapidement déceler les problèmes associés à l'utilisation de matériel courant disponible pour le grand public.

Mots-clefs : réseaux ad hoc mobiles, routage, sécurité, qualité de service, pertes intentionnelles, conservation de flot, degré de suspicion, OLSR, Qolyester.

PH.D. THESIS

UNIVERSITY OF PARIS-SUD 11
COMPUTER SCIENCE

by

Ignacy GAWĘDZKI

September 29th, 2008

DISTRIBUTED ALGORITHMS FOR SECURITY AND QUALITY OF SERVICE IN MOBILE AD HOC NETWORKS

Advisor: Prof. Khaldoun AL AGHA

Abstract

Research in the field of routing for mobile ad hoc networks has been very active in the past couple of years. Currently most mature protocols have been designed with the assumption that no misbehaving entity is present in the network. There are however many applications in which such an assumption is not acceptable and there are many ways by which it is possible to harm the network operation.

In this work we have focused on designing a solution aiming to render proactive routing protocols resilient to the action of malicious nodes. It is mainly an augmentation of the routing protocol in order to enable nodes to perform a distributed monitoring of the network and provide an assessment of the threat of other nodes. This assessment is then used to construct a quality of service metric which is taken into account in the routing protocol. The effect looked after by the use of that metric is that of the avoidance of the nodes that are the most suspected of threatening the network, according to the monitoring methods employed.

We propose in particular to detect data packet loss, be it intentional or not, which remains an easy way to damage proper network functioning, regardless of the motivations of that node. The detection is achieved by a distributed verification of flow conservation based on the exchange of data packet counter values between neighboring nodes. We also propose a scalable method of diffusing these values. Since the results of this detection cannot be used directly to accuse a node of misbehaving, they are only used to maintain a local degree of distrust in other nodes. Local degrees of distrust are then diffused in the network and combined into a global degree of distrust in each node, which in turn is used as a quality of service metric.

The application of that method to the OLSR protocol is described and its performances are evaluated by simulations. We note among others that the proposed solution mitigates the impact of malicious nodes pretty well and that despite the considerable control overhead compared to classical OLSR, the additional part of used medium capacity is small.

Finally, we present our experimental platform, which is an implementation of OLSR with quality of service and security features. The main goal is to be able to actually experiment with real setups and quickly find problems related to the use of commercial off-the-shelf hardware.

Keywords: mobile ad hoc networks, routing, security, quality of service, intentional packet dropping, flow conservation, degree of distrust, OLSR, Qolyester.

Remerciements

Je voudrais adresser mes remerciements en premier lieu à Khaldoun Al Agha, mon directeur de thèse, dont l'expérience et le sérieux n'ont d'égal que sa disponibilité et sa gentillesse. La confiance qu'il a toujours su me témoigner durant ces années ainsi que les précieux conseils qu'il a su me prodiguer aux moments cruciaux ont été pour moi essentiels dans mon travail.

Je remercie Refik Molva et Fabrice Valois qui m'ont fait l'honneur d'être rapporteurs de cette thèse. Je remercie également Joffroy Beauquier, Walter Grote, Serge Hethuin et Philippe Jacquet d'avoir accepté de participer à mon jury.

Je voudrais remercier mes collègues de l'équipe Réseaux Mobiles du LRI, membres permanents, thésards et stagiaires, pour l'ambiance amicale et studieuse qu'ils savent entretenir et avec qui les discussions sont toujours passionnantes.

J'aimerais aussi exprimer ma gratitude à Thierry Géraud, Akim Demaille et Didier Verna, ainsi qu'à tous les anciens du LRDE qui m'ont accordé leur confiance et leur soutien et qui m'ont donné une première occasion de me frotter à la recherche en informatique, dans le cadre des mes études à l'EPITA.

Je remercie mes parents qui m'ont toujours soutenu à plus d'un titre et avec un enthousiasme sans égal dans tous mes projets de prolongement de mes études (rassurez-vous, cette fois-ci c'est vraiment fini !). Je remercie également Dominika pour sa patience et son soutien sans faille dans toutes mes entreprises. Je remercie aussi mes amis les Thésards : Kim, Sylvain, mon vieux binôme pragmatique Guillaume, Matthieu, Bertrand, Sammy, Cédric, Laurence, Adrien et les autres pour leur soutien, leurs critiques constructives et leurs pauses café. Merci aussi à mes amis de la Taverne et d'ailleurs : Alexandre, Clément, Dominique, Emmanuel, François, Joël, Julien, Mariusz, Pascal et Pierre avec lesquels j'ai toujours un immense plaisir à partager les meilleurs moments.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Un réseau ad hoc [1] est un ensemble d'appareils communiquants, ou *nœuds*, chacun muni d'une ou plusieurs interfaces réseau sans fil. Ces nœuds peuvent être mobiles ou fixes et leurs ressources en termes d'autonomie et de capacité de traitement peuvent être limitées. La limitation de l'autonomie peut entraîner une présence seulement transitoire de chaque nœud dans le réseau.

La caractéristique principale d'un réseau ad hoc est qu'un nœud peut solliciter d'autres nœuds comme lui dans son voisinage pour retransmettre des paquets de données dont le destinataire serait hors de sa propre portée. Chaque participant peut donc être à la fois hôte et routeur. La mobilité des nœuds, ainsi que leur présence transitoire, fait que la topologie du réseau est dynamique et que les chemins empruntés par les paquets changent eux aussi au cours du temps. Ceci est une deuxième caractéristique importante des réseaux ad hoc, qui a requis l'élaboration de nouveaux protocoles de routage, puisque les solutions pour réseaux fixes se sont avérées inadaptées.

Les réseaux ad hoc sont particulièrement adaptés pour des applications dans des situations où la mise en place d'une infrastructure planifiée n'est pas envisageable ou simplement inintéressante économiquement. Parmi les exemples de telles applications on trouve les communications entre unités ou soldats sur un champ de bataille, l'organisation des secours dans une zone sinistrée ou encore les extensions de couverture des fournisseurs d'accès à Internet.

Une troisième caractéristique de ces réseaux est leur vulnérabilité par comparaison aux réseaux fixes. D'une part les communications radio sont sujettes à tous types d'interférences et la nature ouverte du médium pose un problème évident de confidentialité. D'autre part les nœuds eux-mêmes sont physiquement vulnérables et on doit s'attendre à ce que leur fonctionnement puisse être altéré. L'application au champ de bataille est assez exemplaire en termes de menaces auxquelles le bon fonctionnement du réseau est exposé : l'ennemi peut essayer de brouiller les

communications ou capturer des nœuds en vue de déchiffrer les messages. Il peut aussi modifier le comportement des nœud capturés par rapport au protocole afin de saper le fonctionnement du réseau.

Enfin une dernière caractéristique des réseaux ad hoc, partagée avec tout réseau sans fil, est la faible capacité de transmission. En effet, les technologies actuellement disponibles ne permettent pas d'atteindre le même ordre de grandeur de vitesse de transmission que les solutions filaires (que ce soit sur fil de cuivre ou encore moins sur fibre optique). Il est difficile de penser pouvoir atteindre sur un médium radio des performances comparables à celles atteintes sur un lien filaire direct et isolé. Cette limitation de la capacité de transmission vient s'ajouter aux autres limitations inhérentes aux nœuds mobiles et il apparaît clairement que les réseaux ad hoc nécessitent l'utilisation de routage à qualité de service pour permettre l'utilisation de bon nombre d'applications modernes. En effet, les requis en termes de bande passante minimale, débit maximal, taux de perte maximal de ces applications sont autant de contraintes qu'il faut pouvoir satisfaire autant que possible lorsque les ressources sont rares.

1.1 Problématique

Dans cette thèse nous nous sommes intéressés à la double problématique des menaces auxquelles sont exposés les réseaux ad hoc et de la prise en compte de celles-ci dans les solutions de routage avec qualité de service. Il est parmi ces menaces certaines contre lesquelles la protection n'est pas du ressort du protocole de routage, tant elles ont trait à d'autres propriétés du réseau. On peut prendre pour exemple les brouillages du médium physique ou bien le déni de service par saturation de paquets. En revanche, la menace présentée par la présence de nœuds malveillants — qu'ils soient des nœuds capturés par l'ennemi sur le champ de bataille ou des nœuds contrôlés par des personnes malveillantes dans un réseau public — semble pouvoir être à la portée d'une solution logicielle au niveau du protocole de routage. On peut s'attendre à ce que des nœuds tentent de détourner le protocole à leur avantage, notamment en détruisant les paquets de données des autres nœuds au lieu de les retransmettre en direction de leur destination.

Un tel comportement est malveillant, qu'il soit motivé par l'égoïsme ou pas. Par exemple, une personne se servant d'un appareil à autonomie limitée (téléphone mobile, ordinateur portable, PDA, etc) pour prendre part à un réseau ad hoc public pourrait ne vouloir dépenser sa propre énergie que pour ses propres communications, fût-ce au détriment des autres participants. Mais en soi, la destruction intentionnelle des paquets de données des autres est aussi un moyen de nuire au bon fonctionnement du réseau. On peut donc imaginer que la capture d'un nœud ennemi sur un champ de bataille soit intéressante notamment dans la perspective de corrompre son logiciel et réduire l'efficacité du réseau dans son ensemble.

Lorsque des nœuds qui détruisent intentionnellement les paquets des autres sont présents dans le réseau, il se pose deux questions : comment détecter de telles pertes, d'une part et de quelle façon exploiter cette information dans le but de renforcer le fonctionnement du réseau malgré cette présence, d'autre part ? Comme des pertes peuvent être provoquées par plusieurs facteurs (puissance du signal insuffisante, interférences ou collision de trame, congestion, expiration du TTL, absence de route, etc), aucune méthode de détection ne peut raisonnablement prétendre discerner les pertes fortuites des destructions intentionnelles. Ainsi la réponse à la seconde question est loin d'être évidente, puisqu'il ne suffit pas d'exclure les nœuds accusés de perte.

Nous proposons de traiter ces deux questions en présentant une réponse à la première question dont l'application fournit des résultats destinés à être incorporés comme une métrique de qualité de service supplémentaire. Ainsi la réponse à la seconde question comporte une formule pour constituer cette métrique et un algorithme de recherche de chemins qui la minimise.

1.2 Présentation générale

Dans le chapitre suivant, nous présenterons les travaux existants dans le domaine de la détection de pertes et la robustesse des protocoles de routage pour réseaux ad hoc. Nous commencerons par une présentation des principaux protocoles de routage pour réseaux ad hoc mobiles, tels qu'en voie de standardisation par l'IETF [2]. Ces protocoles se divisent en deux familles, les protocoles dits *réactifs* et *proactifs*, dont nous présenterons les principaux représentants. Puis nous verrons les différents problèmes de sécurité qui se posent dans les réseaux sans fil de façon générale et ad hoc plus précisément. Enfin nous passerons en revue les différentes solutions proposées pour pallier à ces problèmes, depuis les plus robustes et les plus lourdes, jusqu'aux approches plus légères appliquées aux protocoles réactifs et proactifs. Nous verrons aussi quels systèmes de réputation ont été proposés pour tenter d'identifier de façon distribuée les nœuds suspects dans le réseau.

Dans le chapitre 3, nous exposerons en détails la solution que nous avons élaborée au cours de cette thèse. L'idée fondamentale est que lors de l'utilisation d'un protocole proactif à état de lien, tel qu'OLSR [3], la protection du seul protocole n'est pas suffisante pour protéger le réseau d'une des attaques les plus simples à mettre en œuvre : la destruction de paquets de données destinés à d'autres nœuds. Pour détecter ces destructions, peu importe qu'elles soient intentionnelles ou non, nous proposons une approche protocolaire, qui exploite la possibilité technique pour les nœuds du réseau de compter le nombre d'octets contenus dans les paquets de données qui transitent sur un lien vers un voisin. En considérant ces compteurs, il est possible de vérifier le principe de conservation de flot, qui devrait s'appliquer à tout réseau dans lequel les

paquets de données ne sont pas perdus. Nous présenterons les problèmes qui se posent lorsque l'on veut pouvoir vérifier ce principe dans un réseau ad hoc mobile ainsi que les solutions que nous proposons. Les résultats de cette vérification, effectuée par chaque nœud sur chacun de ses voisins, ne sont pas utilisés pour exclure les nœuds fautifs directement, comme cela a pu être proposé précédemment. Ils servent seulement à maintenir localement en chaque nœud un degré de suspicion, qui sert à son tour, entre autres, d'élément de base pour le calcul d'une métrique globale de qualité de service sur chaque nœud, destinée à refléter sa fiabilité. Ainsi les nœuds peuvent appliquer des algorithmes de calcul de chemin prenant en compte cette métrique, afin de trouver des routes qui évitent les nœuds considérés comme les moins fiables du point de vue de la retransmission des paquets de données.

Une fois la solution présentée en détails, nous verrons deux problèmes qui se posent lors de son application pratique : la perte des messages de contrôle et le passage à l'échelle. En effet, nous verrons comment il est possible de remédier au fait que les messages de contrôle utilisés par notre solution peuvent être perdus, par l'utilisation d'un protocole de retransmission. Par ailleurs, comme la taille des informations nécessaires au fonctionnement de notre méthode, qui sont ajoutées dans les messages de contrôle, est dépendante de la densité du réseau, nous verrons par quel moyen leur diffusion par des messages de contrôle à la capacité limitée peut être effectuée.

Enfin nous verrons quelles questions se sont posées lorsque nous avons considéré l'application de la méthode au protocole OLSR. Nous avons pu notamment améliorer les heuristiques de décision locales en exploitant les degrés de suspicion locaux.

L'évaluation des performances de cette approche est présentée au chapitre 4. Nous avons notamment mis en œuvre la solution appliquée à OLSR dans le simulateur OPNET et effectué un grand nombre de simulations afin de quantifier les avantages (en termes de robustesse du réseau vis-à-vis de la présence de nœuds malveillants) et désavantages (en termes de surcoût de contrôle) de l'utilisation de notre méthode. Au préalable, nous présenterons en détails le modèle utilisé, tant au niveau des couches MAC et physique qu'au niveau des couches OLSR et applicatives.

Comme l'évaluation de la méthode par simulation n'est pas suffisante en soi pour valider notre approche, nous présenterons au chapitre 5 la plateforme expérimentale maintenue au cours de cette thèse, utilisée pour mettre en œuvre les résultats de nos recherches. Il s'agit d'un logiciel de routage OLSR, développé en C++, en utilisant les paradigmes de programmation générique statique. Son approche modulaire nous permet facilement d'effectuer des modifications et des ajouts de fonctionnalités. Nous présentons ici son architecture générale, les différentes solutions rajoutée pour faciliter sa maintenance ainsi que les problèmes soulevés par sa mise en œuvre.

Enfin le chapitre 6 présente nos conclusions et les différentes perspectives ouvertes à l'issue de cette thèse.

Chapitre 2

État de l'art

Ce chapitre a pour but de présenter les solutions existantes qui nous ont orienté dans nos recherches. Après avoir passé en revue les principaux protocoles de routage pour les réseaux ad hoc, nous présentons les différentes propositions pour combler les défauts en sécurité de la plupart de ces protocoles, ainsi que les schémas développés pour accroître la robustesse des réseaux ad hoc dans des environnements adverses.

2.1 Routage dans les réseaux ad hoc

Les caractéristiques des réseaux ad hoc mobiles qui les différencient des réseaux filaires classiques sont autant de raisons pour lesquelles les protocoles de routage pour ces derniers ne sont pas utilisables directement lorsque les liaisons sont peu fiables et la topologie est dynamique. En règle générale, les protocoles de routage s'appuient sur l'échange de messages de contrôle entre les nœuds du réseau pour acquérir les informations nécessaires à leur fonctionnement. On distingue aujourd'hui deux familles principales de protocoles, qui correspondent à deux approches fondamentalement différentes de l'utilisation de la signalisation.

La première famille est celle des protocoles dits *réactifs* dans lesquels les nœuds ne maintiennent pas d'information topologique a priori. Les routes sont recherchées à la demande, lorsqu'un paquet de données est destiné à un nœud pour lequel aucun chemin n'est connu.

La seconde famille est celle des protocoles *proactifs* qui, contrairement aux protocoles réactifs, maintiennent des informations topologiques tout au long de la participation du nœud dans le réseau. Ces informations sont utilisées pour disposer à tout moment d'un chemin vers toute destination possible dans le réseau, sans nécessiter d'échange de messages de contrôle supplémentaires lors de l'envoi d'un paquet de données. Dans ces protocoles, le routage par la source, tout comme le routage saut-par-saut sont possibles.

Dans les paragraphes suivants, nous passons en revue les principales propositions de protocoles pour chacune des familles. Il s'agit de quatre protocoles faisant l'objet d'une publication en tant que RFC par le groupe de travail Manet de l'IETF.

2.1.1 Protocoles réactifs

Dynamic Source Routing (DSR)

Dans le protocole DSR [4], RFC 4728 [5], un nœud source qui doit envoyer un paquet vers une destination doit d'abord disposer d'un chemin vers cette dernière. Fondamentalement, un chemin est acquis par l'envoi de messages de contrôle *Route Request* en inondation sur le réseau : tout nœud qui reçoit un tel message est tenu de rajouter son adresse dans celui-ci et de le retransmettre par diffusion à tous ces voisins directs, à condition qu'il ne l'a pas déjà traité (mécanisme illustré sur la figure 2.1). Pour détecter les doublons, les messages contiennent un numéro de séquence et chaque nœud maintient une table de derniers messages traités.

Lorsque la destination recherchée reçoit un message *Route Request*, elle dispose alors de la liste des nœuds intermédiaires successifs par lesquels il est passé, qui constituent un chemin valide depuis la source. Elle doit alors renvoyer ce chemin dans un message *Route Reply* à destination de la source. Lorsque la source obtient cette réponse, elle peut envoyer des paquets de données en utilisant le routage par la source : le chemin que le paquet doit emprunter est inclus dans son en-tête.

Si la destination ne connaît pas de chemin vers la source lorsqu'elle doit renvoyer un message *Route Reply*, alors elle doit elle aussi diffuser un message *Route Request* à destination de la source pour en trouver un, mais pour éviter de boucler indéfiniment, elle ajoute à celui-ci le message *Route Reply*. Ainsi, quand la source reçoit ce double message, elle dispose d'un chemin vers la destination et peut renvoyer un *Route Reply* vers la destination en routage par la source.

Quand un nœud intermédiaire retransmet un message *Route Reply* et que sa propre adresse fait partie du chemin contenu dans le message, alors il rajoute le chemin depuis lui-même vers la destination dans sa mémoire cache de chemins. Ainsi un message *Route Reply* permet de remplir les caches de tous les nœuds intermédiaires.

Les entrées du cache de chemins peuvent être utilisées à tout moment où un chemin vers une destination est requis. Ainsi, lorsqu'un nœud reçoit un message *Route Request* recherchant une destination pour laquelle il dispose d'un chemin dans son cache, il peut renvoyer directement un message *Route Reply* contenant son chemin auquel il aura préalablement préfixé la liste des nœuds intermédiaires que la requête a traversés jusque là, à condition que le chemin résultant ne comporte pas de cycle.

Lors des transmissions unicast, les nœuds peuvent tester la joignabilité d'un voisin et détecter

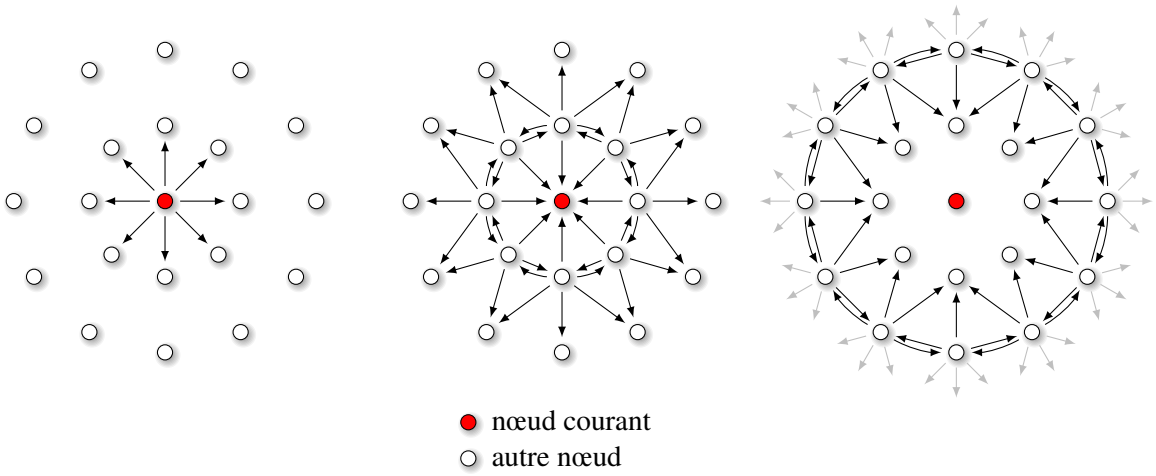


FIG. 2.1: Inondation sur le réseau

lorsque le lien vers celui-ci est rompu. Ainsi, lorsque la transmission d'un paquet routé par la source échoue, un nœud intermédiaire doit en avertir la source par l'envoi d'un message *Route Error* contenant l'en-tête du paquet qui a provoqué l'erreur, ainsi que le lien rompu. Il peut aussi, s'il dispose d'un chemin valide (donc ne traversant pas le lien rompu) vers la destination du paquet, retransmettre le paquet en remplaçant le chemin dans son en-tête par le chemin valide. Le nombre de tels « sauvetages » est limité par un compteur dans l'en-tête du paquet. Lorsqu'un nœud reçoit un message *Route Error*, il doit retirer tous les chemins de son cache qui passent par le lien rompu. Quand la source reçoit le message, elle peut soit utiliser un autre chemin de son cache ou bien rechercher un nouveau chemin par l'émission d'un message *Route Request*.

Dans la spécification du protocole [5], il est prévu l'utilisation d'identifiants de flot (*Flow ID*) qui permettent d'éviter le routage par la source tant que la topologie du réseau ne change pas. Au lieu d'envoyer tous ses paquets correspondant à une même connexion en routage par la source, un nœud peut attribuer un identifiant à ce flux et ainsi demander aux nœuds intermédiaires de rajouter le prochain saut du chemin dans leur table de routage. Chaque nœud intermédiaire devient donc capable de router le paquet en ne consultant que le triplet (source, destination, identifiant) et la source peut éviter de mentionner le chemin complet dans les en-têtes des messages : seul le paquet établissant ces routes doit le contenir. Lorsque la route courante devient invalide, la source peut renvoyer un nouveau paquet contenant le chemin complet et demander aux nœuds d'établir une nouvelle route.

Toutes les données maintenues par les nœuds sont périssables et doivent être rafraîchies périodiquement, il en va donc ainsi pour les chemins dans les caches et les routes dans les nœuds intermédiaires.

Ad hoc On-demand Distance Vector (AODV)

Dans le protocole AODV [6], RFC 3561 [7], des messages de recherche de chemin *RREQ* sont diffusés par inondation sur le réseau et des messages de réponse *RREP* sont renvoyés vers la source de manière analogue à DSR. Seulement au lieu d'envoyer ensuite les paquets de données en routage par la source, les tables de routage des nœuds intermédiaires qui ont traité les messages *RREQ* et *RREP* sont mises à jour afin de permettre le routage saut-par-saut.

L'idée est que lorsqu'un nœud décide de retransmettre un message contenant des informations de chemin pour un nœud — y compris un message *RREQ* qui a déjà une liste de nœuds intermédiaires traversés jusque là — il peut en profiter pour mettre à jour ses propres informations de routage. Outre un nœud suivant et un nombre de sauts associés à une destination, on maintient aussi un numéro de séquence dont le rôle est de s'assurer que les informations contenues dans les messages de contrôle sont plus récentes que les informations locales.

Tout comme dans le protocole DSR, un nœud intermédiaire peut aussi décider de renvoyer un message *RREP* vers la source s'il dispose d'un chemin au numéro de séquence supérieur à celui connu de cette dernière. Ainsi les inondations se trouvent considérablement réduites lorsque la topologie du réseau est stable.

La rupture d'un lien peut être détectée de plusieurs façons différentes, tout comme dans le protocole DSR : accusés de réception de la couche MAC ; accusés de reception explicites dans le protocole ; écoute de la retransmission par le nœud suivant du paquet retransmis. Suite à la détection d'une rupture de lien, un message *RERR* est transmis à destination de tous les nœuds prédécesseurs d'un chemin connu passant par le lien rompu. Ce message sert pour chaque nœud qui le traite à invalider les chemins concernés et pour la source à relancer une recherche de chemin si aucun autre n'est connu. Pour pouvoir envoyer correctement ces messages *RERR*, un nœud doit maintenir pour chaque chemin connu une liste des prédécesseurs, en plus de la destination, du prochain saut, du nombre de sauts restants et du numéro de séquence.

Diverses fonctionnalités optionnelles sont prévues, comme l'utilisation de plusieurs interfaces simultanément, l'agrégation d'adresses dans un sous-réseau, etc.

Dynamic MANET On-demand (DYMO)

DYMO [8] est une évolution d'AODV qui vise à rendre celui-ci plus souple par l'utilisation d'un format de message extensible [9] en voie de standardisation par l'IETF. La gestion de ce nouveau format de message permet un traitement générique des information additionnelles de type inconnu présentes dans certains messages.

2.1.2 Protocoles proactifs

Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)

Le protocole TBRPF [10], RFC 3684 [11], fonctionne en deux phases : détection de voisinage puis diffusion d'informations topologiques. La première phase consiste pour chaque nœud à détecter ses voisins directs et contrôler l'état des liens vers ceux-ci. La seconde, a pour but de faire acquérir par chaque nœud l'information nécessaire sur la topologie du réseau lui permettant de calculer des chemins vers tous les nœuds du réseau.

La découverte de voisinage se fait à l'aide de messages *HELLO* qui servent à chaque paire de nœuds d'établir si le lien entre eux deux est symétrique, asymétrique ou rompu. Pour réduire le surcoût de contrôle, seules les informations différentielles sont nécessaires, c'est à dire que chaque nœud ne diffuse à ses voisins que les changements détectés récemment.

Chaque nœud maintient en permanence un graphe topologique *TG* (*topology graph*) construit avec les informations topologiques reçues de ses voisins. Initialement, un nœud ajoute dans *TG* les liens qu'il détecte comme existant entre lui et ses voisins. Périodiquement, chaque nœud diffuse à tous ses voisins un arbre *RT* (*reported subtree*) contenant les liens de *TG* dont l'une des extrémités est considérée comme appartenant à l'ensemble des nœuds *RN* (*reported nodes*) construit comme suit :

- un voisin j du nœud courant i appartient à *RN* si d'après *TG*, i est le prochain saut depuis un autre voisin sur le chemin le plus court vers j ;
- un nœud k non-voisin de i appartient à *RN* si d'après *TG* le prochain saut depuis i sur le chemin le plus court vers k est dans *RN*.

Pour limiter le surcoût de contrôle, les arbres *RT* complets sont diffusés avec une période assez longue alors que les changements apportés à *RT* le sont plus fréquemment. Ce sont les diffusions de *RT* qui constituent les informations topologiques à partir desquelles les nœuds maintiennent leur propre *TG*.

Comme il n'y a pas de véritable diffusion d'un message à plus d'un saut, les numéros de séquence sont inutiles. Chaque nœud diffuse de l'information qu'il maintient lui-même en se basant sur l'information analogue maintenue par ses voisins eux-mêmes.

La spécification prévoit l'inclusion d'informations supplémentaires dans les messages topologiques afin de permettre diverses variétés de calcul de plus court chemin, ainsi que le routage avec qualité de service.

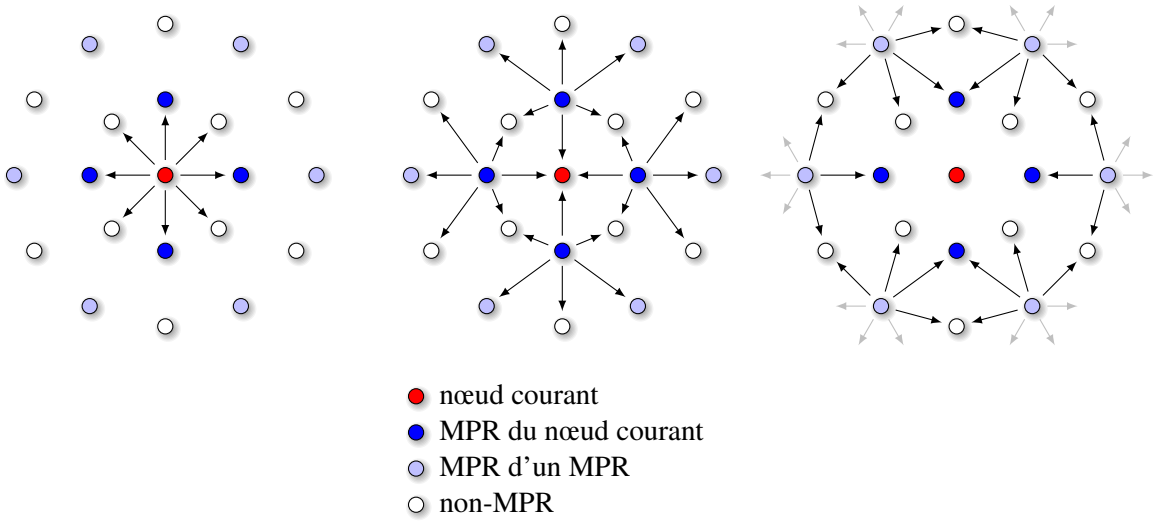


FIG. 2.2: Diffusion optimisée sur le réseau

Optimized Link-State Routing (OLSR)

Comme son nom l'indique, le protocole OLSR [12], RFC 3626 [3], est une version optimisée d'une sous-famille de protocoles de routage bien connue : le routage par état de liens (en anglais *link-state routing*) [13]. Ici aussi on distingue deux phases, la découverte de voisinage et la diffusion d'informations topologiques. En plus de permettre le contrôle de l'état des liens entre le nœud courant et ses voisins, la première phase, basée sur l'échange périodique de messages *HELLO*, permet à chaque nœud d'acquérir une topologie locale totale, c'est-à-dire tous les liens entre lui-même et ses voisins (les voisins à un saut) ainsi que ceux entre ces voisins et leurs propres voisins (les voisins à deux sauts).

La diffusion de messages à tous les nœuds du réseau est ici optimisée, au sens où seul un sous-ensemble des voisins à un saut sont autorisés à retransmettre. En se basant sur sa connaissance de la topologie locale, chaque nœud sélectionne parmi ses voisins ceux qui seront ses *relais multi-point* ou *MPRs* (*multi-point relays*) [14, 15] et qui, en retransmettant ses messages, lui permettront d'atteindre tous ses voisins à deux sauts. Cette restriction dans la diffusion permet de limiter les émissions redondantes, comme l'illustre la figure 2.2.

Les *MPRs* d'un nœuds sont annoncés par ce dernier dans ses messages *HELLO*. Ainsi chaque nœud connaît l'ensemble de ses *MPR-selectors* : ses voisins à un saut qui l'ont choisi comme un de leurs *MPRs*. Chaque nœud dont l'ensemble des *MPR-selectors* n'est pas vide diffuse alors dans tout le réseau des messages *TC* (*topology control*) contenant au moins la liste de ces *MPR-selectors* (et possiblement tous les voisins à un saut symétriques).

La spécification du protocole prévoit des fonctionnalités optionnelles notamment pour le support de plusieurs interfaces de transmission par nœud et le routage vers des réseaux extérieurs au réseau ad hoc. Le mode de diffusion est aussi pensé en prévision de messages supplémentaires destinés à offrir des fonctionnalités supplémentaires non prévues initialement.

OLSRv2

Une version suivante du protocole OLSR est en voie de standardisation par l'IETF sous forme de plusieurs spécifications séparées. Les différentes fonctionnalités du protocoles ont été traitées indépendemment les unes des autres pour permettre leur mise en œuvre dans d'autres protocoles. Ainsi la découverte de voisinage [16] et le format des messages [9] sont spécifiés séparément du protocole en tant que tel [17]. Outre les améliorations apportées par l'utilisation d'un format de message générique (et la possibilité d'inclusion simple d'informations supplémentaires dans les messages qui en découle), certaines fonctionnalités ont pu être remises à plat pour plus de simplicité et plus de cohérence. Par ailleurs, l'utilisation de mécanismes de découverte de voisinage indépendants du reste du protocole permet son extension plus facile tout en s'assurant de la compatibilité descendante.

2.2 Sécurité

Les protocoles présentés au paragraphe précédent ont été élaborés dans l'hypothèse que les nœuds participant au réseau ad hoc coopèrent du mieux qu'ils peuvent. La seule forme de panne naturellement prise en compte est l'arrêt pur et simple de fonctionnement d'un nœud, qui n'est en général pas distingué d'une simple extinction.

Or les réseaux ad hoc sans fil sont potentiellement tout autant que les réseaux filaires, sinon même plus encore, en proie aux adversités. Concernant les réseaux filaires, Il est clair qu'avec l'accroissement de leur taille, leur gestion est devenue décentralisée et il est apparu nécessaire de protéger les réseaux d'une part contre les défaillances et d'autre part contre les attaques d'entités malveillantes. Néanmoins, les solutions apportées aux réseaux filaires se sont limitées au maintien d'une ligne de défense entre le réseau administré et les réseaux voisins, sous la forme de pare-feu, de systèmes de détection d'intrusion et de communications chiffrées.

En ce qui concerne les réseaux ad hoc, la décentralisation doit être extrême puisque les nœuds participants sont possiblement complètement indépendants les uns des autres. Ici la notion de ligne de défense perd son sens et les solutions pour réseaux filaires sont inefficaces. De plus, l'utilisation du médium radio rajoute une forme de vulnérabilité puisque les communications sont exposées aux regards de tous les participants et l'obstacle physique de l'utilisation de

câbles n'est plus là pour décourager les écoutes indésirables.

En pratique, les applications des réseaux ad hoc pouvant se satisfaire d'un tel manque de robustesse sont rares. Comme il est souvent impossible de supposer l'absence totale d'entités malveillantes, il s'est rapidement avéré essentiel d'assurer une protection du fonctionnement des réseaux ad hoc pour permettre leur déploiement.

2.2.1 Attaques possibles

Il a été d'abord nécessaire d'identifier les types d'attaques contre lesquelles on veut se prémunir. Les voici classées par niveau de couche d'abstraction (au sens OSI) à laquelle elles s'appliquent.

Couche physique Une entité malveillante, sans même avoir à prendre part au réseau ad hoc, peut simplement générer de fortes émissions radio ayant pour but de parasiter les transmissions et rendre le fonctionnement normal impossible [18].

Couche liaison Dans l'hypothèse où la couche liaison est égalitaire, un nœud peut très bien saturer le médium en émettant des trames de contrôle ou de données et empêcher ainsi les autres nœuds de communiquer. On parle alors de *déni de service* (*denial of service* ou *DoS* en anglais). Des attaques spécifiques à la couche MAC IEEE 802.11, qui exploitent certains aspects du protocole [19, 20], peuvent aussi provoquer un déni de service

Couche réseau C'est à ce niveau que fonctionnent la plupart des protocoles de routage et que les paquets de données sont retransmis. Un nœud malveillant peut donc détourner le fonctionnement normal du protocole en émettant de fausses informations dans ses messages. Il peut aussi s'attaquer aux paquets de données en les détruisant, en les modifiant ou en les réémettant plus que nécessaire.

Couche application Les attaques à ce niveau sont communes à tous les types de réseau et leur mode opératoire est spécifique à l'application particulière visée. Leur prévention passe inévitablement par la protection des échanges bout à bout par des moyens cryptographiques et l'éventuelle modification des applications.

Cette classification des attaques met en évidence le fait que la seule protection du protocole de routage ne peut apporter de solution à la vulnérabilité des réseaux ad hoc en général. Au mieux, cette protection est efficace contre les attaques à la couche réseau, à condition que le protocole de routage soit aussi en charge de l'acheminement des paquets de données. Les attaques

aux couches inférieures sont du ressort des techniques d'accès au médium pour la couche liaison et des techniques de transmission pour la couche physique. Quant aux attaques à la couche application, elles dépassent largement le domaine des réseaux ad hoc.

Nous avons donc orienté nos travaux vers les solutions de protection de la couche réseau, étant donné que la protection des couches inférieures dépassait largement notre domaine de compétence. Les solutions passées en revue ci-après concernent les attaques du protocole de routage lui-même et/ou les attaques de la retransmission des paquets de données par les nœuds intermédiaires.

2.3 Protection contre les nœuds malveillants

La littérature est riche en propositions de solutions au problème de la résistance des réseaux aux nœuds malveillants. Bien que l'efficacité de celles-ci soit variable, le but ultime recherché à terme est la résistance du fonctionnement des réseaux aux *fautes byzantines*. Ce terme provient du domaine de l'algorithmique répartie [21] et désigne des comportements complètement arbitraires et imprévisibles. Un système réparti résistant aux fautes byzantines est donc résistant aux pires fautes qui soient et donc a fortiori aux comportements malveillants.

2.3.1 Robustesse byzantine

Il a été montré que la résistance des réseaux filaires aux fautes byzantines est envisageable par l'utilisation de chemins redondants et de méthodes cryptographiques de protection [22]. Néanmoins, on observe en général que plus le niveau de protection offert est élevé, plus le surcoût engendré en mémoire, puissance de calcul et/ou messages de contrôle est grand.

Awerbuch et al. présentent un algorithme de routage réactif [23, 24] résistant aux fautes byzantines. Ils supposent l'existence d'un dispositif cryptographique de vérification de l'intégrité d'un message ainsi que de l'authenticité de son émetteur par n'importe quel nœud du réseau, par l'utilisation de signatures à clef publique par exemple.

Le fonctionnement est composé de deux phases : la découverte de chemins et la détection de fautes. Dans la première phase, un nœud source inonde le réseau de *requêtes* signées, contenant les adresses de la source et de la destination ainsi que la liste des poids attribués par la source à différents nœuds du réseau. La signature de cette requête assure son authenticité auprès des nœuds intermédiaires ainsi que de la destination. Lorsque cette dernière reçoit une requête valide, elle inonde le réseau d'une *réponse* contenant elle aussi les adresses source et destination et la liste des poids. Chaque nœud intermédiaire qui reçoit une réponse valide y rajoute son adresse et calcule le poids total du chemin de retour enregistré jusque là, d'après la liste des poids four-

nie par la source. Si c'est la première réponse pour cette requête que le nœud traite, ou bien si le poids est inférieur aux poids des réponses déjà traitées pour la même requête, le nœud signe le tout et retransmet la réponse, sinon il la détruit. Les poids sont utilisés par la source pour trouver des chemins qui passent de préférence par certains nœuds plus que par d'autres, ce qui lui permet d'éviter de passer par des nœuds qu'elle trouve suspects.

Dans la seconde phase, il s'agit pour une source de détecter les fautes byzantines dans l'acheminement des paquets de données. Pour cela, les destinations doivent renvoyer vers la source des *accusés de réception* pour chaque paquet de données valide reçu. Les pertes sont donc détectées par la source par la non-réception de l'accusé et lorsque le taux de celles-là dépasse un seuil prédéfini, la source considère qu'une faute byzantine est commise et déclenche une procédure de détection du nœud fautif. Cette procédure consiste à rajouter dans l'en-tête des paquets suivants une liste de nœuds intermédiaires invités à renvoyer un accusé de réception en plus de la seule destination. Ainsi, la source effectue une recherche par dichotomie du nœud fautif et pour un chemin de longueur n, $\lceil \log n \rceil$ fautes sont nécessaires pour déterminer sa position.

Une fois qu'un nœud byzantin est identifié par la source, son poids est augmenté et un nouveau chemin est recherché pour la destination. Si un chemin ne passant par aucun nœud byzantin identifié jusque là existe, alors il sera trouvé par la procédure de recherche.

Bien que cette approche permette de faire fonctionner un réseau ad hoc en présence de nœuds malveillants, il semble que la solution d'Awerbuch et al. soit bien trop coûteuse en puissance de calcul, occupation mémoire et en messages de contrôle. En effet, les méthodes de chiffrement à clef publique [25] sont connues pour êtres coûteuses et il est clair que l'inondation du réseau par des messages potentiellement volumineux (vu que les requêtes contiennent la liste des poids) n'est pas souhaitable [26]. Ici, le besoin d'avoir des réponses signées par tous les nœuds intermédiaires fait que les optimisations consistant en ce qu'un nœud intermédiaire répond à la source en se substituant à la destination s'il connaît déjà un chemin vers cette dernière ne sont pas envisageables. De plus, les nœuds intermédiaires doivent garder en mémoire les réponses déjà retransmises, afin de pouvoir comparer leur poids à d'éventuelles réponses ultérieures correspondant à la même requête.

2.3.2 Accusés de réception intermédiaires

Watchdog et Pathrater

Afin d'éviter d'avoir à effectuer une recherche explicite du nœud fautif par l'utilisation de sondes, Marti et al. ont proposé une méthode de détection de fautes [27] basée sur l'écoute en *promiscuous mode* sur les interfaces (illustrée sur la figure 2.3). Il s'agit là pour chaque nœud de traiter toutes les trames reçues sur le médium, mêmes si elles ne lui sont pas destinées. En

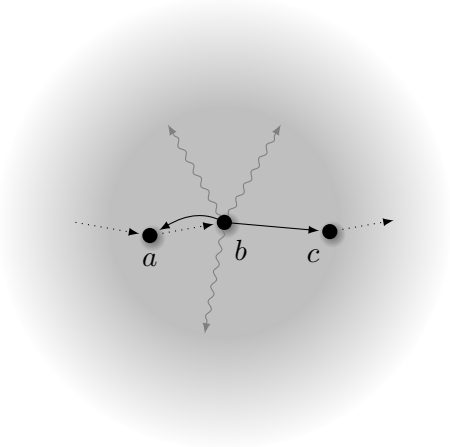


FIG. 2.3: Écoute d'une transmission omnidirectionnelle : a vient de transmettre à b qui retransmet à c et a est capable de vérifier cette transmission.

supposant que tous les nœuds ne possèdent qu'une seule interface réseau sans fil, que toutes les interfaces transmettent sur le même canal et avec la même puissance et que toutes les antennes sont omnidirectionnelles, un nœud est capable de vérifier le contenu de ce que chacun de ses voisins transmet. En particulier, lorsqu'un nœud transmet à son voisin un paquet de données destiné à être retransmis par ce dernier, celui-ci peut effectivement vérifier que son voisin a bien retransmis le bon paquet.

Ainsi, on requiert de chaque nœud de vérifier que ses voisins retransmettent correctement les paquets de données qu'il leur envoie. Cette phase est baptisée ici *watchdog* et consiste en ce que chaque nœud garde en mémoire une copie de chaque paquet retransmis et non destiné à un voisin. À chaque fois qu'une transmission est entendue sur l'interface, le paquet correspondant en mémoire, s'il existe, est retiré. À chaque paquet en mémoire est aussi associée la date de sa retransmission, afin de pouvoir déterminer à quel moment trop de temps s'est écoulé et le paquet doit être considéré comme non-retransmis par le voisin. Lorsque le nombre de non-retransmissions dépasse un certain seuil, une notification est envoyée à la source pour la prévenir de la faute. Les nœuds source peuvent donc maintenir un historique des fautes, sous forme de table de poids par exemple.

L'exploitation des informations recueillies grâce au watchdog est effectuée par un autre élément appelé ici *pathrater* qui maintient des poids pour les différents nœuds qui ont pour vocation de représenter leur fiabilité constatée. La solution proposée est donc utilisée par-dessus un protocole de routage réactif classique (comme DSR par exemple) et permet de disposer d'informations permettant de choisir des chemins fiables.

Cependant, il est clair que cette approche ne fournit pas une résistance aux fautes byzantines.

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

- un nœud n'est pas forcément en mesure de capter les transmissions de ses voisins :
  - si les nœuds n'utilisent pas tous le même canal (en utilisant plusieurs interfaces par exemple) ;
  - si les antennes ne sont pas omnidirectionnelles ;
  - si les nœuds utilisent des puissances d'émission différentes, une transmission d'un voisin vers un autre nœud peut être trop faible ;
  - si une collision arrive au niveau du nœud ;
- un nœud peut capter une transmission d'un voisin qui a échouée au niveau d'un troisième nœud :
  - à cause d'une collision en ce troisième nœud ;
  - si les nœuds utilisent des puissances d'émission différentes et que la puissance d'émission du voisin était trop faible pour le troisième nœud ;
- le watchdog ne permet pas de détecter à coup sûr un paquet détruit à dessein entre deux nœuds malveillants en collusion.

2.3.3 Surcouche de détection

Catch

Le protocole Catch, proposé par Mahajan et al. [28] a pour but de détecter les nœuds égoïstes dans un réseau multi-sauts sans fil. Un nœud égoïste est défini comme un nœud qui détruit intentionnellement les paquets des autres au lieu de les retransmettre. Ce protocole est destiné à être appliqué en parallèle du protocole de routage et fonctionne par l'échange de ses propres paquets de contrôle.

Les nœuds du réseau mesurent le taux de retransmission de leurs paquets par leurs voisins par l'utilisation d'un watchdog (voir paragraphe 2.3.2). De plus, un sous-protocole d'échange de paquets anonymes (*Anonymous Challenge Message* (ACM) *protocol*), pour lesquels l'adresse MAC source de la trame ainsi que l'adresse IP source du paquets sont générées aléatoirement, est utilisé et a pour but de détecter les pertes intentionnelles. Comme les paquets sont anonymes, il est difficile pour celui qui les reçoit de déterminer leur source réelle dans l'hypothèse courante que les antennes utilisées sont omnidirectionnelles. L'idée du sous-protocole ACM est que les voisins *testeurs* d'un nœud *testé* envoient à ce dernier régulièrement des *challenges* anonymes qu'il est censé retransmettre. Si un défaut de retransmission par le testé est constaté par un testeur, alors ce dernier considère le lien comme déficient et ne retransmet plus les paquets en provenance du testé. Comme les challenges sont anonymes, le testé ne peut pas discerner leur source et a tout intérêt à les retransmettre tous au risque de voir un de ses liens critiques

(qui lui permettent de rester connecté au reste du réseau) invalidé. Si le testé ne détruit pas intentionnellement de paquets, alors les résultats du sous-protocole ACM et ceux du watchdog devraient être concordants. Dans le cas où une incohérence entre les deux résultats est constatée par un testeur, alors le testé est considéré par le testeur comme égoïste et ce dernier invalide son lien avec le testé.

Pour pouvoir avertir les autres testeurs du diagnostic posé sur le testé, un testeur utilise un autre sous-protocole : *Anonymous Neighbor Verification* (ANV). Celui-ci se déroule en deux phases : premièrement (ANV Open), les testeurs envoient dans un paquet anonyme destiné au testé (qu'il est censé retransmettre ensuite) le résultat de l'application d'une fonction de hachage à une valeur aléatoire ; deuxièmement (ANV Close), chaque testeur qui considère le testé comme non-égoïste envoie dans un paquet anonyme (à retransmettre également) la valeur initiale à laquelle avait été appliquée la fonction de hachage dans la première phase). Les valeurs diffusées dans la seconde phase servent à chaque testeur à vérifier si tous les autres testeurs sont satisfaits du testé, par l'application de la fonction de hachage sur cette valeur et l'élimination des paquets reçus en première phase et contenant cette valeur hachée. S'il reste des paquets reçus en première phase et non éliminés en seconde phase, alors au moins un testeur considère le testé comme égoïste et par conséquent tous les testeurs s'accordent pour considérer le testé comme tel.

En plus de ces deux sous-protocoles, les testeurs envoient au testé un paquet nommé *Tester Information Exchange* (TIE) à retransmettre et qui contient l'identité du testeur ainsi qu'un bit de signe indiquant quelle part des paquets (données ou ACM) a eu le plus grand taux de retransmission. Les testeurs accumulent les signes des autres testeurs et effectuent le test du signe pour vérifier l'hypothèse selon laquelle les paquets de données et les paquets ACM ont la même probabilité d'être perdus. Enfin, chaque testeur effectue le test de Fisher afin de décider si sa connectivité avec le testé telle que mesurée par le sous-protocole ACM correspond bien à celle mesurée par watchdog.

La faiblesse de ce protocole quant à l'application aux réseaux ad hoc est qu'il est plutôt destiné aux réseaux à topologie statique. En effet, la décision prise par un testeur d'isoler un testé n'est prise qu'à la suite de plusieurs cycles d'ANV Open et ANV Close (appelés *epochs*) dont la durée de chacun est de l'ordre de la minute. Il faut donc que la topologie reste statique à cette échelle de temps, ce qui contredit l'hypothèse d'une topologie dynamique telle que comprise dans le cadre des réseaux ad hoc mobiles.

2.3.4 Routage réactif sécurisé

Dans ce paragraphe, nous passons en revue les principales propositions pour sécuriser les protocoles de routage réactifs.

SRP

Papadimitratos et Haas proposent un protocole de découverte et maintien de chemins réactif protégé contre les fautes byzantines isolées [29], appelé *Secure Routing Protocol* (SRP). Les nœuds malveillants ne sont pas supposés être en collusion les uns avec les autres.

Le fonctionnement de la découverte de chemins est analogue à celle de DSR : des requêtes de recherche de chemin sont inondées dans le réseau et enregistrent la succession de nœuds par lesquels elles passent. Ici on suppose que les nœuds source ont une *association de sécurité* avec chaque nœud destination potentiel : ils sont en mesure de générer une clef de chiffrement symétrique pour chaque flux connue de ses seuls nœuds source et destination. Cette clef sert ensuite à signer tout ou partie des messages en calculant un *code d'authentification de message* (*message authentication code* ou *MAC* en anglais) paramétré par la clef symétrique du flux. Cette méthode est préférable à la signature à clef publique bien plus coûteuse en termes de calcul.

La requête initiale est signée par la source, de façon à ce que la destination puisse détecter les modifications et/ou les falsifications. Lorsque celle-ci la reçoit, elle reprend la requête initiale, la liste des nœuds traversés, rajoute un en-tête de réponse et signe le tout qu'elle revoit par *source routing* en direction de la source. Toute signature est vérifiée par la source et la destination et tout échec de vérification entraîne automatiquement la destruction du message correspondant.

Bien que les adresses des nœuds traversés par la requête ne sont pas protégées contre les manipulations frauduleuses, les réponses dont les routes de retour ne seraient pas traversables n'arrivent pas à la source. Les auteurs comptent donc sur la multiplicité des chemins de la source à la destination (autrement dit, de la multiplicité des requêtes qui arrivent à la destination et pour lesquelles cette dernière renvoie des réponses vers la source) pour assurer que si un chemin sans nœud malveillant existe, il sera ainsi trouvé. Ce mécanisme exclut cependant l'exploitation de l'information dans la mémoire cache des nœuds intermédiaires à des fins d'optimisation (telle que préconisée pour DSR ou AODV). En effet, toutes les requêtes doivent arriver à la destination avant que des réponses puissent être générées, puisqu'elles doivent être signées par celle-ci.

Tout comme dans le protocole DSR, la maintenance de chemins est effectuée par le nœud intermédiaire situé en amont du lien rompu. Il envoie un message de notification de perte de lien en source routing vers la source. Celle-ci vérifie alors que le chemin contenu dans le message (pour le routage) correspond bien au préfixe du chemin pour lequel la rupture de lien est signalée. Si tel n'est pas le cas, la notification est simplement ignorée. Les informations ne sont toutefois

pas protégées davantage et un nœud malveillant peut très bien notifier une rupture de lien fictive. Dans ce cas, les auteurs argumentent que seul le chemin contenant le nœud malveillant est affecté et tant que les nœuds malveillants ne sont pas en collusion, cela permet à la source de trouver un chemin fiable.

SMT

Comme le protocole SRP ne concerne que la découverte de chemins dans le réseau, les mêmes auteurs ont par la suite proposé un protocole d'acheminement de paquets de données [30] qui s'assure que les effets de l'action de nœuds byzantins sont minimisés. Le protocole exploite l'hypothèse selon laquelle dans un réseau ad hoc, il existe plusieurs chemins possibles entre la source et la destination. Pour acheminer les paquets de façon fiable, une source les décompose en plusieurs morceaux redondants et achemine chacun d'entre eux par source routing selon un des chemins possibles. L'idée est qu'il suffit que la destination dispose d'une partie des morceaux pour reconstituer le paquet originel et qu'elle renvoie en direction de la source des accusés de réception par source routing pour chacun des morceaux reçus. La source est donc en mesure de déterminer quels chemins ont fonctionné et potentiellement quels autres ont failli. Ces informations servent ensuite à la source pour maintenir un degré de fiabilité de chacun des chemins et ainsi éviter les chemins pour lesquels des pertes ont été constatées. Ici la seule association de sécurité nécessaire est entre la source et la destination : les paquets de données et de contrôle sont signés par leur émetteur pour détecter les modifications.

Ni pour le protocole SRP, ni pour SMT, la collusion entre nœuds malveillants n'est parée et il est clair que la présence d'un nombre suffisant de nœuds malveillants dans le réseau, même sans collusion, provoque une dégradation significative des performances.

Ariadne

Pour éviter les nœuds malveillants au moment même de la diffusion des requêtes de découverte de chemins (et pas seulement au retour des réponses), il faut pouvoir authentifier les adresses des nœuds intermédiaires accumulés dans ces requêtes. Hu et al. proposent donc le protocole réactif Ariadne [31] qui requiert de chaque nœud intermédiaire qu'il signe le nouveau paquet avant de le propager. Ainsi lorsque cette signature n'est pas vérifiée dans la requête ou dans la réponse correspondante, cette dernière est détruite.

Bien entendu, l'utilisation du chiffrement à clef publique est très coûteux et la signature par MAC requiert l'établissement préliminaire de clefs de chiffrement symétrique uniques entre toute paire de nœuds dans le réseau. Les auteurs proposent donc l'utilisation de TESLA [32, 33], un procédé qui permet l'authentification de la source d'un message diffusé à condition que les

nœuds ont une vague synchronisation de leurs horloges. Le fonctionnement du protocole permet à la destination (resp. à la source) de vérifier si des requêtes (resp. des réponses) ont été faussées par un nœud intermédiaire et les détruire le cas échéant.

Les ruptures de lien sont signalées par le nœud en amont qui renvoie une notification vers la source en utilisant le préfixe du chemin inclus dans le paquet qui a provoqué l'erreur. Tous les nœuds qui traitent ce message (et a fortiori la source) authentifient son émetteur afin de prendre les mesures qui s'imposent pour invalider le chemin.

Le mécanisme de signalisation des ruptures offre cependant la possibilité pour les nœuds malveillants d'envoyer de fausses notifications de rupture, afin de rompre effectivement le lien en aval. Ceux-ci peuvent aussi tâcher de se placer de façon à prendre part à un maximum de chemins et ainsi provoquer un déni de service.

SAODV

L'approche qui consiste à sécuriser les messages de contrôle d'un protocole réactif a été appliqué à AODV par Zapata et Asokan [34]. Ici la partie constante des paquets est signée successivement par les nœuds intermédiaires et toutes les signatures sont vérifiées par tout le monde. La partie non-constante la plus sensible reste le compteur de sauts dans les requêtes et celui-ci ne peut pas être simplement vérifié par une signature. Par conséquent, les auteurs proposent en plus l'utilisation de deux chaînes de hachage [35] pour vérifier que le compteur de sauts n'a pas été décrémenté abusivement. Si on pose H comme étant la fonction de hachage utilisée et s un nombre aléatoire tiré par la source d'un message, alors celle-ci inclut, en plus du compteur de sauts $hc \leftarrow 0$, le nombre maximal de sauts que le message doit parcourir $mh \leftarrow TTL$, la valeur $H^{mh}(s)$ et la valeur de $h \leftarrow s$. Chaque nœud intermédiaire doit donc vérifier qu'on a bien

$$H^{mh}(s) = H^{mh-hc}(h)$$

et retransmettre le message avec $hc \leftarrow hc + 1$ et $h \leftarrow H(h)$.

Cette protection n'est toutefois pas totale et il est clair qu'un nœud malveillant peut simplement ne pas incrémenter hc et ne pas changer h (pour augmenter le nombre maximal de sauts autorisés de 1) ou, au contraire, incrémenter hc et changer h de façon correspondante afin de réduire le nombre maximal de sauts autorisés.

2.3.5 Routage proactif sécurisé

Dans ce paragraphe, nous passons en revue les principales propositions de protocoles de routage proactifs sécurisés, à vecteur de distance puis à état de lien.

SEAD

Le protocole SEAD, proposé par Hu et al. [36] emploie aussi les chaînes de hachage pour sécuriser et authentifier le contenu des messages de contrôle du protocole à vecteur de distance DSDV sur lequel il est basé. Dans les protocoles à vecteur de distance, les informations contenues dans les tables de routage des nœuds (destination, prochain saut et distance) sont échangées entre voisins directs et servent à mettre à jour les tables de chacun. Pour éviter les problèmes de cycles qui peuvent apparaître suite à la rupture d'un lien, à chaque entrée est en plus associé le numéro de séquence du message provenant de la destination sur lequel l'information se base. Ainsi lors de la réception d'un message depuis un voisin, un nœud met à jour une entrée soit si sa distance est plus grande que celle du voisin augmentée de 1, soit lorsque son numéro de séquence est supérieur.

Il faut donc pouvoir sécuriser à la fois le numéro de séquence et la distance associés à chaque entrée dans un message. Pour ce faire, chaque nœud i calcule une chaîne de hachage

$$h_0^i, h_1^i, \ldots, h_n^i$$

telle que

$$h_k^i = H\left(h_{k-1}^i\right) \quad , \qquad 0 < k \leq n \ .$$

Sachant que la distance à une destination est inférieure strictement à m, la longueur de la chaîne est choisie de façon à être divisible par m et chaque destination i authentifie son message de numéro de séquence $s > 0$ avec la valeur

$$h^i_{(\lfloor \frac{n}{m} \rfloor - s)\cdot m}$$

et lorsqu'un nœud met à jour son entrée à partir de celle d'un voisin (en augmentant la distance d'un saut), il applique la fonction de hachage à la valeur qui authentifie l'information reçue. Ainsi pourvu que le nœud qui reçoit l'information dispose d'une valeur authentique pour la destination et un numéro de séquence inférieur, il suffit pour authentifier la nouvelle valeur d'appliquer la fonction de hachage le bon nombre de fois et de comparer le résultat à l'ancienne valeur.

Les auteurs proposent en outre une méthode un peu plus complexe qui empêche les nœuds de rejouer la valeur reçue d'un voisin et ainsi éviter d'incrémenter la distance lors de la mise à jour de l'entrée dans leur table de routage. Elle consiste à utiliser des arbres de hachage chaînés (*hash tree chains*) qui sont un hybride entre les chaînes de hachage et les arbres de hachage de Merkle [37]. Cette solution implique l'adresse de chaque nœud dans le calcul des valeurs de hachage et

oblige donc celui-ci à fournir le numéro de séquence et une distance incrémentée pour chaque destination. Malheureusement l'utilisation des arbres de hachage chaînés demande bien plus de calcul lors de la génération des valeurs par chaque destination. Les auteurs proposent néanmoins une approche plus économique en termes d'étapes de calcul dans laquelle la probabilité qu'un nœud malveillant arrive à falsifier l'information est maintenue à un niveau faible. Par ailleurs, pour éviter à un noeud qui aurait raté plusieurs numéros de séquence pour une destination d'avoir à effectuer trop de calculs pour authentifier une mise à jour, les auteurs suggèrent l'utilisation de chaînes plus courtes, de longueur m dont la valeur initiale est authentifiée par arbre de hachage.

Bien que SEAD soit une solution intéressante pour sécuriser le protocole DSDV, il n'est pas suffisant pour empêcher les nœuds malveillants d'agir sur les paquets de données. En effet, la retransmission de ces paquets n'est pas assurée par le protocole de routage et un nœud peut facilement les falsifier, les rejouer, les modifier ou simplement les détruire.

SLSP

Papadimitratos et Haas proposent *Secure Link State Protocol* (SLSP) [38], un protocole à état de lien dont ils ont modifié les messages de contrôle afin d'en sécuriser le contenu. Les nœuds sont supposés authentifiables par l'utilisation d'un système de cryptographie à clef publique. Les clefs publiques des nœuds ainsi que leurs certificats (signés par une autorité de certification distribuée) sont diffusés dans le réseau. La restriction du diamètre de la diffusion se fait à l'aide d'une chaîne de hachage, alors que l'authentification du message se fait par vérification de la signature avec la clef publique de l'émetteur.

En plus de ces mesures, les nœuds tentent de détecter les usurpations d'identité au niveau de leurs voisins en maintenant une association entre adresses IP et MAC de ceux-ci. Pour limiter les attaques de type déni de service, les nœuds traitent les paquets à retransmettre selon des priorités modulées en fonction de la fréquence de réception des paquets de chaque émetteur. Ainsi, les émetteurs les plus fréquents voient leurs paquets traités avec une priorité moindre.

En ce qui concerne la protection du protocole, l'utilisation des chaînes de hachage permet juste de limiter le diamètre de diffusion des messages de mise à jour topologique. Rien n'empêche en revanche un nœud de rejouer la valeur de hachage reçue (et donc ne pas incrémenter le compteur de sauts et ne pas décrémenter le TTL) et/ou d'augmenter le compteur de sauts plus que nécessaire. Par ailleurs, tout comme pour le protocole SEAD, les paquets de données ne sont pas protégés contre la falsification, le rejeu, la modification ou la destruction.

Sécurisation d'OLSR

Le protocole OLSR a aussi été l'objet de recherches visant à le sécuriser. Parmi celles-ci on peut noter les travaux de Raffo et al. [39] qui ont proposé de sécuriser le contenu de ses messages de contrôle pour empêcher les nœuds de publier des informations fausses ou périmées. Fourati et al. ont aussi proposé une approche pour garantir la véracité du contenu des messages TC [40] en requérant leur validation par les voisins de leurs émetteurs et leur signature en utilisation la cryptographie à seuil.

Bien que ces approches puissent être efficaces pour protéger le contenu des messages de contrôle en présence de nœuds malveillants, les paquets de données ne sont pas concernés et demeurent vulnérables.

2.3.6 Systèmes de réputation

Lorsque l'observation seule ne permet pas une mesure directe et objective de la malveillance des nœuds, il est nécessaire que chaque nœud maintienne un degré de confiance en chacun des autres nœuds qu'il a observés. La valeur de ce degré de confiance est influencée par les observations de telle façon qu'elle puisse fournir une estimation la plus juste possible de la malveillance du nœud observé.

La littérature sur les systèmes de réputation est très fournie [41, 42, 43] et leur champ d'application est très large. Néanmoins en ce qui concerne le routage pour réseaux ad hoc, on distingue deux courants principaux, l'un ayant pour but de décourager les comportements malveillants et l'autre de les éliminer. Bien que la nuance semble subtile, dans le premier cas il ne s'agit que de mesures d'encouragement aux comportements bienveillants qui ne peuvent pas empêcher l'action des nœuds malveillants. L'idée est que si un nœud veut lui-même pouvoir transmettre ses propres paquets à travers le réseau, il a tout intérêt à afficher ses bonnes intentions en coopérant dans le réseau. Cependant, il est clair qu'un nœud malveillant n'ayant pas spécialement besoin ou envie de transmettre son propre trafic peut très bien avoir un comportement malveillant tout à fait gratuit. On peut donc voir ces systèmes de réputation comme une variante de système d'encouragement, au même titre que le sont les propositions à base de rétribution virtuelles telles que proposées par Buttyán et Hubaux [44, 45] ou Zhong et al. [46].

Dans le second cas, où le système de réputation est mis en place afin d'éliminer les nœuds dont l'action malveillante a été observée, un nœud agissant de façon nuisible et gratuite sera exclu du réseau ou au moins évité du point de vue de la retransmission des paquets. Il s'agit donc bien dans ce cas là de rendre un réseau ad hoc résistant à l'action des nœuds malveillants.

CONFIDANT

Le système CONFIDANT, proposé par Buchegger et Le Boudec [47] a pour but de détecter et exclure les nœuds malveillants dans un réseau ad hoc. Il est composé de plusieurs modules :

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

Dans leur version préliminaire du système, les auteurs ne précisent pas quels sont les critères de décision qu'un nœud a une bonne réputation. Il est juste évoqué le fait que les degrés locaux de malveillance détectée doivent croître avec les observations de fautes et décroître avec le temps afin de permettre aux nœuds de se racheter ou bien aux fautes rares d'avoir peu d'incidence. Par ailleurs, le protocole de routage envisagé est DSR et ses mécanismes réactifs sont exploités pour la recherche de chemins qui évitent une liste de nœuds suspects désignés par la source.

Dans un rapport de recherche ultérieur [48], les auteurs donnent une description plus détaillée du système en précisant la méthode utilisée, basée sur un modèle bayesien de représentation [49, 50], pour la maintenance des réputations. Les nœuds intègrent les observations directes des comportements malveillants et les réputations des voisins (observations indirectes) dans leur degré de réputation des autres nœuds. Une observation indirecte est intégrée à la condition que le degré de confiance du nœud émetteur est suffisant. Ce dernier est maintenu en fonction de la compatibilité des observations indirectes annoncées par le nœud émetteur et les observations directes du nœud courant. Ainsi les annonces des nœuds qui diffèrent trop des observations directes locales sont finalement ignorées, afin de se prémunir contre la diffamation.

Bien que le modèle bayesien semble bien adapté au traitement des réputations et des degrés de confiance, comme le montrent les résultats de simulation présentés par les auteurs, on est en droit de questionner la pertinence de l'utilisation de la détection de type watchdog. De plus, ce système de réputation ne garantit pas ici des degrés de réputation uniformes, ce qui le rend inadapté aux protocoles proactifs qui requièrent des nœuds qu'ils prennent les mêmes décisions quant au routage, afin d'éviter les cycles.

CORE

Le système CORE, proposé par Michiardi et Molva [51], est assez similaire. Il est toutefois décrit de façon plus générique sans doute pour être adapté plus facilement à plusieurs autres cas que l'application proposée au protocole DSR. Ici les nœuds maintiennent tout un ensemble de données sur les comportements des autres nœuds qui servent ensuite à construire des réputations directes (issues des observations directes du comportement attendu d'un voisin) et indirectes (déduites des messages diffusés par les voisins et concernant des nœuds tiers). Les formules de combinaison de ces réputations sont fournies et l'accent est mis sur la nécessité de donner plus d'importance aux comportements passés afin de mieux tolérer les fautes passagères. Il est explicitement indiqué que les réputations indirectes ne peuvent pas être négatives afin de ne pas donner la possibilité aux nœuds malveillants de diffamer les autres.

L'application au protocole DSR proposée fournit une mesure de réputation directe par l'utilisation d'un watchdog pour détecter si les nœuds retransmettent bien les requêtes de chemins, ainsi qu'une mesure de réputation indirecte, lorsque la réponse de chemin liste les nœuds qui ont effectivement retransmis la requête. Par ailleurs, l'utilisation du watchdog est aussi préconisée afin de fournir des réputations directes pour la retransmission des paquets de données. Des réputations indirectes pour ce type de service sont supposées possibles si le protocole de retransmission des paquets de données fournit un message d'accusé de réception listant les nœuds ayant coopéré, ce qui semble dans l'ensemble peu probable.

Les valeurs combinées de réputation sont ensuite disponibles pour par exemple refuser de fournir un service à un nœud considéré comme malveillant, voire de l'exclure de la topologie du protocole de routage.

LARS

Dans un rapport de Hu [52], les systèmes CONFIDANT et CORE sont passés en revue de façon critique et de nombreux points faibles y sont dénoncés. Par ailleurs, ce rapport propose le système LARS qui est une forme de système de réputation où seules les observations de première main sont diffusées. Un mécanisme de chiffrement à clef publique est déployé et pour

éviter la diffamation, chaque diffusion d'observation négative doit nécessairement être signée par la clef privée du nœud incriminé. Pour rendre cette signature possible, la cryptographie à seuil est utilisée. Chaque nœud distribue des parts de sa clef privée à un certain nombre de ses voisins qui collectivement deviennent capable d'effectuer l'opération de signature à sa place.

Ici aussi un système de watchdog est utilisé pour détecter les non-retransmissions ou d'autres fautes. Le protocole de routage préconisé est DSR ou AODV et un système d'envoi d'accusés de réception par la destination pour tout paquet de données est nécessaire. Lorsque la source ne reçoit pas un accusé de réception, elle lance une procédure de *trace* qui consiste en un paquet spécial parcourant le chemin du paquet perdu et demandant à chaque nœud de rediffuser le paquet perdu (qu'il ont gardé en mémoire un certain temps). Cette procédure a pour but de s'assurer que soit le nœud fautif se dénoncera tout seul (en ne rediffusant pas le paquet en direction de la source), soit il sera reconnu comme fautif par ses voisins.

2.4 Conclusion

On peut constater en somme que lorsqu'on s'intéresse à des méthodes de résilience aux nœuds malveillants pour les protocoles de routage proactifs pour réseaux ad hoc, il n'existe pas de solution qui sécurise à la fois le protocole et la retransmission des paquets. Les solutions qui se veulent les plus complètes sont coûteuses et les solutions concernant les protocoles proactifs ne se concentrent que sur la robustesse du protocole lui-même.

Par ailleurs, les fautes de retransmission sont exclusivement détectées par l'utilisation d'un watchdog, malgré tous ses défauts avérés, notamment lorsqu'on se place dans des cas d'utilisation diversifiés (interfaces multiples, antennes directionnelles, contrôle de puissance, etc).

Comme aucune méthode ne semble pouvoir fournir une mesure infaillible de la malveillance de chaque nœud, l'utilisation d'un système de réputation est vivement recommandée. Or l'utilisation d'un protocole de routage proactif à état de lien tel qu'OLSR nécessite que les réputations des nœuds soient uniformes. Autrement dit, il faut nécessairement que tous les nœuds se fassent la même idée de la fiabilité d'un nœud donné, pour peu que cette fiabilité doive servir au calcul de routes fiables. En effet, le routage étant effectué saut par saut, il est essentiel que tous les nœuds aient la même vision de la topologie du réseau, faute de quoi des cycles dans le routage peuvent apparaître.

Dans le chapitre suivant, nous présentons en détail une approche que nous avons proposée pour détecter puis éviter les nœuds du réseau qui perdent le plus de paquets de données. Cette approche s'avère notamment particulièrement bien adaptée à l'application aux protocoles de routage proactifs, pour lesquels une solution de protection des messages de contrôle est déjà mise en place.

Chapitre 3

Résilience aux pertes intempestives

Un des grands avantages des protocoles de routage proactifs et plus particulièrement des protocoles à état de lien est qu'ils peuvent être mis en œuvre plus facilement sur des systèmes qui possèdent déjà une pile protocolaire de niveau 3 (couche Réseau du modèle OSI). Cette dernière est en charge du routage, c'est à dire de trouver la prochaine station par laquelle un paquet de données doit passer pour atteindre sa destination. La pile protocolaire TCP/IP, extrêmement répandue à l'heure actuelle, offre au minimum une fonction de routage saut par saut au niveau 3, basé sur l'utilisation de tables de routage. Une telle table associe à chaque adresse destination l'adresse de la prochaine station vers laquelle un paquet doit être transmis. Un protocole à état de lien peut être mis en œuvre sous la forme d'un processus utilisateur dont l'action, complémentaire à au routage saut par saut, se réduit à maintenir la table de routage de la station sur laquelle il s'exécute.

L'utilisation des mécanismes de routage existants est très avantageuse en termes de performances, puisque la retransmission de niveau 3 est prise en charge par le noyau du système d'exploitation et sa mise en œuvre est optimisée, forte de plusieurs décennies de développement de la pile TCP/IP. Cet avantage devient toutefois un inconvénient face aux nouveaux besoins de fiabilité dans les réseaux ad hoc. En effet, le protocole de routage n'ayant pas lui-même la charge de la retransmission des paquets de données, aucune mesure de protection s'appliquant à lui seul ne résout tous les problèmes possibles.

Parmi les attaques possibles sur un réseau en général, une des plus faciles à mettre en œuvre est la perte intentionnelle et intempestive des paquets de données. Dans le cas des protocoles proactifs, cette attaque cible justement la retransmission, élément indépendant du protocole de routage lui-même. Dans ce chapitre, nous étudions comment, en agissant au niveau du protocole de routage et en utilisant des outils déjà disponibles au niveau du mécanisme de retransmission des paquets de la plupart des systèmes d'exploitation, on peut réduire l'impact de telles

attaques [A, B, C, D, F].

3.1 Vérifier la conservation de flot dans un réseau

Lorsque le protocole de routage est complètement protégé mais n'est pas en charge de la retransmission des paquets de données, on veut pouvoir vérifier que celle-ci est effectuée correctement par tous les nœuds du réseau. Une façon indépendante des couches 1 et 2 du modèle OSI (respectivement couches Physique et Liaison) consiste à considérer les paquets de données traversant les liens comme des flots et à enrichir le protocole de routage afin de vérifier la propriété de conservation de flot dans le graphe topologique. Les systèmes d'exploitation modernes offrent la possibilité d'utiliser des compteurs de trafic réseau au niveau des nœuds. Il s'agit donc de faire échanger l'état de ces compteurs entre les nœuds du réseau et de leur faire effectuer des vérifications sur ceux-ci pour vérifier la conservation de flot.

Une approche similaire pour réseaux fixe a été proposée par Bradley et al. [53], mais elle n'est pas du tout adaptée aux réseaux ad hoc mobiles sans fil. Elle suppose une topologie de réseau statique et une synchronisation des nœuds pour l'échange des compteurs.

On verra par la suite qu'on ne peut en général pas distinguer les pertes intentionnelles de celles qui peuvent survenir de façon inopinée. En outre, on verra aussi qu'on ne peut parfois pas incriminer un seul nœud suite à la détection d'un problème. Cette constatation nous mènera vers la conception d'un système de notation des nœuds, visant en définitive à considérer la fiabilité des nœuds comme une métrique de qualité de service. Une telle approche permet alors de contourner les nœuds à problème autant que possible, tout en ne les excluant jamais. Ceci répond notamment à un des reproches formulés par Hughes et al. [54] visant la méthode de Bradley et al. qui excluait directement les nœuds incriminés, ce qui pouvait donner lieu à des attaques *kamikaze*.

Dans la suite de ce paragraphe, après l'introduction du modèle formel des compteurs, nous présentons des propriétés sur ces compteurs et enfin une méthode pour leur vérification déduite notamment de l'expression de la conservation de flot dans un réseau.

3.1.1 Modèle réseau et compteurs par lien

Pour la suite du propos, nous avons besoin de définir formellement le graphe topologique, les liens et les voisins.

Définition 1 (graphe topologique) *La topologie du réseau est représentée par un graphe* $G = (V, E)$ *avec* V *l'ensemble des stations ou nœuds (les sommets) et* $E : \mathbb{R}^+ \rightarrow V \times V$ *une*

fonction qui à chaque instant t associe un ensemble de couples de sommets (i, j), les liens unidirectionnels existant à cet instant (les arcs).

Définition 2 (lien symétrique) *Pour toute paire de nœuds i et j, on note $\{i, j\}$ un lien symétrique entre i et j, c'est-à-dire l'ensemble des liens asymétriques $\{(i, j), (j, i)\}$. Par extension, on notera E^2 la fonction qui à tout instant t associe l'ensemble des liens symétriques existant à cet instant.*

Définition 3 (voisinage symétrique) *Pour tout nœud i du réseau et tout instant t, soit $\mathcal{N}_i(t)$ l'ensemble des voisins symétriques directs de i, soit*

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \qquad \mathcal{N}_i(t) = \big\{ j : \{i, j\} \in E^2(t) \big\} \tag{3.1}$$

Nous pouvons maintenant définir les compteurs par lien (voir FIG. 3.1).

Définition 4 (entrée/sortie par lien) *Pour tout lien (i, j) et pour tout instant t, soit $I_{ij}(t)$ (respectivement $O_{ij}(t)$) le nombre total d'octets ayant traversé le lien (i, j) dans des paquets non destinés à j (respectivement non originaires de i).*

Formellement, le principe de conservation de flot appliqué à la quantité de données traversant les liens du graphe topologique s'exprime de la façon suivante.

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \qquad \sum_{j \in V} \left(I_{ji}(t) - O_{ij}(t) \right) = 0 \tag{3.2}$$

En se plaçant dans l'hypothèse où les quantités $I_{ij}(t)$ et $O_{ij}(t)$ sont connues de tous les nœuds pour tout lien (i, j) et à tout instant t, alors une perte de paquet, aussi infime soit elle, implique que le principe de conservation de flot dans le réseau n'est pas vérifié. Nous allons voir par la suite comment on peut déduire de cette idée une méthode reposant sur l'extension du protocole de routage existant dans le but de détecter les pertes de trafic.

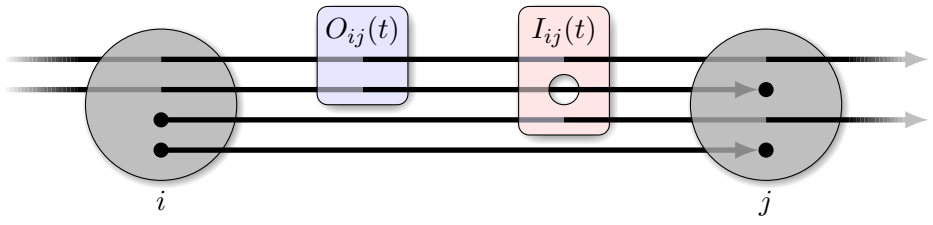


FIG. 3.1: Les compteurs d'entrée et sortie

3.1.2 Compteurs locaux aux nœuds

Dans la réalité, les quantités en jeu dans (3.2) ne sont pas directement connues puisqu'il n'existe pas un oracle indépendant des nœuds qui comptabilise le trafic sur tous les liens. Le seul moyen technique à notre disposition est la possibilité pour les nœuds de comptabiliser ce trafic eux-mêmes. Alors pour chacune des quantités introduite au paragraphe précédent, on ne dispose que de ce qu'on appellera ses deux *vues* par chacune des extrémités du lien.

Définition 5 (vue d'une entrée/sortie par lien) *Pour tout lien (i, j) et pour tout instant t, soient $I^i_{ij}(t)$ et $O^i_{ij}(t)$ (respectivement $I^j_{ij}(t)$ et $O^j_{ij}(t)$) les vues de i (respectivement de j) des quantités $I_{ij}(t)$ et $O_{ij}(t)$.*

On doit donc requérir de chacun des nœuds du réseau de mettre en place des compteurs de paquets, c'est-à-dire des variables en mémoire qui sont incrémentées du nombre d'octets contenus dans chaque paquet de données qui passe par un lien. Supposons pour le moment, toute considération pratique mise de côté, que ces variables sont maintenues dans une zone de mémoire partagée par tous les nœuds. Ainsi tout compteur est disponible pour chaque nœud. On voudrait que tous les nœuds puissent vérifier la conservation de flot en chacun des autres nœuds à tout moment. Se pose alors la question de savoir comment reformuler (3.2) en termes de vues et non plus en termes de $I_{ji}(t)$ et $O_{ij}(t)$ directement. Pour chaque occurrence de $I_{ji}(t)$ dans la formule, on peut choisir d'utiliser $I^i_{ji}(t)$ ou $I^j_{ji}(t)$, de même pour $O_{ij}(t)$ et ses vues $O^i_{ij}(t)$ et $O^j_{ij}(t)$. Les trois différentes formules que l'on obtient ne sont pas équivalentes et aucune d'elle n'est vraiment utilisable seule directement dans la pratique.

Considérons toutefois la version suivante :

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \qquad \sum_{j \in V} \left(I^i_{ji}(t) - O^i_{ij}(t) \right) = 0 \ . \tag{3.3}$$

On remarque qu'elle n'implique que les vues du nœud i, ce qui présente des avantages et des inconvénients, comme nous allons le voir tout de suite.

Cohérence par nœud

On appellera *cohérence par nœud*, la vérification de (3.3) à partir des compteurs d'un nœud. Il est clair que si la propriété de conservation de flot ainsi formulée n'est pas vérifiée, alors soit le nœud originaire des compteurs a effectivement perdu du trafic, soit il a commis une erreur de maintenance de ses compteurs. Quoi qu'il en soit, on appellera la quantité de non-conservation de flot le *bilan par nœud*, défini de la façon suivante.

Définition 6 (bilan par nœud) *Pour tout nœud i et pour tout instant t, soit $\mathcal{B}_i(t)$ le bilan par nœud de i défini par la relation suivante :*

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \qquad \mathcal{B}_i(t) = \sum_{j \in \mathcal{N}_i(t)} \left(I^i_{ji}(t) - O^i_{ij}(t) \right) \tag{3.4}$$

Un avantage de la formulation de la propriété de conservation de flot comme dans (3.3) est certainement que le bilan par nœud de i peut être calculé en ne se basant que sur les variables maintenues par i. En effet, lorsqu'au lieu de variables en mémoire partagée nous allons considérer que ces valeurs sont diffusées dans les messages de contrôle, il suffira de se baser sur les messages en provenance de i pour effectuer ce genre de calcul. En revanche, l'inconvénient principal est que cette formulation n'assure pas que i a bien maintenu ses compteurs au cours du temps. Autrement dit, i peut toujours mentir en prétendant n'avoir perdu aucun trafic et alors son bilan par nœud sera nul.

De façon évidente, une vérification supplémentaire doit être effectuée, afin de vérifier la validité des valeurs des compteurs d'un nœud.

Cohérence par lien

Les deux vues de $I_{ij}(t)$ et $O_{ij}(t)$ sont deux versions d'une même quantité qui devraient en principe être égales. En disposant à la fois des compteurs de i et de j pour un lien (i, j), on peut vérifier que ceux-ci sont égaux pour tout instant t. On appellera donc *cohérence par lien* la vérification de l'égalité des compteurs des deux extrémités d'un lien.

Définition 7 (bilans par lien) *Soient ${\mathcal{B}^I}^i_{ij}(t)$ et ${\mathcal{B}^O}^i_{ij}(t)$ (respectivement ${\mathcal{B}^I}^j_{ij}(t)$ et ${\mathcal{B}^O}^j_{ij}(t)$) les bilans par lien du point de vue de i (respectivement de j) définis par les relations suivantes :*

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \quad \forall j \in \mathcal{N}_i(t) \qquad \begin{cases} {\mathcal{B}^I}^i_{ij}(t) = I^i_{ij}(t) - I^j_{ij}(t) \\ {\mathcal{B}^O}^i_{ij}(t) = O^i_{ij}(t) - O^j_{ij}(t) \\ {\mathcal{B}^I}^j_{ij}(t) = I^j_{ij}(t) - I^i_{ij}(t) \\ {\mathcal{B}^O}^j_{ij}(t) = O^j_{ij}(t) - O^i_{ij}(t) \end{cases} \tag{3.5}$$

Si pour un nœud i, le bilan par nœud est non nul ou un des bilans d'un de ses liens vers les autres nœuds est non nul à un instant t, alors soit i a perdu du trafic ou mal maintenu ses compteurs, soit un de ses voisins a mal maintenu ses compteurs. À l'inverse, si tous les bilans concernant un nœud i et ses liens sont nuls, cela ne signifie pas qu'aucun paquet n'a été perdu. En effet, il se peut que deux nœuds malveillants en collusion aient diffusé de faux compteurs, destinés à masquer une perte de trafic.

3.1.3 Échange périodique des compteurs

L'hypothèse selon laquelle les compteurs des nœuds sont maintenus en mémoire partagée accessible à tous en permanence n'est pas du tout réaliste. Néanmoins, on peut supposer que les compteurs sont diffusés régulièrement dans les messages de contrôle du protocole de routage existant. On peut aussi supposer qu'il existe une borne supérieure P finie et connue par tous les nœuds de l'intervalle de temps séparant deux diffusions successives d'un message de contrôle par un nœud.

Chacun des messages successifs émis contient un numéro de séquence qui permet de le distinguer des autres. Chaque nœud maintient donc un compteur de séquence qui est incrémenté juste avant l'envoi d'un message de contrôle et dont la valeur courante est utilisée comme numéro de séquence dans ce dernier. À tout moment, le compteur de séquence contient la valeur du numéro de séquence du dernier message de contrôle émis.

Définition 8 (compteur de séquence) *Pour tout instant t, soit $\mathsf{S}_i(t)$ la valeur du compteur de séquence du nœud i. Pour tout numéro de séquence n, soit $\mathsf{T}_i(n)$ l'instant de l'envoi par i du message de contrôle dont le numéro de séquence est n.*

Définition 9 (intervalle maximum d'émission) *Soit P l'intervalle de temps maximum entre deux émissions successives d'un message de contrôle par un nœud.*

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \mathsf{T}_i(n+1) - \mathsf{T}_i(n) \leq \mathsf{P} \tag{3.6}$$

D'après la définition 8 et la façon dont le compteur de séquence est incrémenté, on a la propriété suivante :

$$\forall t \in \mathbb{R}^+, \quad \forall i \in V, \qquad \begin{cases} \mathsf{T}_i\left(\mathsf{S}_i(t)\right) \leq t < \mathsf{T}_i\left(\mathsf{S}_i(t)+1\right) \\ \mathsf{S}_i\left(\mathsf{T}_i\left(\mathsf{S}_i(t)\right)\right) = \mathsf{S}_i(t) \ . \end{cases} \tag{3.7}$$

Définition 10 (ensemble de voisins) *Pour tout nœud i et tout numéro de séquence n, soit $\mathcal{N}_i(n)$ l'ensemble des nœuds ayant été considérés comme voisins de i entre les émissions de messages numéros $n-1$ et n.*

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \mathcal{N}_i(n) = \bigcup_{t \in [\mathsf{T}_i(n-1); \mathsf{T}_i(n)[} \mathcal{N}_i(t) \tag{3.8}$$

On supposera désormais que les valeurs des compteurs sont diffusées dans les messages de contrôle uniquement aux voisins directs de leur émetteur. Ainsi, les voisins disposent des compteurs pour les instants auxquels les messages sont émis. On peut réécrire l'expression du

bilan par nœud en fonction non pas de l'instant t, mais du numéro de séquence n du message :

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \mathcal{B}_i(n) = \sum_{j \in \mathcal{N}_i(n)} \left(I^i_{ji}\left(\mathsf{T}_i(n)\right) - O^i_{ij}\left(\mathsf{T}_i(n)\right)\right) \tag{3.9}$$

Les voisins d'un nœud ne sont donc en mesure de calculer son bilan par nœud qu'à chaque instant de réception de ses compteurs.

Mobilité et valeurs différentielles

Dans un réseau à topologie statique, les voisins d'un nœud sont en nombre fini et ne changent pas au cours du temps. Dans ce cas de figure, la publication de compteurs pour tous les voisins depuis le début du fonctionnement du nœud va de soi et ne pose pas de problème particulier tant que le nombre de ces voisins est faible.

En revanche, dans un réseau mobile, dont la topologie est par définition dynamique, les voisins d'un nœud changent au cours du temps. Potentiellement, le nombre total de voisins différents qu'un nœud particulier peut avoir eu au cours de son fonctionnement n'a de limite que le nombre total des autres nœuds. Il n'est donc pas envisageable de faire publier les compteurs de tous les voisins passés et présents.

Si on ne publie les valeurs des compteurs que pour les voisins présents au moment de l'émission du message de contrôle, alors on doit s'attendre à ce que le bilan par nœud ne soit pas nul dans certains cas de figure, bien que le principe de conservation de flot n'ait pas été violé. La figure 3.2 présente un exemple de changement de voisinage qui peut poser problème. Ici, le nœud l arrive dans le voisinage de i après que j en soit parti : l n'a aucun moyen de savoir que j était voisin de i par le passé si ce dernier ne publie plus de compteurs concernant le lien $\{i, j\}$. Supposons qu'un certain trafic de données ait transité de k vers j en passant par i, autrement dit on a $I^i_{ki}(t) = O^i_{ij}(t) \neq 0$. Il est clair que lorsque l calcule le bilan par nœud de i, $\mathcal{B}_i(t)$, il y a de fortes chances pour que celui-ci soit non nul, puisque l dispose de $I^i_{ki}(t)$ mais pas de $O^i_{ij}(t)$.

Une façon de résoudre ce problème consiste à utiliser des *valeurs différentielles*. Au lieu de diffuser dans les messages les valeurs absolues des compteurs, il s'agit de diffuser la différence de ces valeurs absolues entre l'instant présent et l'instant du dernier envoi de message.

Définition 11 (valeurs différentielles) *Pour tout compteur de la forme $X^i_{ij}(t)$ (respectivement $X^j_{ij}(t)$), où X peut être I ou O, on définit sa valeur différentielle $\dot{X}^i_{ij}(t)$ (respectivement $\dot{X}^j_{ij}(t)$) par la formule suivante :*

$$\forall t \in \mathbb{R}^+, \quad \forall i, j \in V, \qquad \begin{cases} \dot{X}^i_{ij}(t) = X^i_{ij}(t) - X^i_{ij}(\mathsf{T}_i\left(\mathsf{S}_i(t)\right)) \\ \dot{X}^j_{ij}(t) = X^j_{ij}(t) - X^j_{ij}(\mathsf{T}_j(\mathsf{S}_j(t))) \end{cases}. \tag{3.10}$$

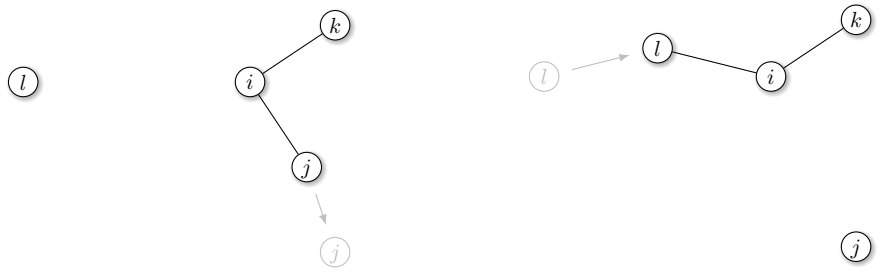


FIG. 3.2: Changement de voisinage : (a) l n'est pas encore voisin de i alors que j s'éloigne ; (b) l devient voisin alors que j ne l'est plus. Le nouveau voisin l n'est pas censé savoir que j a existé dans le voisinage par le passé et quelles étaient les valeurs des compteurs sur le lien $\{i, j\}$ au moment où il l'a quitté.

On notera aussi $\mathsf{T}_i^-(n)$ l'instant juste avant l'émission par i du message au numéro de séquence n, tel que $X_{ij}^i(\mathsf{T}_i^-(n)) = X_{ij}^i(\mathsf{T}_i(n))$ et $X_{ij}^j(\mathsf{T}_j^-(n)) = X_{ij}^j(\mathsf{T}_j(n))$, pour tout compteur X, tous nœuds i et j et tout numéro de séquence n. Cette notation sera utile pour désigner les valeurs différentielles des compteurs et on a donc

$$\forall n \in \mathbb{N}, \quad \forall i, j \in V, \qquad \begin{cases} \dot{X}_{ij}^i\big(\mathsf{T}_i(n)\big) &= 0 \\ \dot{X}_{ij}^j\big(\mathsf{T}_j(n)\big) &= 0 \\ \dot{X}_{ij}^i\big(\mathsf{T}_i^-(n)\big) &= X_{ij}^i\big(\mathsf{T}_i(n)\big) - X_{ij}^i\big(\mathsf{T}_i(n-1)\big) \\ \dot{X}_{ij}^j\big(\mathsf{T}_j^-(n)\big) &= X_{ij}^j\big(\mathsf{T}_j(n)\big) - X_{ij}^j\big(\mathsf{T}_j(n-1)\big) \end{cases} . \tag{3.11}$$

Publication des compteurs

Pour chaque lien avec un voisin j, un nœud i publie dans ses messages de contrôle les valeurs différentielles de ses compteurs concernant ce lien.

Définition 12 (VCP) *Pour tous nœuds i et j et tout numéro de séquence n, soient les valeurs de compteurs publiées (VCP), le quadruplet suivant :*

$$\forall i, j \in V, \quad \forall n \in \mathbb{N}, \qquad V_j^i(n) = \begin{pmatrix} \dot{I}_{ij}^i\left(\mathsf{T}_i^-(n)\right), \dot{O}_{ij}^i\left(\mathsf{T}_i^-(n)\right), \\ \dot{I}_{ji}^i\left(\mathsf{T}_i^-(n)\right), \dot{O}_{ji}^i\left(\mathsf{T}_i^-(n)\right) \end{pmatrix} . \tag{3.12}$$

Entre deux émissions de VCP, un nœud i reçoit d'un voisin j zero ou plus VCP le concernant.

Définition 13 (numéros de séquence accumulés) *Pour tous nœuds i et j et pour tout numéro de séquence n, soit $\mathbb{S}_j^i(n)$ l'ensemble des numéros de séquence des messages de contrôles en*

provenance de j que i a reçus entre les émissions de ses messages de contrôle numéros $n-1$ et n.

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \mathbb{S}_j^i(n) = \{m : \mathsf{T}_i(n-1) \leq \mathsf{T}_j(m) < \mathsf{T}_i(n)\} \tag{3.13}$$

On peut donc définir l'ensemble des VCP inverses reçues par i.

Définition 14 (VCP inverses) *Pour tous nœuds i et j et pour tout numéro de séquence n, soit $R_j^i(n)$ l'ensemble des VCP en provenance de j que i a reçues entre les émissions de ses messages de contrôle numéros $n-1$ et n.*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N} \qquad R_j^i(n) = \bigcup_{m \in \mathbb{S}_j^i(n)} \left\{ V_i^j(m) \right\} \tag{3.14}$$

Ensemble, les définitions précédentes servent à définir précisément ce que l'on entend par publication de lien.

Définition 15 (publication de lien) *Pour tous nœuds i et j et pour tout numéro de séquence n, une publication de lien $L_j^i(n)$ en provenance de i et concernant le lien $\{i,j\}$ est définie par :*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad L_j^i(n) = \left(n, V_j^i(n), R_j^i(n)\right) \quad . \tag{3.15}$$

Enfin, on peut définir un ensemble de publications.

Définition 16 (ensemble de publications) *Pour tout nœud i et pour tout numéro de séquence n, un ensemble de publications $A^i(n)$ est défini par :*

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad A^i(n) = \bigcup_{j \in V} \left\{ L_j^i(n) \right\} \quad . \tag{3.16}$$

3.1.4 Vérification de la cohérence

Lorsqu'un nœud j, voisin de i, reçoit de ce dernier un ensemble de publications $A^i(n)$ de numéro de séquence n, alors il peut effectuer une vérification de cohérence par nœud de i basée sur les valeurs différentielles de ses compteurs. Ce bilan par nœud différentiel peut être exprimé de la façon suivante :

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \dot{\mathcal{B}}_i(n) = \sum_{j \in \mathcal{N}_i(n)} \left(\dot{I}_{ji}^i \left(\mathsf{T}_i(n) \right) - \dot{O}_{ij}^i \left(\mathsf{T}_i(n) \right) \right) \quad . \tag{3.17}$$

Afin de pouvoir exprimer les bilans par lien de manière similaire, il faudrait disposer des compteurs des deux extrémités i et j relevés au même instant. Quand bien même les émissions des publications par les différents nœuds du réseau seraient effectuées au même instant,

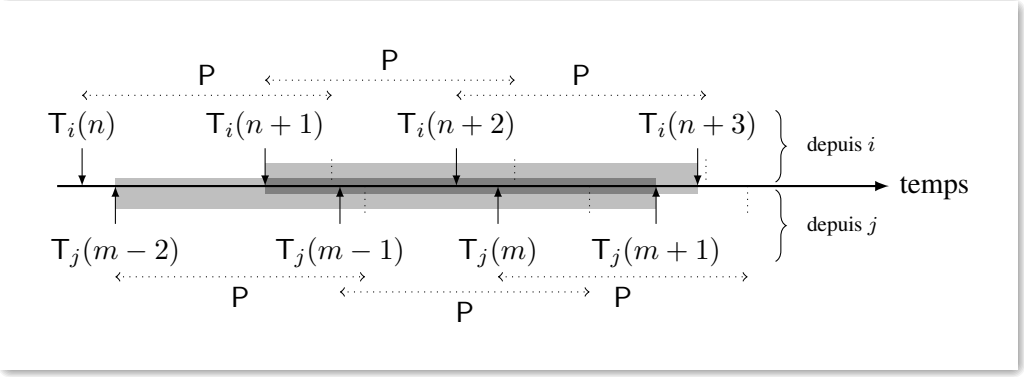


FIG. 3.3: Publications désynchronisées : sont indiqués ici les instants d'arrivée des publications de i et de j, ainsi que les bornes supérieures P sur les intervalles d'arrivées. Les zones en gris représentent les intervalles de temps comptabilisés dans les VCP de i aux numéros $n+2$ à $n+3$, ainsi que ceux comptabilisés dans les VCP de j aux numéros $m-1$ à $m+1$.

pour qu'un nœud k soit capable d'effectuer la vérification de cohérence du lien $\{i, j\}$, il devrait disposer des compteurs de i et j relevés au même moment. Alors soit k dispose à la fois des ensembles de publication de i et j, soit il ne dispose que de celui de i et attend le suivant pour disposer des compteurs de j dans les VCP inverses de i. Dans la pratique cependant, il n'est pas du tout réaliste de synchroniser tous les nœuds d'un réseau à topologie dynamique et donc les bilans par lien ne peuvent être calculés directement. Les VCP des publications de lien des deux extrémités d'un lien concernent donc des compteurs relevés sur différents intervalles de temps (voir FIG. 3.3).

Considérons donc ce que l'on appellera les *bilans désynchronisés par lien* basés sur les ensembles de publication de i.

Définition 17 (bilans désynchronisés) *Pour tout nœud i et un de ses voisins j et pour tout numéro de séquence n, le bilan désynchronisé $\widetilde{\mathcal{B}}^{X\,i}_{kl}(n)$ avec (k,l) étant soit (i,j), soit (j,i), pour X étant soit I, soit O, est défini par :*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\},$$
$$\widetilde{\mathcal{B}}^{X\,i}_{kl}(n) = \dot{X}^i_{kl}\left(\mathsf{T}^-_i(n)\right) - \sum_{m \in \mathbb{S}^i_j(n)} \dot{X}^j_{kl}\left(\mathsf{T}^-_j(m)\right) \quad . \quad (3.18)$$

Afin de pouvoir analyser l'expression (3.18), nous avons besoin de définitions supplémentaires.

Définition 18 (delta de publication) *Pour tous nœuds i et j et pour tout numéro de séquence*

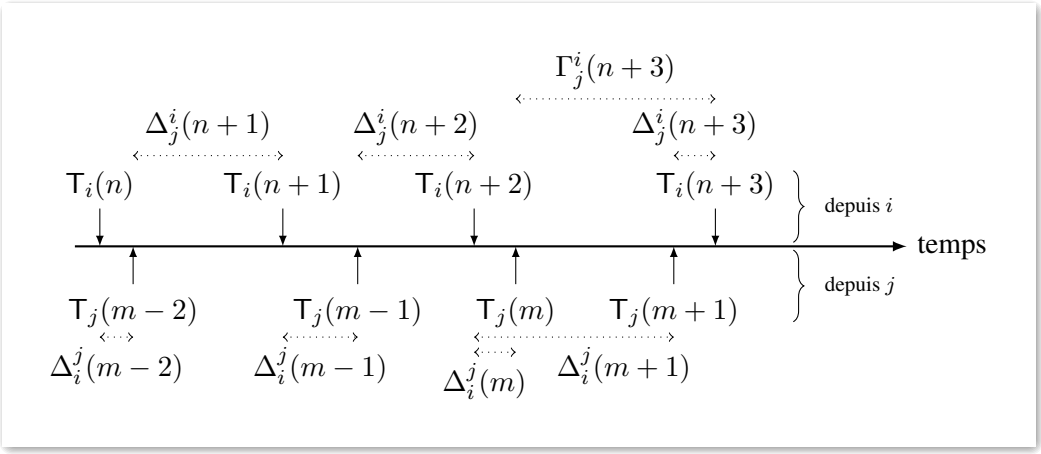


FIG. 3.4: Delta et gamma de publication (Γ_j^i et Γ_i^j, ont été omis là où ils sont égaux à Δ_j^i et Δ_i^j respectivement)

n, soit $\Delta_j^i(n)$ la différence de temps entre l'instant où $A^i(n)$ est émis et la dernière publication de j a été reçue par i (voir FIG. *3.4).*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \Delta_j^i(n) = \begin{cases} \mathsf{T}_i(n) - \max\limits_{m \in \mathbb{S}_j^i(n)} \mathsf{T}_j(m) & \textit{si } \mathbb{S}_j^i(n) \neq \emptyset \ , \\ \mathsf{T}_i(n) - \mathsf{T}_i(n-1) + \Delta_j^i(n-1) & \textit{sinon.} \end{cases} \tag{3.19}$$

On remarque qu'on a toujours $\Delta_j^i(n) \leq \mathsf{P}$, puisque tous les nœuds émettent au moins une fois toutes les P unités de temps.

Définition 19 (flot final) *Pour tous nœuds i et j et tout numéro de séquence n, soit $\mathcal{FX}_{ij}^i(n)$ (respectivement $\mathcal{FX}_{ji}^i(n)$) le flot final sur le lien (i,j) (respectivement (j,i)) défini par*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \mathcal{FX}_{kl}^i(n) = X_{kl}\left(\mathsf{T}_i(n)\right) - X_{kl}\left(\mathsf{T}_i(n) - \Delta_j^i(n)\right) \ . \tag{3.20}$$

On remarque qu'on a

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \mathcal{FX}_{kl}^i(n) \geq 0 \ , \tag{3.21}$$

puisque les compteurs sont croissants.

Dans le cas où les compteurs ne sont pas falsifiés et qu'aucune perte n'a eu lieu sur un lien,

on peut exprimer $\widetilde{\mathcal{B}^{X}}^{i}_{kl}(n)$ en fonction de flots finaux de la manière suivante :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \widetilde{\mathcal{B}^{X}}^{i}_{kl}(n) = \mathcal{F}^{X}{}^{i}_{kl}(n) - \mathcal{F}^{X}{}^{i}_{kl}(n-1) \ . \quad (3.22)$$

Bien que les valeurs de $\Delta^i_j(n)$ ne soient pas connues pour les nœuds autres que i et j eux-mêmes, il est clair que cette expression nous donne directement une borne supérieure à la valeur d'un bilan désynchronisé par lien. On peut supposer que la capacité maximale d'un lien, c'est-à-dire la quantité maximale de trafic qui peut le traverser par unité de temps, est finie et connue.

Définition 20 (capacité d'un lien) *Soit C_{ij} la capacité du lien (i,j), soit la quantité maximale de trafic pouvant traverser le lien (i,j) par unité de temps.*

On a donc d'après les définitions 9 et 18 une borne sur le flot final :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \mathcal{F}^{X}{}^{i}_{kl}(n) \leq \mathsf{C}_{kl} \cdot \mathsf{P} \ . \quad (3.23)$$

Bilans désynchronisés simples

D'après (3.22) et (3.23), on obtient directement une borne sur les bilans désynchronisés par lien :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \max\left(\left|\widetilde{\mathcal{B}^{I}}^{i}_{ij}(n)\right|, \left|\widetilde{\mathcal{B}^{O}}^{i}_{ij}(n)\right|, \left|\widetilde{\mathcal{B}^{I}}^{j}_{ij}(n)\right|, \left|\widetilde{\mathcal{B}^{O}}^{j}_{ij}(n)\right|\right) \leq \mathsf{C}_{ij} \cdot \mathsf{P} \ . \quad (3.24)$$

On a donc l'implication suivante : si les compteurs des extrémités d'un lien sont en accord à tout moment et leurs valeurs sont publiées telles quelles, alors les bornes de (3.24) sont vérifiées. Par la contraposée, si ces bornes ne sont pas vérifiées pour une publication de lien, alors soit les compteurs n'ont pas été en accord, soit leurs valeurs ont été faussées. En revanche, on ne dispose pas ici de la réciproque, selon laquelle la vérification des bornes impliquerait que les compteurs sont en accord et leurs vraies valeurs publiées.

La vérification de ces bornes permet de détecter une incohérence par lien apparue depuis la publication de lien précédente, à la condition qu'elle soit suffisamment importante. Cela requiert donc un trafic important avec un taux de perte important également, qui entraîne une incohérence des compteurs importante et détectable de cette manière.

Bilans désynchronisés cumulés

Dans le cas où le taux de perte est faible ou bien le trafic peu important mais continu, alors les incohérences peuvent croître trop lentement pour être détectables par la méthode du paragraphe

précédent. Au lieu de considérer les valeurs immédiates des bilans désynchronisés, un nœud peut sommer ces valeurs sur les publications successives. Il peut ainsi allonger la période de temps sur laquelle il effectue la détection d'incohérence en reconstituant des bilans sur des périodes de temps plus longues. Considérons alors la forme d'un bilan cumulé sur un ensemble de N numéros de séquence $[n;n+N]$.

Lemme 1 *Pour tous nœuds i et j, pour tout numéro de séquence n et pour tout nombre de N publications successives, les bilans désynchronisés cumulés ont l'expression suivante :*

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{X\,i}_{kl}(p) = \mathcal{F}^{X\,i}_{kl}(n+N) - \mathcal{F}^{X\,i}_{kl}(n-1) \ . \quad (3.25)$$

PREUVE Par récurrence. Le cas $N=0$ est celui de (3.22). En prenant l'expression pour N, on y ajoute les deux membres de celle de $\widetilde{\mathcal{B}}^{X\,i}_{kl}(n+N+1)$:

$$\begin{aligned} \forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \sum_{p=n}^{n+N+1} \widetilde{\mathcal{B}}^{X\,i}_{kl}(p) = \ & \mathcal{F}^{X\,i}_{kl}(n+N) - \mathcal{F}^{X\,i}_{kl}(n-1) \\ & + \mathcal{F}^{X\,i}_{kl}(n+N+1) - \mathcal{F}^{X\,i}_{kl}(n+N) \ , \end{aligned} \quad (3.26)$$

qui se simplifie en :

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \sum_{p=n}^{n+N+1} \widetilde{\mathcal{B}}^{X\,i}_{kl}(p) = \mathcal{F}^{X\,i}_{kl}(n+N+1) - \mathcal{F}^{X\,i}_{kl}(n-1) \ . \quad (3.27)$$

qui prouve enfin l'hypothèse de récurrence. ∎

Un corollaire du lemme 1 est que les bornes des bilans désynchronisés cumulés sont les mêmes que celles des bilans désynchronisés simples :

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \qquad \max\left(\left| \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{I\,i}_{ij}(p) \right|, \left| \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{O\,i}_{ij}(p) \right|, \left| \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{I\,j}_{ij}(p) \right|, \left| \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{O\,j}_{ij}(p) \right| \right) \leq \mathsf{C}_{ij} \cdot \mathsf{P} \ . \quad (3.28)$$

Il en découle donc qu'une manière de vérifier la cohérence par lien est pour un nœud de sommer les bilans par lien et d'observer si ces valeurs cumulées respectent leurs bornes théo-

riques. Cette méthode permet de détecter des incohérences qui croissent moins vite, pour peu que le trafic soit continu sur une période de temps plus longue.

Bilans désynchronisés cumulés et faible trafic

L'observation de l'évolution des bilans désynchronisés au cours du temps en présence ou non de perte de trafic permet de remarquer plusieurs choses intéressantes.

Prenons un intervalle de numéros de séquence $[n;n+N]$ de publications du lien (i,j) de la part de i. En absence totale de trafic sur l'intervalle de temps $[\mathsf{T}_i(n-1)-\Delta^i_j(n-1);\mathsf{T}_i(n+N)[$, les bilans sont censés être nuls, puisque les VCP sont nulles.

En absence de trafic sur les intervalles $[\mathsf{T}_i(n-1)-\Delta^i_j(n-1);\mathsf{T}_i(n)[$ et $[\mathsf{T}_i(n+M-1)-\Delta^i_j(n+M-1);\mathsf{T}_i(n+M)[$, les bilans cumulés sur $[n;n+M]$ sont censés être nuls également. En effet, de tels bilans sont des différences entre compteurs qui ont compté exactement le même trafic, donc si les compteurs sont cohérents, la différence est forcément nulle (voir FIG. 3.5).

En revanche, si les bilans commencent à être accumulés à un moment où des paquets de données transitent par le lien, il se peut fortement qu'on ne se retrouve pas avec des bilans nuls une fois le trafic terminé (voir FIG. 3.6). En fait les valeurs des bilans accumulés ne doivent pas être considérées de façon absolue, puisque leur décalage par rapport à zéro n'est pas connu.

En observant les valeurs des bilans cumulés en absence de pertes (FIG. 3.7), on remarque que les bilans tendent vers une valeur minimale lorsque le trafic sur le lien diminue. On s'attend donc à trouver une relation entre l'écart à ce minimum et la quantité de trafic traversant le lien à un instant. Ainsi lorsque le trafic sur le lien cesse complètement, on doit avoir des bilans cumulés à leur minimum.

En présence d'incohérence des compteurs (FIG. 3.8), les bilans cumulés ont tendance à se décaler vers des valeurs plus grandes ou plus petites, selon le signe de la différence des compteurs. Par conséquent, on observe que les bilans ne tendent plus vers leur valeur minimale avec la diminution du trafic.

En supposant qu'on connaisse $\mathcal{F}^{x\,i}_{kl}(n)$, autrement dit la quantité de trafic qui passe sur le lien (k,l) dans l'intervalle $[\mathsf{T}_i(n)-\Delta^i_j(n);\mathsf{T}_i(n)[$, on trouve une expression d'une borne supérieure des bilans cumulés du point de vue de i.

Théorème 1 (borne supérieure des bilans) *Pour tous nœuds i et j, pour tout numéro de séquence n et pour tout nombre de N publications successives, les bilans désynchronisés cumulés*

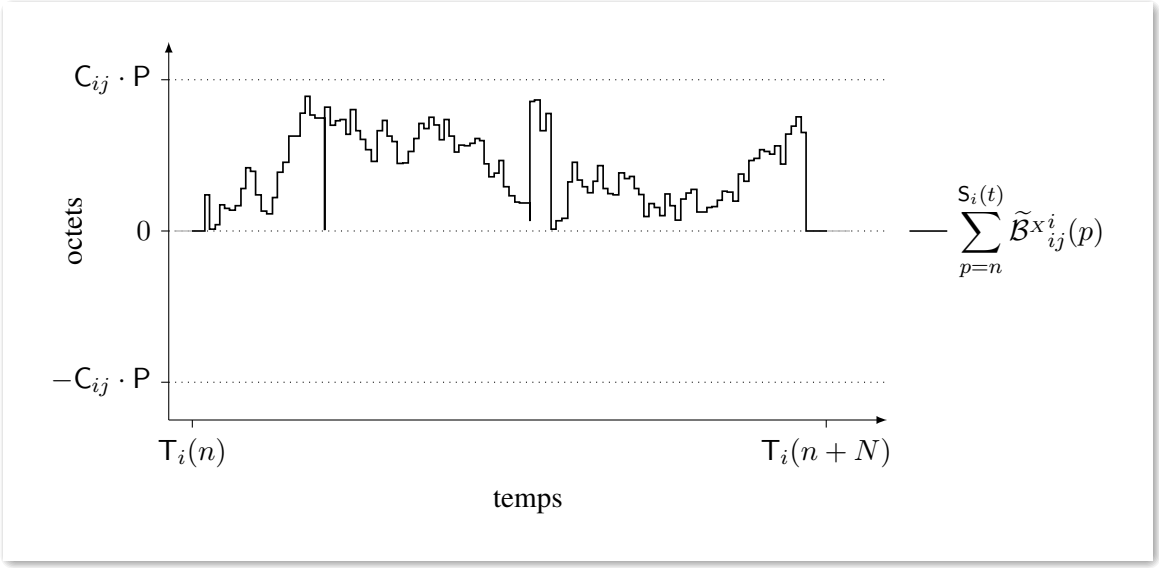


FIG. 3.5: Bilan nul en présence de trafic : si le trafic a eu lieu uniquement dans l'intervalle de temps couvert à la fois par les compteurs des deux extrémités, le bilan est nul.

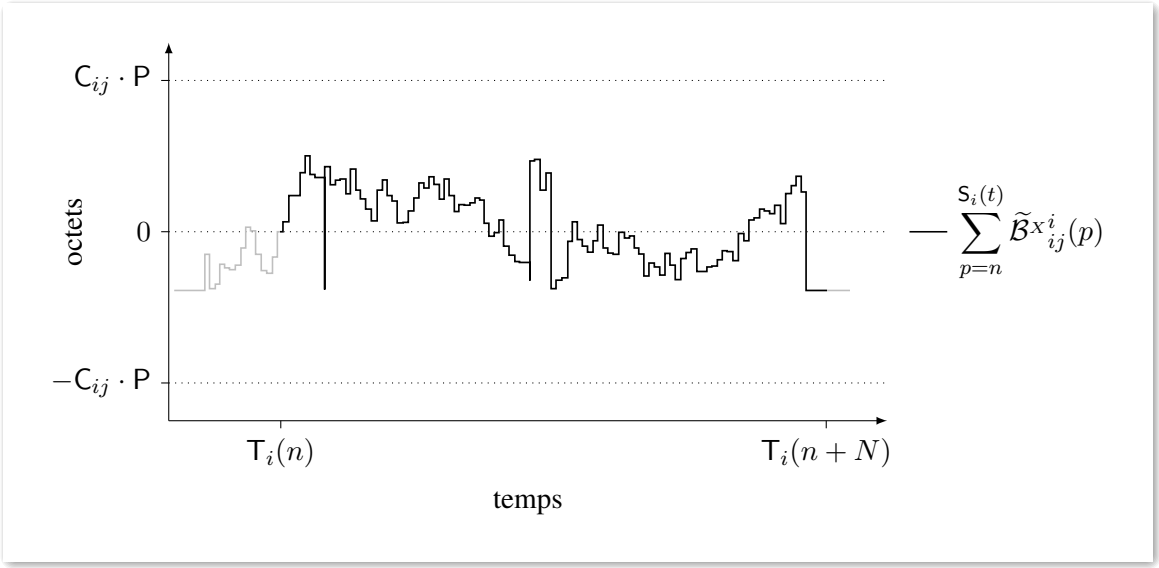
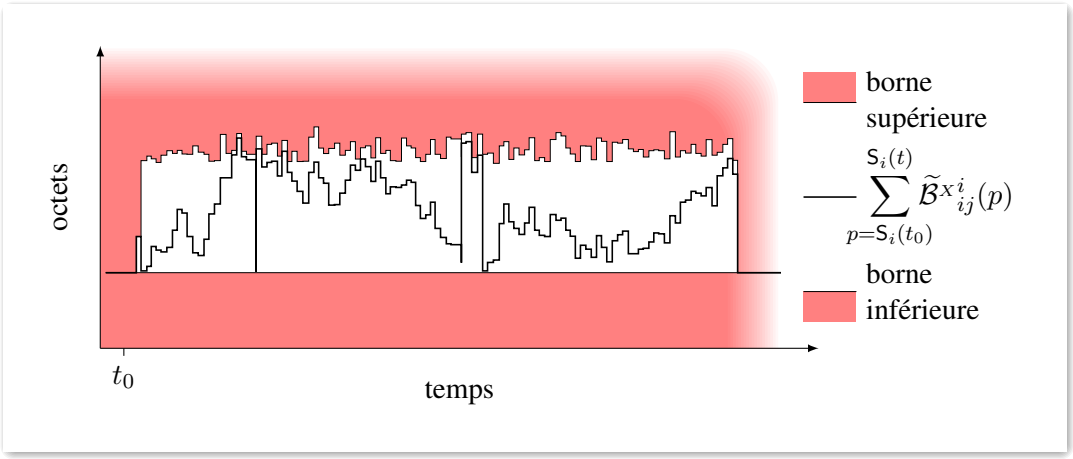
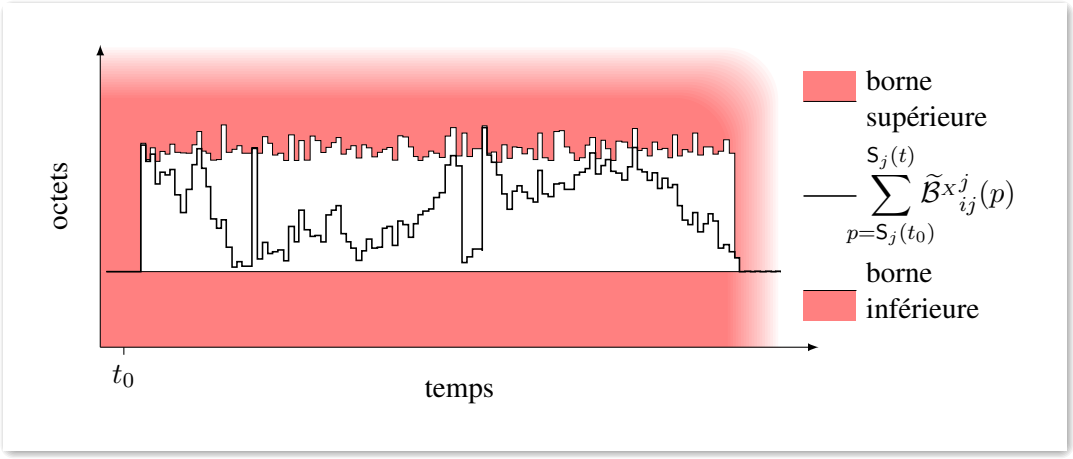


FIG. 3.6: Bilan non nul en présence de trafic : si on commence à sommer un bilan en présence de trafic, alors le bilan peut ne pas être nul en absence ultérieure de trafic.

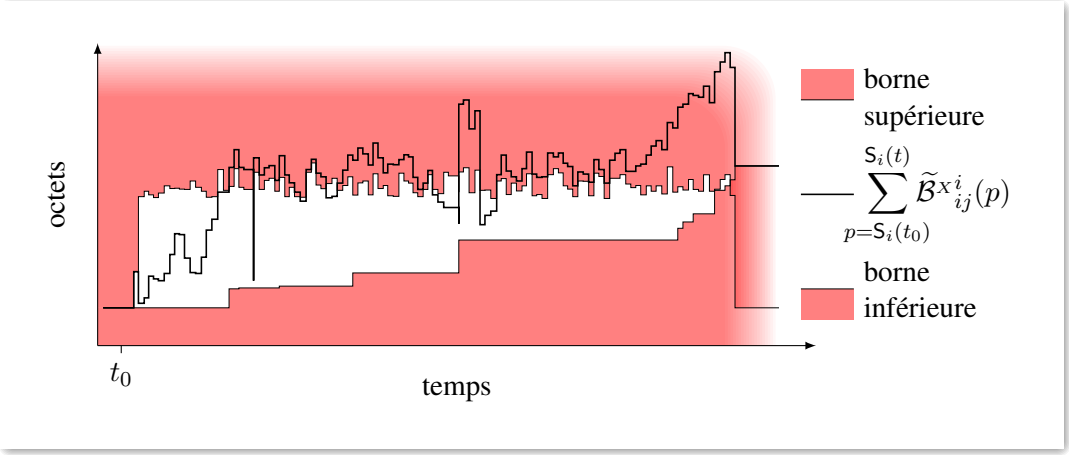


(a)

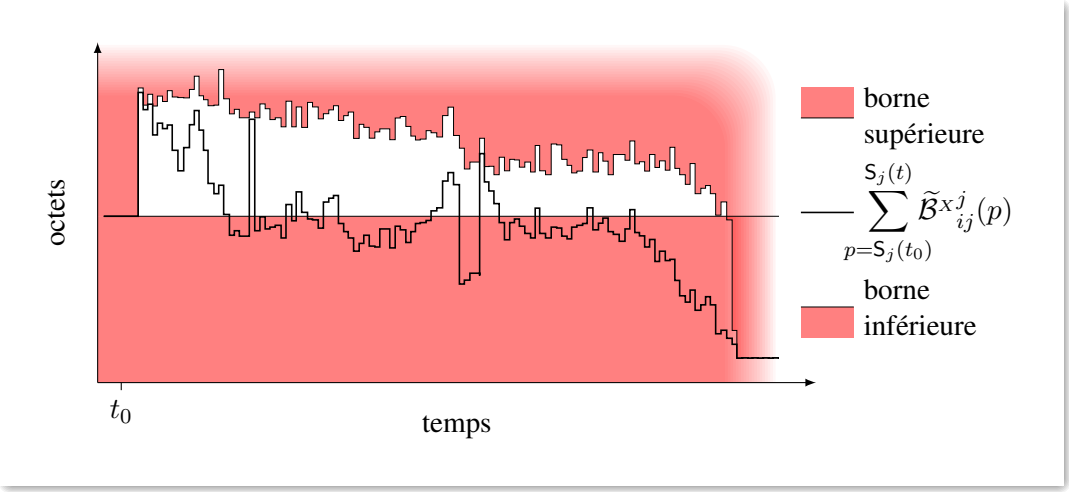


(b)

FIG. 3.7: Bilans en absence de pertes : (a) du point de vue de i et (b) du point de vue de j.



(a)



(b)

FIG. 3.8: Bilans en présence de trafic avec perte de paquets : (a) du point de vue de i et (b) du point de vue de j. On constate dans le premier cas le bilan a tendance à augmenter et dans le second à diminuer.

ont la borne supérieure suivante :

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p) \leq \min_{q\in[n;n+N]} \sum_{p=n}^{q} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p) + \mathcal{F}^{x\,i}(n+N) \ . \quad (3.29)$$

PREUVE En reprenant l'expression du flot final et des bilans cumulés de (3.25), on a

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \min_{q\in[n;n+N]} \sum_{p=n}^{q} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p) = \min_{q\in[n;n+N]} \left(\mathcal{F}^{x\,i}_{kl}(q)\right) - \mathcal{F}^{x\,i}_{kl}(n-1) \ . \quad (3.30)$$

En réintroduisant cette expression dans (3.29) et en utilisant l'expression des bilans cumulés de (3.25) pour le membre de gauche, on obtient

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \mathcal{F}^{x\,i}_{kl}(n+N) - \mathcal{F}^{x\,i}_{kl}(n-1) \leq \min_{q\in[n;n+N]} \left(\mathcal{F}^{x\,i}_{kl}(q)\right) - \mathcal{F}^{x\,i}_{kl}(n-1) + \mathcal{F}^{x\,i}_{kl}(n+N) \ , \quad (3.31)$$

qui se réduit simplement en

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad 0 \leq \min_{q\in[n;n+N]} \left(\mathcal{F}^{x\,i}_{kl}(q)\right) \ , \quad (3.32)$$

qui est toujours vraie puisque $\mathcal{F}^{x\,i}_{kl}(n) \geq 0$. ∎

De façon similaire, on peut exprimer une borne inférieure des bilans cumulés du point de vue de i.

Théorème 2 (borne inférieure des bilans) *Pour tous nœuds i et j, pour tout numéro de séquence n et pour tout nombre de N publications successives, les bilans désynchronisés cumulés ont la borne inférieure suivante :*

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p) \geq \max_{q\in[n;n+N]} \left(\sum_{p=n}^{q} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p) - \mathcal{F}^{x\,i}_{kl}(q) \right) \ . \quad (3.33)$$

PREUVE En reprenant l'expression des bilans cumulés de (3.25), on a

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \max_{q\in[n;n+N]} \left(\sum_{p=n}^{q} \widetilde{\mathcal{B}}^{X\,i}_{kl}(p) - \mathcal{F}^{X\,i}_{kl}(q) \right) = -\mathcal{F}^{X\,i}_{kl}(n-1) \quad (3.34)$$

En réintroduisant cette expression dans (3.29) et en utilisant l'expression des bilans cumulés de (3.25) pour le membre de gauche, on obtient

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ \mathcal{F}^{X\,i}_{kl}(n+N) - \mathcal{F}^{X\,i}_{kl}(n-1) \geq -\mathcal{F}^{X\,i}_{kl}(n-1) \ , \quad (3.35)$$

qui se réduit simplement en

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \mathcal{F}^{X\,i}_{kl}(n+N) \geq 0 \ , \quad (3.36)$$

qui est toujours vraie par définition. ∎

D'après les théorèmes 1 et 2, plus le trafic est faible, plus les bornes sont serrées et à l'extrême, quand le trafic est nul, elles sont toutes les deux égales à zéro. Mais le calcul de ces bornes repose sur la connaissance des valeurs exactes des flots finaux et ceux-ci ne sont pas connus directement par les nœuds qui effectuent les vérifications. Cependant, nous allons voir que des bornes supérieures aux flots finaux peuvent être extraites des valeurs de compteurs publiées par les nœuds.

Définition 21 (gamma de publication) *Pour tous nœuds i et j et pour tout numéro de séquence n, soit $\Gamma^i_j(n)$ la différence de temps entre l'instant où $A^i(n)$ est émis et la première publication de j a été reçue par i depuis l'émission de $A^i(n-1)$ (voir* FIG. *3.4).*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N} : \mathbb{S}^i_j(n) \neq \emptyset, \qquad \Gamma^i_j(n) = \mathsf{T}_i(n) - \min_{m\in\mathbb{S}^i_j(n)} \mathsf{T}_j(m) \quad (3.37)$$

Considérons une certaine publication $L^i_j(n)$ avec un certain numéro de séquence n. On sait que la valeur $\dot{X}^i_{kl}(\mathsf{T}^-_i(n))$ est la déclaration selon i de la quantité de trafic ayant traversé le lien (k,l) pour le compteur X entre les instants $\mathsf{T}_i(n-1)$ et $\mathsf{T}_i(n)$. Si la publication contient un ensemble de publications inverses $R^i_j(n)$ non vide, alors la publication $V^j_i(m) \in R^i_j(n)$ avec le plus petit m contient $\dot{X}^j_{kl}(\mathsf{T}^-_j(m))$ qui est la déclaration selon j de la quantité de trafic ayant traversé le même lien pour le compteur X entre les instants $\mathsf{T}_j(m-1)$ et $\mathsf{T}_j(m)$. Or il se trouve

qu'en combinant ces intervalles, on peut obtenir un intervalle contenant celui du flot final.

Pour poursuivre ce raisonnement, nous avons besoin de définitions supplémentaires.

Définition 22 (dernier et prochain) *Pour tous nœuds i et j et pour tout numéro de séquence n, soient $\mathsf{L}^i_j(n)$ (respectivement $\mathsf{N}^i_j(n)$) le numéro de séquence non inférieur à n (respectivement supérieure à n) de la dernière (respectivement de la prochaine) publication du lien $\{i,j\}$ par i qui contient au moins un n-uplet VCP inversé.*

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \begin{cases} \mathsf{L}^i_j(n) = \max\left\{m : m \leq n \wedge \mathbb{S}^i_j(m) \neq \emptyset\right\} \\ \mathsf{N}^i_j(n) = \min\left\{m : m > n \wedge \mathbb{S}^i_j(m) \neq \emptyset\right\} \end{cases} \tag{3.38}$$

Ces définitions permettent de généraliser le raisonnement que l'ensemble des VCP inverses soit vide ou non. En considérant le lien (k,l), pour une publication de i de numéro de séquence n, on détermine aisément des intervalles contenant l'intervalle du flot final :

$$\begin{aligned} &\forall i,j \in V, \quad \forall n \in \mathbb{N}, \\ &\qquad \begin{cases} \left[\mathsf{T}_i(n) - \Delta^i_j(n); \mathsf{T}_i(n)\right[\subseteq \left[\mathsf{T}_i\left(\mathsf{L}^i_j(n)-1\right); \mathsf{T}_i(n)\right[\\ \left[\mathsf{T}_i(n) - \Delta^i_j(n); \mathsf{T}_i(n)\right[\subseteq \left[\mathsf{T}_i(n) - \Delta^i_j(n); \mathsf{T}_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right[\end{cases} . \end{aligned} \tag{3.39}$$

En accumulant les $L^i_j(n)$ pour des n successifs, on peut extraire pour chacun d'entre eux le minimum de ce que les extrémités i et j déclarent avoir vu passer pendant des intervalles contenant $[\mathsf{T}_i(n) - \Delta^i_j(n); \mathsf{T}_i(n)[$.

$$\begin{aligned} &\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \\ &\qquad \mathcal{M}^{x\,i}_{kl}(n) = \min\left(\sum_{p=\mathsf{L}^i_j(n)}^{n} \left(\dot{X}^i_{kl}\left(\mathsf{T}^-_i(p)\right)\right), \dot{X}^j_{kl}\left(\mathsf{T}^-_i\left(\mathsf{N}^i_j(n)\right) - \Gamma^i_j\left(\mathsf{N}^i_j(n)\right)\right)\right) \end{aligned} \tag{3.40}$$

Si on suppose que les valeurs de compteurs déclarées par i et j sont véridiques, alors il est clair que la valeur du flot final $\mathcal{F}^{x\,i}_{kl}(n)$ est nécessairement bornée par ce minimum :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\}, \qquad \mathcal{F}^{x\,i}_{kl}(n) \leq \mathcal{M}^{x\,i}_{kl}(n) \ . \tag{3.41}$$

On peut ainsi incorporer cette borne à la place du flot final dans les équations des théorèmes

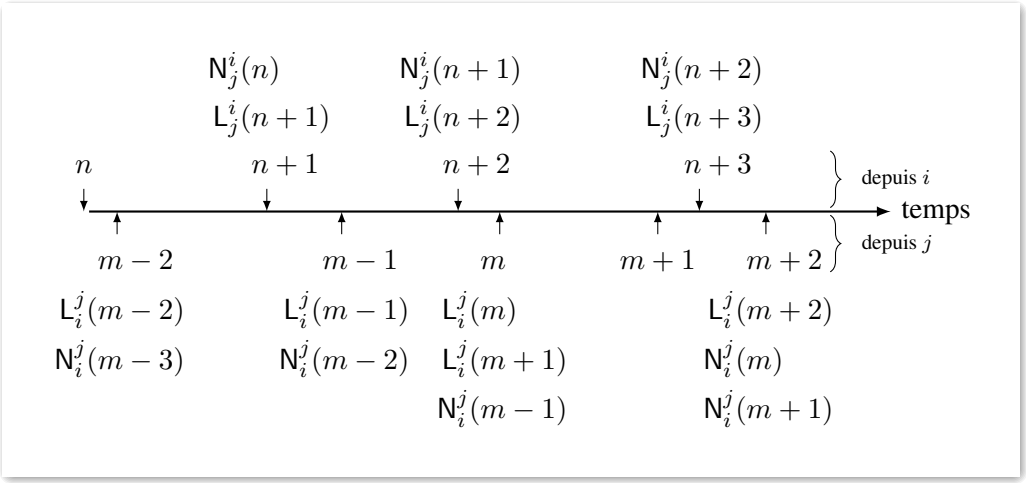


FIG. 3.9: Dernier et prochain : on a ici $\mathsf{L}_i^j(m+1) = m$ et $\mathsf{N}_i^j(m) = m+2$, puisque $\mathbb{S}_i^j(m+1) = \emptyset$ et $\mathbb{S}_i^j(m+2) \neq \emptyset$.

1 et 2

$$\forall i,j \in V, \quad \forall n,N \in \mathbb{N}, \quad \forall (k,l) \in \{(i,j),(j,i)\},$$
$$\max_{p\in[n;n+N]} \left(\sum_{q=n}^{p} \widetilde{\mathcal{B}}^{x\,i}_{kl}(q) - \mathcal{M}^{x\,i}_{kl}(p) \right)$$
$$\leq \sum_{p=n}^{n+N} \widetilde{\mathcal{B}}^{x\,i}_{kl}(p)$$
$$\leq \min_{p\in[n;n+N]} \sum_{q=n}^{p} \widetilde{\mathcal{B}}^{x\,i}_{kl}(q) + \mathcal{M}^{x\,i}_{kl}(n+N) \ . \quad (3.42)$$

Si on suppose que soit les valeurs de i ou soit celles de j ne sont pas véridiques, alors de deux choses l'une : soit les fausses valeurs sont trop grandes et alors (3.41) est toujours vérifiée, soit elles sont trop petites et alors (3.41) n'est plus vraie. Dans le second cas, on observe que (3.42) n'est plus forcément vraie, ce qui est souhaitable.

On remarque que (3.42) est tout à fait vérifiable par les nœuds voisins de i, puisqu'elle n'est formulée qu'en termes d'informations contenues dans les $L_j^i(n)$ successifs. Ces bornes sont illustrées dans les figures 3.7 et 3.8. En cas de présence de pertes, ces bornes permettent de détecter efficacement qu'un bilan est en train d'augmenter ou diminuer, selon la direction du trafic.

On peut remarquer par ailleurs, que tous les termes de (3.42) sont maintenus sans besoin de garder les $L_{kl}^i(n)$ passés et donc ne requièrent qu'une quantité limitée de mémoire par lien

observé.

Résumé de la méthode de vérification

La méthode de vérification du principe de conservation de flot dans le réseau consiste donc en les points suivants.

1) Tout nœud i transmet périodiquement son ensemble de publications $A^i(\mathsf{S}_i(t))$ à tous ses voisins $j \in \mathcal{N}_i(t)$.
2) À chaque réception d'un ensemble de publication $A^i(n)$ en provenance d'un voisin i, un nœud doit
   2a) calculer le bilan par nœud $\dot{\mathcal{B}}_i(n)$ de i selon (3.17)

   et pour chaque lien $(k,l) = (i,j)$ ou $(k,l) = (j,i)$ entre i et un de ses voisins j,
   2b) extraire les bornes des flots finaux ${\mathcal{M}^{x}}^{i}_{kl}(n')$ selon (3.40) pour tout n' correspondant à un bilan cumulé en attente de vérification ;
   2c) calculer les bilans par lien désynchronisés ${\widetilde{\mathcal{B}}^{x}}^{i}_{kl}(n)$ selon (3.18) ;
   2d) mettre à jour les bilans cumulés $\sum_{p=n_0}^{n} {\widetilde{\mathcal{B}}^{x}}^{i}_{kl}(p)$ à partir des bilans par lien désynchronisés ;
   2e) mettre à jour les bornes des bilans cumulés $\sum_{p=n_0}^{n'} {\widetilde{\mathcal{B}}^{x}}^{i}_{kl}(p)$ dont la borne du flot final ${\mathcal{M}^{x}}^{i}_{kl}(n')$ est connue et les vérifier selon (3.42) ; mettre en attente les bilans cumulés dont la borne du flot final n'est pas encore connue.

Le principe de conservation de flot n'est pas vérifié dès que l'on détecte soit une valeur de $\dot{\mathcal{B}}_i(n)$ non nulle au point 2a, soit que les bornes sont dépassées au point 2e. La valeur $\dot{\mathcal{B}}_i(n)$ peut être considérée comme la quantité de trafic que le nœud i *admet* avoir perdue, tandis que la valeur absolue de la quantité de dépassement par $\sum_{p=n_0}^{n} {\widetilde{\mathcal{B}}^{x}}^{i}_{kl}(p)$ de ses bornes théoriques n'est qu'une borne inférieure à la quantité de trafic supposée perdue sur le lien.

À chaque fois qu'une incohérence de compteurs est constatée sur un lien, les bilans cumulés et les bornes sur les bilans et le flot final sont réinitialisés. L'accumulation des bilans par lien et le maintien des bornes reprend à la réception de l'ensemble de publications suivant comme si c'était le premier.

3.2 Notation de la fiabilité des nœuds

L'utilisation directe du résultat de la vérification du principe de conservation de flot pour exclure les nœuds fautifs n'est pas souhaitable pour plusieurs raisons.

Premièrement, on doit pouvoir tolérer une certaine perte de trafic dans un réseau à topologie dynamique, puisque les protocoles existants ne peuvent pas garantir l'acheminement des paquets de données à tous les coups. La mobilité et les ressources énergétiques restreintes des nœuds, ainsi que des conditions variables de transmission radio sont autant de facteurs jouant sur la perte inopinée de paquets. On ne doit donc pas restreindre le champ d'application des protocoles aux cas improbables en pratique de conditions idéales assurant l'absence de pertes fortuites.

Deuxièmement, lorsque la vérification échoue au niveau d'un lien (un bilan cumulé par lien qui dépasse ses bornes), on ne peut pas décider quelle extrémité est fautive. Dans l'absolu, ces dernières doivent être traitées de la même façon. Si les deux extrémités d'un lien à problème doivent être exclues du réseau, alors il devient facile pour un nœud malveillant de mettre en œuvre l'attaque *kamikaze* par laquelle il vient annoncer des compteurs qui contredisent ceux de sa victime afin de la faire exclure en même temps que lui.

Enfin, nous avons vu que seuls les voisins d'un nœud sont capables d'effectuer la vérification du principe de conservation de flot en ce nœud. Dès lors, une action localisée en ces nœuds voisins, comme l'exclusion directe du/des noeud(s) fautif(s) de la topologie, peut compromettre le fonctionnement du routage. En effet, la vision de la topologie du réseau doit être uniforme sur les nœuds, c'est-à-dire que tous les nœuds qui participent au routage doivent avoir une vision de la topologie cohérente afin d'éviter l'apparition de cycles.

La réponse à ces questions que nous apportons est de distinguer deux degrés de suspicion : une version locale, maintenue en fonction des évènements directement observables et une version globale, calculée de façon uniforme à partir d'informations obtenues depuis les autres nœuds et qui sert en tant que métrique de routage.

Nous allons voir maintenant comment ces deux degrés de suspicion sont calculés et comment le routage est effectué.

3.2.1 Degré de suspicion local

Le degré de suspicion local reflète l'idée que se fait un nœud de la fiabilité d'un autre nœud qu'il a soumis par le passé aux vérifications présentées au paragraphe 3.1. La façon de maintenir ce degré de suspicion doit répondre aux impératifs suivants : un nœud non connu a un degré initialement nul ; un nœud qui échoue à une vérification devrait voir son degré augmenter d'autant plus qu'il a échoué par le passé ; un nœud qui n'a pas échoué depuis longtemps devrait voir son degré tendre vers zéro.

Au niveau de chaque nœud i, à chaque fois qu'un de ses voisins j a un bilan par nœud non nul ou bien se trouve à l'extrémité d'un lien dont un bilan cumulé dépasse ses bornes, on pose L la borne inférieure à la quantité de trafic supposée perdue et on augmente le degré de suspicion

de i envers j, D_j^i, de la façon suivante :

$$D_j^i \leftarrow \max(D_j^i, 1) \cdot (1 + L \cdot r_I) \ , \tag{3.43}$$

où r_I est un paramètre du système qui indique à quelle vitesse le degré de suspicion doit augmenter. On a donc bien une augmentation exponentielle avec les échecs successifs aux vérifications du principe de conservation de flot. Cette mesure a pour but de prendre en compte plus sérieusement la récidive.

Le degré de suspicion doit être maintenu en mémoire tant qu'il n'est pas nul, pour qu'il ne suffise pas qu'un nœud malveillant sorte du voisinage pour racheter la confiance des autres. Comme on veut que des échecs à la vérification soient perçus comme bénins s'ils sont rares, les degrés de suspicion doivent être diminués à mesure que s'écoule le temps. Nous proposons donc une mise à jour des degrés de la façon suivante :

$$D_j^i \leftarrow \max(D_j^i - \Delta t \cdot r_D, 0) \ , \tag{3.44}$$

où r_D est un paramètre du système qui indique à quelle vitesse le degré de suspicion doit diminuer et Δt le temps écoulé depuis la dernière mise à jour. On a ainsi une diminution linéaire, ce qui a pour effet d'annuler les degrés des nœuds pour lesquels aucun échec n'a été constaté depuis un certain temps. Par conséquent, le nombre de degrés à maintenir en mémoire n'augmente pas indéfiniment, puisque les degrés nuls peuvent être détruits.

Le degré de suspicion local ainsi maintenu peut servir dans des calculs dont le résultat n'a pas à être uniforme sur tout le réseau et peut donc se baser sur un jugement *subjectif* des nœuds. Dans tous les cas, ce degré local est diffusé aux autres nœuds du réseau, par le biais des messages de contrôle du protocole de routage, afin de maintenir le degré de suspicion global.

3.2.2 Degré de suspicion global

Pour éviter de se retrouver avec des cycles dans le routage, le calcul des tables de routage doit se baser sur une vision uniforme de la topologie du réseau. Autrement dit, tous les nœuds doivent avoir la même vue du réseau et effectuer un calcul de chemins qui assure l'absence de cycles, comme l'algorithme de Dijkstra avec des poids positifs sur les arcs du graphe.

C'est pour cette raison que les degrés locaux seuls ne suffisent pas. Différents nœuds peuvent en effet avoir eu des observations différentes concernant un même nœud et donc des degrés de suspicion locaux différents à son égard. Il est donc nécessaire de mettre en place un mécanisme de maintien de degrés uniformes sur les nœuds du réseau. Nous proposons donc que les degrés de suspicion locaux soient diffusés à tout le réseau et que tous les nœuds utilisent la même

méthode de combinaison de ces degrés locaux pour produire leurs degrés globaux.

En plus de l'adresse de chaque voisin pour lequel un nœud publie une entrée dans ses messages topologiques, le degré de suspicion local est inclus également. Ainsi, en combinant les messages, un nœud reconstitue la topologie du réseau avec pour chaque sommet autant de degrés locaux que le sommet a d'arcs incidents (voir FIG. 3.10). C'est à partir de ces degrés locaux incidents que le degré global du sommet est calculé. Afin de limiter l'impact de nœuds malveillants qui diffuseraient des degrés locaux exagérément grands en guise de diffamation, nous proposons de calculer le degré global comme la médiane des degrés locaux. L'avantage de l'utilisation de la médiane est que l'impact des degrés locaux déviant trop des autres est limité.

Le calcul d'une médiane simple a toutefois le défaut de ne pas permettre de distinguer un degré local faible concernant un nœud i dû aux scénarios suivants :

- le nœud n'a pas constaté d'échec de i aux vérifications du principe de conservation de flot ;
- le nœud n'a pas eu vraiment d'occasion de soumettre i à des vérifications.

Pour combler ce défaut et rendre la méthode d'autant plus robuste, nous proposons que chaque publication de degré local soit accompagnée d'un indice de confiance, c'est-à-dire d'un poids qui indique à quel point le nœud émetteur pense que son degré doit être pris en compte. Cet indice de confiance, initialement nul, doit être maintenu en parallèle du degré de suspicion local. Il doit augmenter avec la quantité supposée de trafic transitant par le nœud auquel il est associé et diminuer avec le temps qui passe, de façon à ce qu'un grand indice de confiance signifie que le degré local est basé sur des vérifications récentes.

Suite à la vérification par i d'un voisin j, pour un trafic supposé de T octets, l'indice de confiance local c_j^i doit augmenter de la façon suivante :

$$c_j^i \leftarrow c_j^i + T \cdot r'_I \ , \tag{3.45}$$

où r'_I est un paramètre du système qui indique à quelle vitesse le degré de confiance local doit augmenter. De la même manière que pour le degré de suspicion local, nous proposons que l'indice de confiance décroisse linéairement avec le temps :

$$c_j^i \leftarrow \max(c_j^i - \Delta t \cdot r'_D, 0) \ , \tag{3.46}$$

où r'_D est un paramètre du système qui indique à quelle vitesse l'indice de confiance doit diminuer et Δt le temps écoulé depuis la dernière mise à jour.

L'indice de confiance C_j^i utilisé pour la publication est une valeur normalisée sur l'intervalle

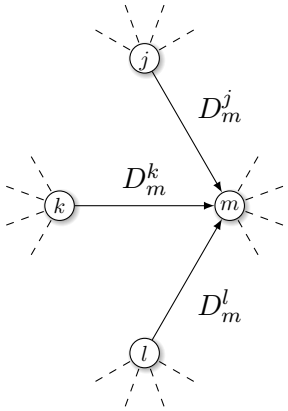


FIG. 3.10: Degrés de suspicion du nœud m sur les arcs incidents à son sommet.

$[0;1[$, calculée à partir de c_j^i de la manière suivante :

$$C_j^i = 1 - \frac{1}{1 + c_j^i \cdot r_A} \ , \tag{3.47}$$

où r_A est un paramètre du système qui indique à quelle vitesse l'indice de confiance publié doit tendre vers 1 lorsque l'indice de confiance local tend vers l'infini. L'indice de confiance est donc un nombre variant sur l'intervalle entre 0 (qui correspondant à « mon degré de suspicion ne vaut rien ») et 1 (qui correspond à « je suis certain de mon degré de suspicion »). Pour que ce degré de confiance puisse être interprété correctement dans le calcul du degré de suspicion global, il faut qu'on puisse exprimer le fait que toute entrée dans le calcul de médiane n'est pas équivalent (donc n'est pas de poids 1) mais doit être pondéré par le degré de confiance. Le calcul d'une médiane avec des entrées pondérées est appelée calcul de médiane pondérée et s'exprime comme suit. Pour tous les couples (D_j^i, C_j^i) récoltés à partir des messages topologiques des autres nœuds, on obtient pour chaque nœud j le degré de suspicion global W_j tel que

$$\forall j \in V, \quad \left\{ \left(D_j^0, C_j^0\right), \ldots, \left(D_j^n, C_j^n\right) \right\}, \qquad \begin{cases} \displaystyle\sum_{i:D_j^i<W_j} C_j^i \leq \frac{1}{2} \cdot \sum_i C_j^i \\ \displaystyle\sum_{i:D_j^i>W_j} C_j^i \leq \frac{1}{2} \cdot \sum_i C_j^i \ . \end{cases} \tag{3.48}$$

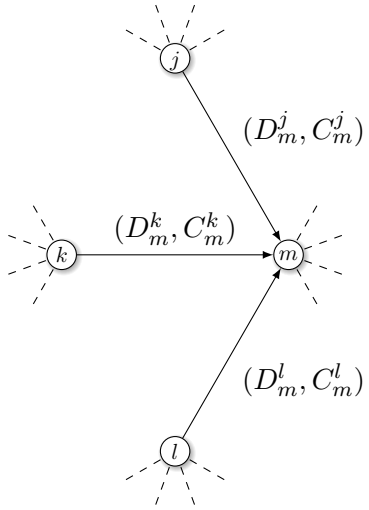


FIG. 3.11: Degrés de suspicion et degrés de confiance du nœud m sur les arcs incidents à son sommet.

3.2.3 Calcul de chemins de moindre suspicion

Le degré de suspicion global W_i est une métrique par nœud sur le graphe topologique maintenu en chaque nœud du réseau. Une manière simple de calculer des chemins qui minimisent à la fois le nombre de sauts et le degré de suspicion des nœuds consiste à définir des poids sur les liens de la manière suivante :

$$\forall i, j \in V, \qquad w_{ij} = 1 + f\left(W_i\right) \quad , \tag{3.49}$$

où w_{ij} est le poids du lien (i, j) et f est une fonction positive croissante qui exprime le surcoût à rajouter à un lien sortant pour un degré de suspicion global donné. Un algorithme de calcul de plus court chemin, tel que Dijkstra, peut être utilisé sur le graphe ainsi étiqueté pour calculer les chemins.

3.3 Transmissions et pertes des messages de contrôle

Dans un réseau sans fil, une transmission peut échouer pour diverses raisons telles que des interférences radio de source extérieure ou intérieure au réseau, des collisions avec d'autres transmissions, etc. Les protocoles de routage pour ces réseaux ont donc été adaptés pour parer à l'éventualité de la perte d'un message de contrôle. Or nous avons vu que la méthode présentée aux paragraphes 3.1 et 3.2 requiert d'un nœud i qu'il reçoive tous les ensembles de publications

de la part de j tant que ce dernier est considéré par i comme son voisin.

Si un message de contrôle émis par j et contenant un ensemble de publications se perd avant d'arriver à i, ce dernier ne peut plus effectuer les calculs nécessaires au maintien des bornes utilisées pour la vérification de la cohérence par lien. Le nœud i n'a plus qu'à réinitialiser ces paramètres et faire comme si j venait fraîchement d'arriver dans son voisinage. Il suffirait donc pour un nœud malveillant de ne pas envoyer un message sur deux pour ne pas se soumettre à la vérification tout en restant dans le voisinage.

3.3.1 Requêtes de retransmission

La détection de perte de messages périodiques est effectuée de deux manières : par le suivi des numéros de séquence et par expiration du délai maximum P séparant deux messages consécutifs. À tout moment, chaque nœud maintient un ensemble des numéros de séquence perdus pour chacun de ses voisins.

Définition 23 (numéros de séquence perdus) *Pour tout instant t, pour tout nœud i et pour tout voisin j, $M_j^i(t)$ est l'ensemble des numéros de séquence des ensembles de publications émis par j que i a perdus.*

Pour signifier à l'émetteur des messages perdus que ceux-ci doivent être réémis, les publications de lien sont augmentées de l'ensemble des numéros de séquence des messages perdus.

Définition 24 (publication de lien augmentée) *Pour tous nœuds i et j et pour tout numéro de séquence n, une publication de lien augmentée $\hat{L}_j^i(n)$ en provenance de i et concernant le lien $\{i, j\}$ est définie par :*

$$\forall i, j \in V, \quad \forall n \in \mathbb{N}, \qquad \hat{L}_j^i(n) = \left(n, V_j^i(n), R_j^i(n), M_j^i\left(\mathsf{T}_i(n)\right)\right) \tag{3.50}$$

Définition 25 (ensemble de publications augmentées) *Pour tout nœud i et pour tout numéro de séquence n, un ensemble de publications augmentées $\hat{A}^i(n)$ est défini par :*

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \hat{A}^i(n) = \bigcup_{j \in V} \left\{\hat{L}_j^i(n)\right\} \tag{3.51}$$

Lors de la réception d'une publication de lien augmentée depuis un voisin concernant le lien entre ce voisin et ce nœud, l'ensemble des numéros de séquence perdus est ajouté à l'ensemble des numéros de séquence des ensembles de publications à retransmettre. Lorsqu'un nœud i construit un message de numéro de séquence n, non seulement il ajoute $A^i(n)$ mais aussi tous les $A^i(n')$ pour lesquels n' est contenu dans l'ensemble des numéros de séquence des ensembles

de publications à retransmettre et il vide cet ensemble. Ceci est possible à condition que chaque nœud maintienne une liste finie des quelques derniers ensembles de publications émis.

3.3.2 Réordonnancement des ensembles retransmis

Un nœud doit soumettre à la vérification décrite aux paragraphes 3.1 et 3.2 les ensembles de publications dans l'ordre strict des numéros de séquence successifs pour que la méthode fonctionne. Les ensembles de publications pouvant être perdus puis retransmis, ils peuvent arriver dans le désordre et doivent donc être réordonnés.

Chaque nœud i maintient donc un compteur S^i_{jk} pour chacun des liens entre un voisin j et un de ses voisins k qui à tout instant contient le numéro de séquence de la dernière publication de lien émise par j pour le lien $\{j, k\}$ soumise à la vérification. Il maintient aussi une liste des publications de lien en attente d'être traitées Q^i_{jk}, triée par ordre croissant de leurs numéros de séquence.

À la réception d'un ensemble de publications $\hat{A}^j(n)$, i extrait les différents $\hat{L}^j_k(n)$ et les traite un par un. Pour chaque publication de lien, il applique l'algorithme 1. Il se divise en quatre phases comme suit :

1) ligne 1 : si la publication a déjà été reçue, abandonner ;
2) lignes 3–10 : tenter de replacer les VCP inverses dans la publication la plus ancienne qui les a déclarées comme perdues ;
3) lignes 11–16 : tenter de replacer les VCP inverses déclarées comme perdues à partir des publications ultérieures ;
4) lignes 17–22 : soumettre toutes les publications complètes suivant exactement la dernière vérifiée.

Bien entendu, cette technique a un coût en termes de surcharge de contrôle, puisque la taille des messages de contrôle est potentiellement plus importante. Nous avons modélisé ce protocole pour l'éprouver dans le model checker Spin [55] et nous avons pu exhiber un cas particulier dans lequel il diverge. Lorsqu'exactement un paquet sur deux est perdu, le nombre de publications à retransmettre croît linéairement avec le temps. Cependant ce cas de figure est peu probable dans la pratique puisqu'au bout d'un certain temps, le cycle des pertes peut être cassé et les publications retransmises reçues. Dans les autres cas de fonctionnement courant, Q^i_{jk} ne grandit pas de façon indéfinie puisque lorsqu'un voisin est perdu, la liste des publications incomplètes en attente est détruite avec toutes les données internes concernant ce voisin, excepté le degré de suspicion et l'indice de confiance locaux. Enfin la liste des ensembles de publications émis est aussi de taille limitée, puisqu'il n'y a pas de raison qu'un voisin déclare comme perdue une publication trop ancienne.

Algorithme 1 : Réordonnancement des publications de lien

Entrée : publication de lien $(n, V_k^j(n), R_k^j(n), M_k^j(n))$,
dernier numéro de séquence traité S_{jk}^i,
liste de publications de lien en attente Q_{jk}^i.

si $n \leq \mathsf{S}_{jk}^i \vee (n, V_k^j(n), R_k^j(n), M_k^j(n)) \in \mathsf{Q}_{jk}^i$ **alors retourner**
$r \leftarrow \emptyset$
pour tout $n' < n : (n', V_k^j(n'), R_k^j(n'), M_k^j(n')) \in \mathsf{Q}_{jk}^i$ **faire**
  pour tout $m' \in M_k^j(n') : (m', V_j^k(m')) \in R_k^j(n)$ **faire**
    $R_k^j(n') \leftarrow R_k^j(n') \cup \{(m', V_j^k(m'))\}$
    $R_k^j(n) \leftarrow R_k^j(n) \backslash \{(m', V_j^k(m'))\}$
    $r \leftarrow r \cup \{m'\}$
  $M_k^j(n') \leftarrow M_k^j(n') \backslash r$
  pour tout $m' \in M_k^j(n) : (m', V_j^k(m')) \in R_k^j(n')$ **faire**
    $M_k^j(n) \leftarrow M_k^j(n) \backslash \{m'\}$
pour tout $n' > n : (n', V_k^j(n'), R_k^j(n'), M_k^j(n')) \in \mathsf{Q}_{jk}^i$ **faire**
  $M_k^j(n') \leftarrow M_k^j(n') \backslash r$
  pour tout $m' \in M_k^j(n) : (m', V_j^k(n')) \in R_k^j(n')$ **faire**
    $R_k^j(n) \leftarrow R_k^j(n) \cup \{(m', V_j^k(m'))\}$
    $R_k^j(n') \leftarrow R_k^j(n') \backslash \{(m', V_j^k(m'))\}$
    $M_k^j(n) \leftarrow M_k^j(n) \backslash \{m'\}$
tant que $\mathsf{Q}_{jk}^i \neq \emptyset$ **faire**
  soit $(n', V_k^j(n'), R_k^j(n'), M_k^j(n'))$ le premier élément de Q_{jk}^i
  si $n' \neq \mathsf{S}_{jk}^i + 1$ *or* $M_k^j(n') \neq \emptyset$ **alors retourner**
  $\mathsf{Q}_{jk}^i \leftarrow \mathsf{Q}_{jk}^i \backslash \{(n', V_k^j(n'), R_k^j(n'), M_k^j(n'))\}$
  $\mathsf{S}_{jk}^i \leftarrow \mathsf{S}_{jk}^i + 1$
  vérifier $(n', V_k^j(n'), R_k^j(n'))$

3.4 Réseaux denses et petits messages de contrôle

Le problème principal posé par le surcoût de contrôle de la méthode des paragraphes 3.1, 3.2 et 3.3 est que ce dernier dépend du nombre de voisins à déclarer par nœud dans les messages de contrôle du protocole de routage par état de lien. Or en général, les paquets contenant les messages de contrôle ont une taille maximale limitée, ce qui peut devenir gênant lorsque la densité du réseau est trop importante pour que les valeurs des compteurs de tous les liens puissent toutes tenir dans un seul message.

Les protocoles de routage par état de lien sont prévus pour parer à cette éventualité en permettant aux messages de contrôle de ne contenir qu'une partie des informations à publier, tant que les informations concernant chaque voisin sont publiées suffisamment souvent pour ne pas expirer au niveau des nœuds à qui elle sont destinées. On peut donc considérer que la fréquence d'émission des messages est variable pour s'adapter à la quantité d'information à publier.

Si on prend le cas où tous les voisins n'ont pas la place d'être publiés dans un message de contrôle, alors si les valeurs des compteurs sont toujours relevées au moment de la publication d'un lien, la vérification du bilan par nœud selon (3.17), n'est plus toujours possible. En effet, on ne dispose plus que de valeurs relevées sur des intervalles différents. On remarque cependant que la vérification de la cohérence par lien reste possible.

Dans les paragraphes suivants, nous allons considérer deux approches possibles pour résoudre ce problème.

3.4.1 Publications partielles simples

Pour que le calcul du bilan par nœud reste possible, il faut disposer de valeurs de compteurs relevées sur un même intervalle. Si un certain $\hat{A}^i(n)$ est trop grand pour être compris dans un seul message, il doit être découpé et envoyé dans plusieurs messages successifs. Ceci nécessite alors l'utilisation d'un numéro de séquence propre à chaque publication de lien, indépendemment du numéro de séquence n de l'ensemble des publications. On redéfinit donc $\hat{A}^i(n)$ comme étant :

$$\forall i \in V, \quad \forall n \in \mathbb{N}, \qquad \hat{A}^i(n) = \bigcup_{j \in \mathsf{A}^i(n)} \left\{ \hat{L}^i_j \left(\mathsf{S}^i_j \left(\mathsf{T}_i(n) \right) \right) \right\} \ , \tag{3.52}$$

où $\mathsf{A}^i(n)$ est l'ensemble des voisins de i publiés dans son message numéro n et $\mathsf{S}^i_j(t)$ est le numéro de séquence de la publication de lien concernant $\{i, j\}$ à l'instant t.

La figure 3.12 illustre le fonctionnement de cette méthode sur un exemple. Lorsque le nombre de liens à publier augmente, la fréquence des messages de publications partielles augmente également. Bien que la cohérence par lien peut être effectuée dès la réception d'une

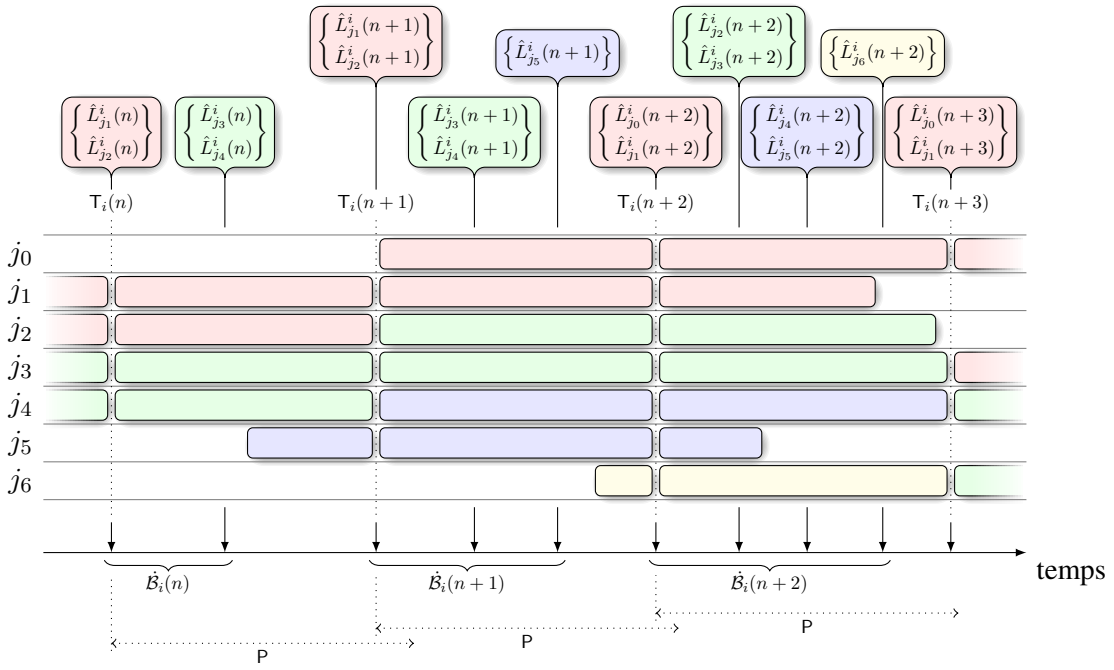


FIG. 3.12: Publications partielles simples : les lignes j_0 à j_6 représentent la présence de chacun des sept voisins de i sur l'intervalle de temps entre $\mathsf{T}_i(n)$ et $\mathsf{T}_i(n+3)$ et les panneaux fléchés indiquent à quel moment quelle information est émise par i. Les compteurs sont tous relevés à des instants séparés d'au plus P unités de temps et plusieurs messages (chacun ne pouvant contenir que deux publications de lien) sont utilisés pour diffuser l'ensemble des publications de lien (les rectangles sur les lignes). Les bilans par nœuds sont effectués une fois que toutes les publications partielles de même numéro de séquence sont reçues.

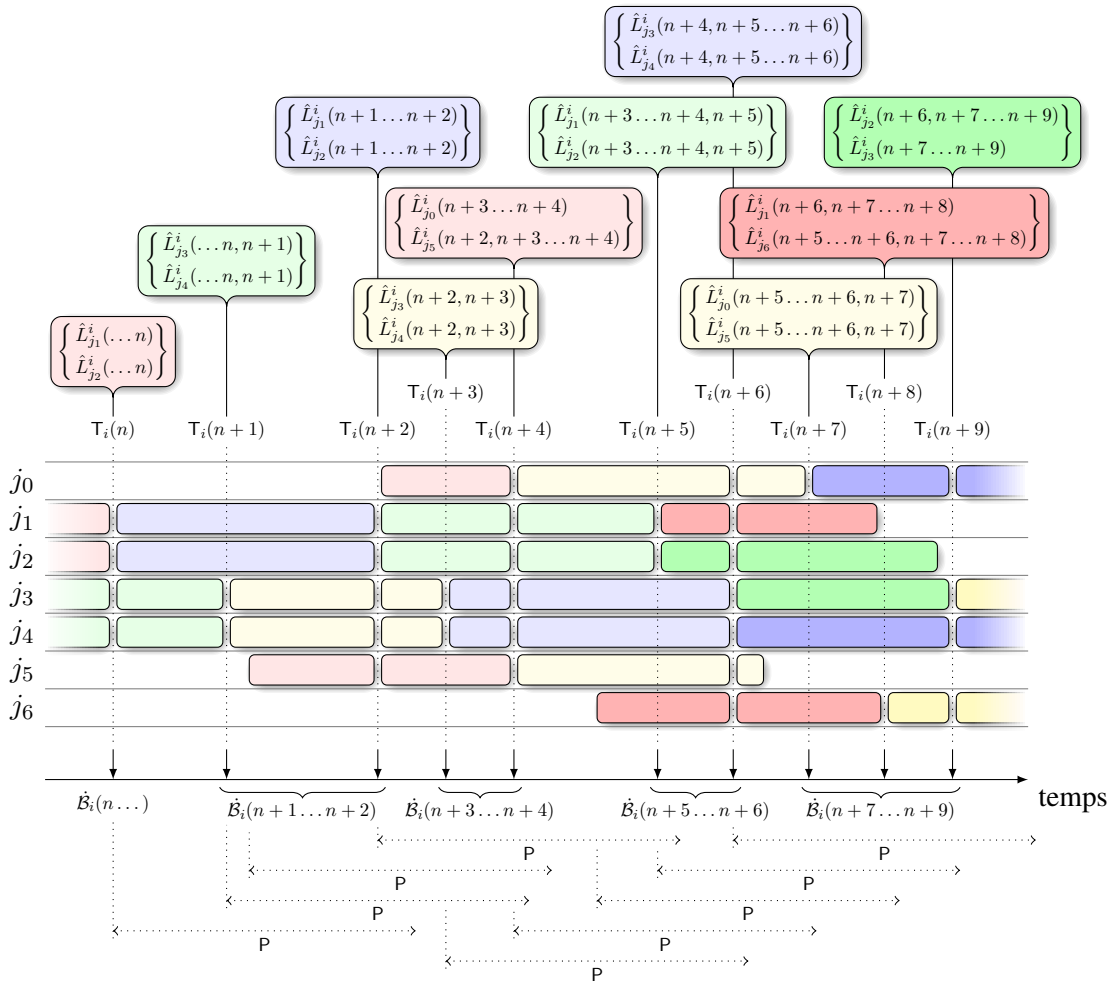


FIG. 3.13: Publications partielles doubles : les lignes j_0 à j_6 représentent la présence de chacun des sept voisins de i sur l'intervalle de temps entre $\mathsf{T}_i(n)$ et $\mathsf{T}_i(n+9)$ et les panneaux fléchés indiquent à quel moment quelle information est émise par i. Les compteurs sont tous relevés simultanément dès que toutes les valeurs du relevé simultané précédent ont été émises ainsi qu'individuellement pour chaque lien avant l'émission de sa publication. Les instants d'émission des messages sont programmés de façon à ce que chaque lien ne soit publié pas plus tard que P unité de temps après sa dernière publication. Les bilans par nœuds sont effectués une fois que toutes les publications partielles entre deux relevés simultanés sont reçues.

publication partielle, la cohérence par nœud n'est possible qu'après réception de toutes les publications partielles correspondant au même relevé.

Si on veut pouvoir vérifier la cohérence par nœud au plus tard P unités de temps après le relevé des compteurs, la capacité de certains messages de publications partielles sera sous-utilisée, comme l'illustrent les messages $\{L^i_{j_5}(n+1)\}$ et $\{L^i_{j_6}(n+2)\}$.

3.4.2 Publications partielles doubles

Pour éviter d'introduire un retard dans la vérification de cohérence par lien, il faut donc trouver un moyen de rendre disponibles les valeurs des compteurs les plus « fraîches » tout en permettant de calculer le bilan par nœud. Une solution intéressante est de diviser les valeurs publiées en deux : la limite du découpage se faisant au même moment pour tous les liens. Autrement dit, on associe une variable à chaque compteur destinée à accueillir la valeur courante du compteur au moment d'un relevé simultané. Pour chaque lien publié, on diffuse la valeur du compteur au moment du relevé simultané (telle que gardée dans la variable) et la valeur courante depuis ce relevé. Dans tous les cas, à chaque relevé, qu'il soit simultané ou non, les compteurs sont remis à zéro.

On redéfinit donc une publication de lien comme suit :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \hat{L}^i_j(n) = \left(n, V'^i_j(n), V^i_j(n), R^i_j(n), M^i_j\left(\mathsf{T}_i(n)\right)\right) \quad , \tag{3.53}$$

où $V'^i_j(n)$ est l'ensemble des valeurs des variables associées aux compteurs des liens $\{i,j\}$.

Le fonctionnement est illustré sur la figure 3.13. On voit bien que les voisins d'un nœud sont toujours capables de calculer le bilan par nœud, bien qu'avec du retard et les bilans par lien sans retard.

Un nœud doit effectuer le relevé simultané de ses compteurs aussitôt que toutes les variables associées ont été publiées. Ainsi lorsque tous les liens tiennent à la fois dans un message, les compteurs sont relevés à chaque envoi.

Les instants d'émission, quant à eux, sont déterminés de façon à s'assurer que les émissions de publications d'un même lien ne soient pas espacées de plus de P unités de temps. Une façon de faire est de maintenir une liste triée de moments d'émission des prochaines publications de liens :

$$\left(\ldots, \left(\mathsf{T}_i(n), \{j_k, \ldots\}\right), \ldots\right) \tag{3.54}$$

qui contient des paires qui associent à un ensemble de voisins l'instant auquel la publication du lien doit être effectuée. La gestion de cette liste permet d'optimiser le contenu des messages afin de réduire le nombre d'émissions au minimum. Lorsqu'un nouveau lien apparaît dont les

compteurs doivent être publiés, on a le choix d'utiliser un message déjà programmé dans lequel il reste de la place ou bien de rajouter une nouvelle entrée dans la liste, quitte à avancer dans le temps les messages précédent pour éviter des émissions trop fréquentes.

3.4.3 Réception des publications partielles

Lorsqu'un nœud génère un message de contrôle qui contient des publications doubles, il se peut très bien qu'il y inclue des liens dont les variables ont été affectées à des moments différents. En effet, un relevé simultané peut survenir alors qu'il reste encore de la place dans le message en cours de construction et il n'y a a priori pas de raison de ne pas inclure davantage de publications pour des liens qui n'ont pas encore été publiés dans ce message.

Pour un voisin qui reçoit ces publications, il n'est pas toujours possible de reconstruire les bilans par nœud sans ambiguïté. Il ne peut pas savoir si un relevé simultané a eu lieu au cours de la construction du message et si tel est le cas, quels sont les liens pour lesquels les nouvelles valeurs sont publiées.

Une chose certaine est qu'il n'est pas possible qu'un relevé simultané survienne plus d'une fois durant la construction d'un message (autrement il existe au moins un lien qui est publié deux fois). Par conséquent, pour chaque publication dans un message, de deux choses l'une : soit celui-ci précède le relevé simultané, soit il lui succède. Il suffit donc que l'émetteur du message inclue un bit d'information par publication qui permet de différencier ces deux cas. On peut par exemple définir ce bit comme étant une couleur pouvant prendre les valeurs *bleu* ou *rouge* et maintenir au niveau de l'émetteur la couleur courante avec laquelle estampiller chaque publication. Lorsqu'un relevé a lieu, la couleur courante bascule de *bleu* vers *rouge* et vice-versa. Un nœud qui reçoit ces messages calculera donc des bilans par nœud des valeurs *bleues*, puis *rouges*, puis à nouveau *bleues* et ainsi de suite. Si le message est le premier reçu par le voisin, alors il doit attendre le basculement de la couleur pour commencer à calculer les bilans par nœud (il n'a aucun moyen de savoir si le dernier basculement a eu lieu à la toute fin de la génération du message précédent ou avant).

3.5 Authentification des messages

Notre méthode repose sur la capacité des nœuds à authentifier les émetteurs des données dans les publications contenues dans les messages. On suppose donc que l'infrastructure nécessaire à la vérification de l'authenticité et de l'intégrité de toute information signée numériquement est mise en place. Nous avons vu précédemment que les ensembles de publications contiennent des informations de première main (les VCP) et de seconde main (les VCP inverses). Toute

information de première main peut être vérifiée par une signature commune à tout l'ensemble des publications, alors que les informations de seconde main nécessitent des signatures séparées. Si un nœud reçoit par exemple l'ensemble $A^i(n)$, il doit pouvoir vérifier pour chaque lien entre i et un de ses voisins j la publications de lien $L^i_j(n)$ qui à son tour contient, entre autres, les VCP inverses $R^i_j(n)$ dont l'origine est de seconde main (j en l'occurrence). Pour permettre aux nœuds ayant reçu $A^i(n)$ de vérifier tous les $R^i_j(n)$ qu'il contient, ces derniers doivent être signés par les j correspondants. Autrement dit, au lieu de $R^i_j(n)$, i devrait fournir $\bar{R}^i_j(n)$, la version signée de $R^i_j(n)$, ce qui implique que i dispose de la signature générée par j lui-même. La solution consiste donc en ce que chaque nœud i fournisse $\bar{L}^i_j(n)$, la version signée de $\hat{L}^i_j(n)$ dans son ensemble de publications :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \bar{L}^i_j(n) = \left(n, V'^i_j(n), V^i_j(n), S^i\left(V''^i_j(n)\right), R^i_j(n), M^i_j\left(\mathsf{T}_i(n)\right)\right) \quad , \tag{3.55}$$

où $V''^i_j(n)$ est la somme, composante par composante, de $V'^i_j(n)$ et $V^i_j(n)$. Enfin l'ensemble des VCP inverses est défini comme suit :

$$\forall i,j \in V, \quad \forall n \in \mathbb{N}, \qquad \bar{R}^i_j(n) = \bigcup_{m \in S^i_j(n)} \left\{\left(V''^j_i(m), S^j\left(V''^j_i(m)\right)\right)\right\} \quad , \tag{3.56}$$

où S^k est la fonction utilisée par k pour générer une signature. Comme les valeurs distinctes des variables associées et des compteurs ne sont pas nécessaires pour les VCP inverses, elles sont sommées composante par composante par le nœud qui les reçoit, alors que les la signature est fournie directement par le nœud émetteur.

3.6 Application à OLSR

Nous avons cherché à appliquer la méthode de résilience aux pertes intempestives au protocole de routage proactif à état de lien OLSR. Ce protocole remplit évidemment toutes les conditions requises pour l'utilisation de notre méthode et offre en plus certaines fonctionnalités que nous avons pu exploiter.

3.6.1 Prérequis

Pour que l'application à OLSR ait du sens en pratique, il faut nécessairement que plusieurs choses soient mises en place. Premièrement, tous les paquets de contrôle ainsi que de données doivent être signés par leur émetteur et cette signature doit être vérifiable par tous les nœuds. Ceci suppose donc que, deuxièmement, un schéma d'authentification soit mis en place, comme

une infrastructure de clef publique distribuée ou une distribution de clefs préalable.

Troisièmement, pour que le routage saut-par-saut soit résistant aux envois de fausses informations dans les messages de contrôle, l'authentification de l'émetteur des messages n'est pas suffisant en soi. Il faut que des méthodes de vérification du contenu des messages soit mis en place, tels que celles évoquées au chapitre 2, paragraphe 2.3.5.

3.6.2 Publications dans les messages HELLO

Tout naturellement, les publications de lien sont incluses dans les messages HELLO. Ainsi à chaque lien (au sens OLSR) contenu dans un message HELLO, est rajouté la publication de lien (au sens de notre méthode).

Le protocole OLSR est prévu pour supporter un taux de perte non nul de ses messages de contrôle. Par conséquent, la méthode du paragraphe 3.3 doit être utilisée pour s'assurer que toutes les publications sont bien transmises.

Comme la taille des paquets pouvant être transmis sur un lien est en général limitée par les couches inférieures, OLSR est aussi prévu pour pouvoir envoyer des messages de contrôle *partiels*, ne contenant qu'une partie de l'ensemble des informations à publier. Si toutes les informations ne peuvent pas être transmises dans un seul message, alors plusieurs messages doivent être générés pour que les informations n'expirent pas au niveau de leur destinataire entre deux publications. Ainsi la méthode de publications partielles du paragraphe 3.4.2 doit être appliquée.

3.6.3 Sélection des MPRs

Le rôle des MPRs dans OLSR est triple :
- retransmettre les messages diffusés ;
- publier des liens topologiques vers leurs MPR-selectors ;
- retransmettre le trafic de données vers leurs MPR-selectors.

Ainsi la méthode utilisée pour les sélectionner peut avoir un impact fort sur le réseau, puisqu'elle décide de quels nœuds seront sources de messages TC et donc les extrémités sources des arcs dans le graphe topologique maintenu en chaque nœud.

Dans la spécification de référence du protocole, l'heuristique de sélection des MPRs a pour impératif de permettre de joindre tout voisin à deux sauts tout en maintenant l'ensemble de MPRs le plus petit possible. Le nombre total de MPRs détermine non seulement le nombre de retransmissions des messages diffusés dans le réseau, mais représente directement le nombre de nœuds qui génèrent les messages TC.

Lorsque les nœuds disposent d'un degré de suspicion pour chacun de leurs voisins, il est souhaitable de l'utiliser dans l'heuristique de sélection des MPRs. Le but est alors non seulement

de joindre tous les voisins à deux sauts avec le minimum de MPRs, mais aussi de le faire avec une probabilité de perte minimale.

Les MPRs ajoutent aux entrées de leurs messages TC les degrés de suspicions ainsi que les indices de confiance correspondant. Ainsi quand un nœud choisit un voisin comme MPR, il donne à celui-ci le pouvoir de diffuser son degré de suspicion le concernant.

Si on représente par $\mathcal{N}_i$ l'ensemble des voisins symétriques de i (voisins à un saut), par $\mathcal{N}_i^2$ l'ensemble des voisins symétriques des voisins symétriques de i qui ne sont pas aussi des voisins symétriques de i (les voisins à deux sauts stricts de i)

$$\forall i \in V, \qquad \mathcal{N}_i^2 = \bigcup_{j \in \mathcal{N}_i} \Big(\mathcal{N}_j \setminus \big(\mathcal{N}_i \cup \{i\} \big) \Big) \tag{3.57}$$

et par $\mathcal{M}_i$ l'ensemble des MPRs de i, alors une heuristique possible pour calculer $\mathcal{M}_i$ est l'algorithme 2. Son fonctionnement est le suivant :

1) ligne 1 : c est l'ensemble des voisins candidats à l'élection des MPRs ;
2) ligne 2 : n est l'ensemble des voisins à deux sauts qui doivent être joignables par au moins un MPR ;
3) lignes 5–17 : recherche du candidat de degré de suspicion minimal, de nombre de voisins à deux saut pas encore joints maximal, de degré (au sens d'un sommet dans un graphe) maximal ;
4) lignes 18–20 : ajout du candidat élu dans l'ensemble des MPRs et retrait de l'ensemble des candidats, retrait de tous les voisins à deux sauts joints de l'ensemble des voisins qui doivent être joignables.

3.6.4 Diffusion des messages TC et calcul de chemins

Quand un nœud reçoit un message TC, il met à jour sa vision de la topologie du réseau avec les informations qui y sont contenues. Ici, en plus de rajouter des arcs, entre le MPR émetteur du message et ses MPR-selectors, il rajoute aussi les degrés de suspicion et indices de confiance associés. Chaque nœud maintient donc un graphe orienté et aux arcs étiquetés de la topologie du réseau.

Lorsqu'un nœud doit recalculer sa table de routage, il doit pour chaque sommet du graphe topologique calculer le poids correspondant à la médiane pondérée des degrés de suspicion le concernant. Il calcule donc cette valeur en considérant les étiquettes de tous les arcs incidents au sommet. Si le nœud sujet du calcul est un MPR et que le sommet objet du calcul est un de ses MPR-selectors, alors la médiane pondérée doit aussi prendre en compte les derniers degré de suspicion et indice de confiance diffusés par le nœud, afin que le calcul reste uniforme. Autrement

Algorithme 2 : Sélection des MPRs de moindre degré de suspicion

Entrée : nœud courant i,
voisins à un saut $\mathcal{N}_i$,
voisins à deux sauts stricts $\mathcal{N}_i^2$,
vecteur de degrés de suspicion $\boldsymbol{D}^i = (\dots, D_j^i, \dots)$.
Sortie : ensemble de MPRs $\mathcal{M}_i$.

$\mathcal{M}_i \leftarrow \emptyset$
$c \leftarrow \{j \in \mathcal{N}_i : \mathcal{N}_j \setminus \{\mathcal{N}_i \cup \{i\}\} \neq \emptyset\}$
$n \leftarrow \mathcal{N}_i^2$
tant que $n \neq \emptyset$ **faire**
  $f \leftarrow \mathit{Vrai}$
  $d \leftarrow 0$
  $r \leftarrow 0$
  $x \leftarrow 0$
  pour tout $j \in c$ **faire**
    $r_j \leftarrow |\mathcal{N}_j \cap n|$
    $x_j \leftarrow |\mathcal{N}_j|$
    si $f \vee D_j^i < d \vee D_j^i = d \wedge (r_j > r \vee r_j = r \wedge x_j < x)$ **alors**
      $f \leftarrow \mathit{Faux}$
      $d \leftarrow D_j^i$
      $r \leftarrow r_j$
      $x \leftarrow x_j$
      $m \leftarrow j$
  $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \{m\}$
  $c \leftarrow c \setminus \{m\}$
  $n \leftarrow n \setminus \mathcal{N}_m$

dit, il faut que le calcul de chaque nœud concernant le degré de suspicion global soit effectué en utilisant les mêmes données. Chaque MPR doit garder, pour chacun de ses MPR-selectors, les derniers degré de suspicion et indice de confiance publiés dans un message TC. Enfin le calcul des chemins se fait par un algorithme de plus court chemin sur le graphe topologique sur lequel chaque arc (i, j) est pondéré par w_{ij} tel que

$$w_{ij} = 1 + W_i \ , \tag{3.58}$$

où W_i est le degré de suspicion global du nœud i. On peut voir cette expression comme un moyen de *pénaliser* les arcs sortant d'un nœud suspect, de façon à privilégier les chemins passant par les nœuds les moins suspects possibles, tout en cherchant à garder des chemins courts.

3.7 Conclusion

Nous avons présenté dans ce chapitre une méthode permettant de rendre un réseau ad hoc mobile à protocole de routage à état de lien résistant à la présence de nœuds malveillants causant des pertes intempestives de paquets de données. La méthode est basée sur la vérification de conservation de flot dans le réseau et repose essentiellement sur l'ajout d'informations dans les messages de contrôle. La méthode ne dépend pas notamment de fonctionnalités spécifiques offertes par les couches MAC ou physique sous-jacentes.

3.7.1 Contributions

Le principal apport de notre solution est de mettre en œuvre la vérification de conservation de flot dans un réseau ad hoc mobile, sans pour autant nécessiter de synchronisation entre les nœuds. Cette caractéristique fait que la méthode est aisément intégrable dans un protocole existant, comme nous l'avons décrit pour le cas d'OLSR. La principale difficulté a été de trouver une manière plus stricte de vérifier la cohérence par lien des valeurs des compteurs, sachant que ceux-ci sont comptabilisés sur des intervalles de temps différents. Nous sommes arrivés à exhiber des bornes déduites des valeurs des compteurs elles-mêmes, que ces dernières doivent satisfaire si elles sont cohérentes.

Le découpage en étapes successives et relativement indépendantes (comptage des paquets et diffusion des compteurs, vérification des cohérences, maintenance et diffusion du degré de suspicion local, agrégation des degrés locaux et calcul du degré global, routage avec qualité de service), permet plus de souplesse dans la mise en œuvre et surtout l'intégration de résultats de méthodes supplémentaires, existantes ou bien à venir. C'est notamment grâce à cette approche que nous avons pu facilement faire en sorte que les nœuds suspectés de causer des pertes inten-

tionnelles ne sont pas systématiquement exclus du réseau mais seulement évités, ce qui répond bien aux attaques *kamikaze*.

Le calcul du degré de suspicion global formulé comme une médiane pondérée par les indices de confiance est une façon efficace, d'une part, de privilégier dans le calcul les degrés locaux établis en présence d'une plus grande quantité de trafic (que ces degrés soient forts ou faibles) et d'autre part, de se prémunir contre les publications de degré à caractère diffamatoire émises par les nœuds malveillants eux-mêmes.

Enfin les aspects pratiques de mise en œuvre de la diffusion des compteurs dans des messages de taille limitée ont été considérés. Nous avons donc proposé une méthode de diffusion partielles de valeurs de compteurs qui permet néanmoins le calcul des bilans par nœud et par lien avec un décalage limité dans le temps. De même, nous avons proposé une méthode de retransmission de messages perdus qui, bien qu'elle soit faillible dans certains cas très particuliers, permet de faire fonctionner notre solution dans des mises en œuvres réalistes. Enfin nous avons donné un moyen optimisé de rendre l'authentification des messages possibles.

3.7.2 Analyse

La méthode de vérification de la cohérence par nœud est une équation découlant directement de l'énoncé du principe de conservation de flot. Par conséquent, l'équation est vérifiée tant que le principe de conservation de flot est respecté. Par la contraposée, aussitôt que l'équation n'est pas vérifiée, le principe de conservation de flot non plus et on a une preuve que le nœud testé a détruit un ou plusieurs paquets de données. Cependant, cette implication ne garantit en aucun cas qu'un nœud pour lequel l'équation est vérifiée n'a pas détruit de paquet de données.

La méthode de vérification de cohérence par lien, nécessaire pour s'assurer qu'un nœud ne publie pas de fausses valeurs de compteurs (ou bien que des paquets de données n'ont pas été perdus après avoir été comptabilisés en sortie d'un nœud et avant d'avoir été comptabilisés en entrée de son voisin) est basée sur l'accumulation des différences et la vérification du respect de bornes déduites des valeurs publiées elles-mêmes. De même que pour la vérification de cohérence par nœud, ici l'implication est à sens unique : tant que les valeurs des compteurs sont cohérentes, les bornes sont respectées et par la contraposée, le non-respect de ces bornes est une preuve de non cohérence des valeurs des compteurs. Cette implication n'entraîne aucunement l'impossibilité du respect des bornes malgré une incohérence des valeurs.

Une recherche exhaustive, par exemple à l'aide d'outils de *model checking* permettrait d'identifier les séquences d'échange de données et de publications telles qu'une incohérence de compteurs passe inaperçue du point de vue des vérifications. Une telle recherche est cependant difficile à mettre en œuvre étant donnée la taille considérable de l'espace à explorer. Nous avons toutefois

le sentiment que de telles séquences ne pourraient pas contenir un nombre arbitrairement grand de destructions de paquets et donc une incohérence de compteurs strictement croissante. Nous pensons que la mise en place d'une stratégie de triche exploitant ces faiblesses devient d'autant plus difficile que le nombre des paquets détruits grandit.

Concernant les étapes ultérieures de notre solution, leur analyse formelle n'est pas aisée sans modèle précis de la nature des données entrantes (c'est-à-dire les instants de détection d'incohérence et leurs valeurs). Pour cette raison, nous avons jugé plus intéressant de mettre en œuvre la solution dans un simulateur et de mesurer les performances du système dans divers cas de figure. C'est donc précisément ce que nous décrivons dans le prochain chapitre, qui a pour but l'étude qualitative et quantitative du fonctionnement de la solution dans l'application au protocole OLSR.

Chapitre 4

Évaluation des performances

Dans les réseaux ad hoc mobiles, les ressources limitées, tant au niveau de la capacité du médium radio que de la mémoire et de la puissance de traitement des nœuds, font que le surcoût engendré par l'application d'une solution est au moins aussi important que l'amélioration qu'elle apporte.

Le caractère dynamique de la topologie, la nature distribuée des algorithmes et les nombreux paramètres influant sur les conditions de fonctionnement du réseau font que l'étude analytique de l'impact d'une solution est souvent trop compliquée voire impossible. Dans ces situations, nous avons recours à la simulation qui, loin de fournir une quelconque preuve de correction ni même une mesure fiable des performances, permet d'avoir un ordre d'idée du comportement du système par comparaison à un système similaire dans des conditions sinon identiques.

Dans le cas de notre solution, la simulation a été notre principal outil d'évaluation et de perfectionnement. Ce chapitre présente le modèle utilisé, les différents aspects des performances sur lesquels s'est portée notre attention et enfin les résultats obtenus et les conclusions que l'on peut en tirer.

4.1 Modèle de simulation

Nous avons développé notre modèle de simulation pour le simulateur OPNET 12.0, composé de trois composants interchangeables prenant en charge la couche applicative, le protocole de routage et les couches de transmission et mobilité respectivement. L'approche de la mise en œuvre du modèle est telle que seul l'ordonnanceur d'OPNET, ses files d'attente et ses mécanismes de distributions de probabilité sont utilisés. Ainsi le besoin de reprogrammer nous-mêmes les différents composants est contrebalancé par le fait que le modèle supporte des scénarios à très grand nombre de nœuds.

4.1.1 Gestion de la mobilité

Le mouvement des nœuds est géré par la couche de transmission et de mobilité. Il est modélisé selon le principe de la *marche aléatoire* modifié et adapté aux réseaux ad hoc [56], illustré sur la figure 4.1. À chaque nœud sont associés une position sur une surface rectangulaire et un vecteur déplacement (cap et vitesse). Chaque nœud se déplace à vecteur déplacement constant (cap et vitesse constants) durant un intervalle de temps tiré aléatoirement (selon une loi exponentielle de paramètre λ_m, c'est là un aspect différent de la marche aléatoire classique, dans laquelle les intervalles de temps sont constants). À la fin de l'intervalle de déplacement constant, un nouvel intervalle ainsi qu'un nouveau vecteur déplacement sont tirés aléatoirement (le cap et la vitesse tous deux tirés uniformément, respectivement sur $[0;2\pi[$ et $[v_{min};v_{max}]$).

La surface sur laquelle les nœuds évoluent ayant des dimensions paramétrables mais toujours finies, le comportement des nœuds aux limites de la surface est particulier. Parmi les approches possibles pour gérer ces cas de figure [57] , nous avons fait le choix des *bords élastiques* qui consiste en ce que les nœuds rebondissent sur les bords de la surface (voir figure 4.2). Autrement dit, si la surface est un rectangle de largeur x_{max} et de hauteur y_{max}, alors à chacune des composantes de toute position calculée (x, y), on applique la fonction b définie par

$$b : (x, x_{\max}) \mapsto \begin{cases} -x - x_{\max} \cdot \left\lfloor \frac{-x}{x_{\max}} \right\rfloor & \text{si } x < 0 \\ -x + x_{\max} \cdot \left(1 + \left\lfloor \frac{x}{x_{\max}} \right\rfloor\right) & \text{si } x > x_{\max} \\ x & \text{sinon.} \end{cases} \tag{4.1}$$

Pour calculer la position instantanée (x_i, y_i) au temps t d'un nœud i dont le vecteur déplacement est (v_i, θ_i) et dont le dernier changement de mobilité a été effectué au temps t' à la position (x'_i, y'_i) on utilise la relation suivante :

$$\begin{aligned} x_i &= b\left(x'_i + v_i \cdot \left(t - t'\right) \cdot \cos\theta_i, x_{max}\right) \\ y_i &= b\left(y'_i + v_i \cdot \left(t - t'\right) \cdot \sin\theta_i, y_{max}\right) \quad . \end{aligned} \tag{4.2}$$

Cette approche présente l'avantage de ne pas forcer des changements trop brusques et irréalistes dans la topologie par rapport aux approches qui identifient les bords deux à deux et font qu'un nœud peut traverser un bord et se retrouver à l'autre extrémité de la surface.

4.1.2 Couche physique

Le modèle utilisé pour la transmission consiste à calculer en chaque nœud les niveaux de puissance du signal et du bruit. Le bruit est ici simplement la somme des puissances des signaux

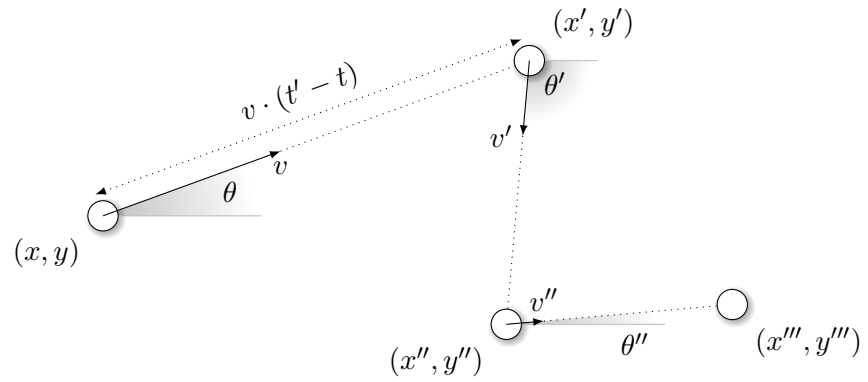


FIG. 4.1: Le modèle de mobilité *Random Ad Hoc Mobility*. Durant un intervalle de temps tiré aléatoirement, le mobile se déplace linéairement selon un vecteur déplacement (v, θ) constant. À la fin de l'intervalle, un nouvel intervalle et un nouveau vecteur sont tirés aléatoirement.

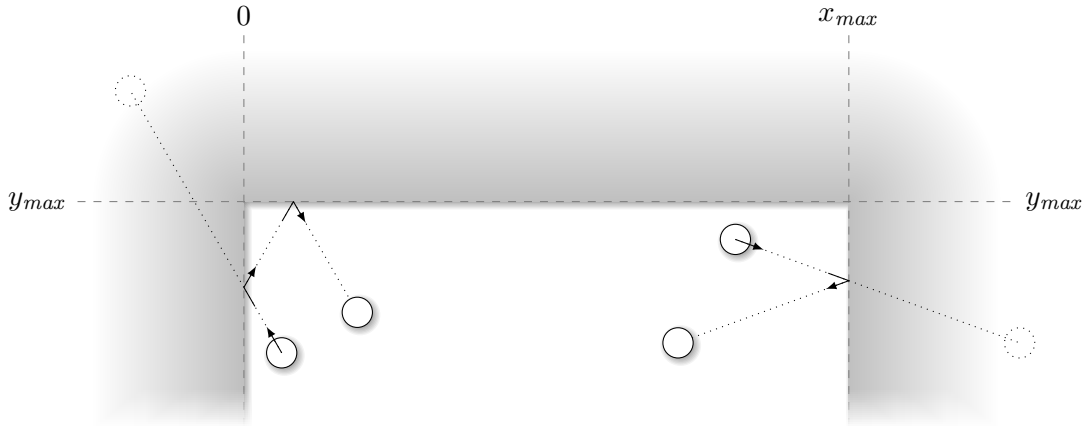


FIG. 4.2: Surface à bords élastiques : les nœuds rebondissent sur les limites de la surfaces où ils évoluent.

que le nœud n'est pas en train de décoder. Pour calculer la puissance au niveau du récepteur d'un signal émis par un émetteur, la formule suivante, tirée de l'équation des télécommunications de Friis [58], est utilisée :

$$P_r = P_t \cdot \left(\frac{\lambda}{4\pi d}\right)^2 \quad , \tag{4.3}$$

où P_t et P_r sont respectivement les puissances d'émission et de réception (en Watts), λ est la longueur d'onde de la porteuse utilisée (en mètres) et d est la distance séparant l'émetteur du récepteur (en mètres).

Simulation des interférences

Lorsqu'un nœud transmet une trame, alors en tout autre nœud, soit les conditions sont telles qu'un décodage de cette trame est engagé, soit son signal s'ajoute au bruit ambiant. Il s'agit donc de déterminer l'impact de la transmission de chaque trame au niveau des nœuds. Pour simplifier les calculs, nous avons considéré que le déplacement des nœuds dans l'espace est négligeable pendant le temps nécessaire à la transmission d'une trame. Ainsi, un seul calcul de puissance en chaque nœud différent de l'émetteur est nécessaire pour chaque trame transmise.

À chaque instant, l'impact de la transmission d'un nouveau signal en un nœud autre que l'émetteur dépend de l'état dans lequel se trouve ce nœud :

- *en transmission* : le nœud est en cours de transmission d'une trame, le nouveau signal s'ajoute invariablement au bruit ambiant ;
- *en réception* : le nœud est déjà engagé dans la tentative de décodage d'une trame, le nouveau signal s'ajoute invariablement au bruit ambiant et le décodage échoue si le rapport signal/bruit passe sous le seuil de décodage SNR_{RX} ;
- *occupé* : le niveau de bruit est déjà supérieur au seuil de détection de porteuse et le nœud considère le médium comme occupé ; le nœud tente de décoder cette nouvelle trame seulement si la puissance du signal est supérieure au seuil de réception et le rapport signal/bruit supérieur à SNR_{RX}, sinon le signal se rajoute au bruit ambiant ;
- *libre* : le niveau de bruit est inférieur au seuil de détection de porteuse et le nœud considère le médium comme libre ; il tente de décoder cette nouvelle trame seulement si la puissance du signal est supérieure au seuil de réception et le rapport signal/bruit supérieur à SNR_{RX}, sinon le signal se rajoute au bruit ambiant et si le bruit est supérieur au seuil de détection de porteuse, le nœud change d'état.

À la fin de la transmission de la trame, en tous les nœuds pour lesquels le signal était considéré comme du bruit, la puissance du signal est ôtée de la puissance du bruit ambiant, qui est réévaluée par rapport au seuil de détection de porteuse. Pour tous les nœuds qui avaient entamé

un décodage, si celui-ci n'est pas marqué comme échoué, alors la trame a été décodée avec succès, sinon le décodage a échoué.

4.1.3 Couche MAC

La couche MAC utilisée simule le fonctionnement du standard IEEE 802.11. Les paramètres de temps sont ceux correspondant à DS PHY. Nous nous sommes permis toutefois quelques approximations qui n'ont pas ou très peu d'incidence sur les simulations : une trame est transmise en totalité au même débit, l'en-tête de trame est considéré comme décodé instantanément au début du décodage (et donc les valeurs de destinataire et de taille totale sont connus à cet instant), la fragmentation et le chiffrement WEP ne sont pas utilisés.

4.1.4 Couche OLSR

Le composant en charge de la couche OLSR reçoit des paquets de données de la part de la couche applicative et échange des paquets la couche MAC. Le protocole OLSR est enrichi des messages et algorithmes du chapitre 3.

4.1.5 Couche applicative

Le composant en charge de la couche applicative génère des paquets de données qu'il transmet à la couche OLSR pour routage. En principe, les paquets de données reçus par la destination devraient remonter à la couche applicative, mais ici seule leur comptabilisation est nécessaire.

Ce composant peut fonctionner selon deux modes : orienté paquet et orienté flux. Dans le mode orienté paquet, des paquets de données sont générés à des intervalles de temps tirés aléatoirement selon une loi exponentielle de paramètre λ_p, leurs nœuds source et destination sont tirés uniformément et leur taille est tirée uniformément sur un intervalle $[s_{min};s_{max}]$. En revanche, dans le mode orienté flux, ce sont des flux réguliers de paquets (dont les interarrivées suivent une loi exponentielle) de taille constante (tirée uniformément sur $[s_{min};s_{max}]$ une fois pour tout le flux) et de nœuds source et destination constants (tirés uniformément une fois pour tout le flux) qui sont générés à des instants tirés aléatoirement (suivant une loi exponentielle), afin de mieux modéliser des flux de données orientés connexion, comme des transferts de fichiers.

4.2 Paramètres du modèle

Certains paramètres du modèle sont fixés pour toutes les simulations car leur influence sur la solution ne présente pas d'intérêt particulier. Les autres paramètres sont ceux dont l'impact sur l'efficacité de la solution nous intéresse.

4.2.1 Couche applicative

La taille des paquets de données est simplement tirée des cas concrets en TCP/IP, avec $s_{min} = 40$ octets et $s_{max} = 1\ 500$ octets. Quant au paramètre λ_p de la loi exponentielle des intervalles de temps entre deux générations successives de paquets, sa valeur est déterminante. En effet, il apparaît qu'une valeur trop grande de λ_p peut saturer le réseau de paquets de données, puisqu'aucun mécanisme de contrôle de flux n'est mis en place. Il est donc essentiel de régler ce paramètre à une valeur à la fois suffisamment grande pour que les statistiques sur les paquets de données soient pertinentes et suffisamment petite pour que le réseau ne sature pas.

Nous avons empiriquement déterminé qu'une valeur de $\lambda_p = 20$ est satisfaisante.

4.2.2 Couche OLSR

Les paramètres par défaut préconisés dans la spécification du protocole OLSR ont été utilisés. Leurs valeurs sont résumées dans le tableau 4.1.

Quant aux paramètres de notre solution (voir le paragraphe 3.2 du chapitre 3), leurs valeurs sont présentées dans le tableau 4.2. En outre, les valeurs des degrés locaux ne sont pas diffusées telles quelles dans les messages TC, mais plutôt le résultat de l'application de la fonction suivante :

$$D_j^i \mapsto \begin{cases} 0 & \text{si } D_j^i < 1 \\ 1 + \ln D_j^i & \text{sinon} \end{cases} . \tag{4.4}$$

Par conséquent, la fonction f utilisée pour le calcul des poids sur les arcs du graphe topologique est simplement l'identité. De cette façon, le résultat est pratiquement le même mais l'utilisation de plus petites valeurs de degré de suspicion local dans les messages TC permet leur codage simple sur un nombre de bits limités.

TAB. 4.1: Paramètres de la couche OLSR

HELLO_INTERVAL	2 s	DUP_HOLD_TIME	30 s
REFRESH_INTERVAL	2 s	MAXJITTER	0,5 s
TC_INTERVAL	5 s	MPR_COVERAGE	1
NEIGHB_HOLD_TIME	6 s	TC_REDUNDANCY	0
TOP_HOLD_TIME	15 s	Willingness	WILL_DEFAULT

TAB. 4.2: Paramètres de la solution

P	2 s	r'_I	1
r_I	1	r'_D	1
r_D	10^{-4}	r_A	10^6

4.2.3 Couche MAC

Les paramètres de la couche MAC sont ceux préconisés par le standard IEEE 802.11 et ses amendements. Les valeurs des différentes durées paramétrables sont résumées dans le tableau 4.3.

4.2.4 Couche physique

Les nœuds émettent sur la bande des 2,5 GHz, soit un paramètre $\lambda = 1{,}2 \cdot 10^{-2}$ m. Ils émettent avec une puissance $P_t = 50$ mW, ce qui implique, avec le modèle d'atténuation de (4.3), un rayon de portée de réception d'au mieux environ 214 m et un rayon de portée de détection de porteuse d'environ 380 m.

Les paramètres de mobilité des nœuds sont $x_{max} = 1\,000$ m, $y_{max} = 1\,000$ m, $\lambda_m = 1/600$, $v_{min} = 0$ m/s et $v_{max} = 1$ m/s.

4.2.5 Paramètres variables

Le nombre total de nœuds détermine la densité du réseau, puisque la surface reste la même pour toutes les simulations. Nous avons donc lancé des simulations pour un nombre total de nœuds variant entre 30 et 100.

Les nœuds malveillants, appelés ici *tricheurs*, sont paramétrés par leur nombre total, leur

TAB. 4.3: Paramètres de la couche MAC

Seuil RTS/CTS	50 octets	Durée d'un Slot	$2 \cdot 10^{-5}$ s
Tentatives de backoff max.	7	Durée d'un SIFS	10^{-5} s
Fenêtre de contention min.	31 slots	Durée d'un DIFS	$5 \cdot 10^{-5}$ s
Fenêtre de contention max.	1 023 slots	Durée d'un EIFS	$1{,}7 \cdot 10^{-3}$ s
Débit broadcast	11 Mbps	Seuil de détection de porteuse	-75 dBm
Débit unicast	54 Mbps	Seuil de réception	-70 dBm
SNR_{RX}	3 dB		

probabilité de ne pas retransmettre un paquet de données et leur stratégie de gestion des compteurs dans le cas où le comptage des paquets est mis en œuvre. Lorsqu'un tricheur décide de ne pas détruire un paquet et de le retransmettre correctement, il gère ses compteurs de la même manière qu'un nœud bienveillant. En revanche, lorsqu'il détruit un paquet, des comportements différents sont possibles. En effet, on doit s'attendre à ce qu'un tricheur tâche de masquer d'une manière ou d'une autre ses actions néfastes pour tenter de déjouer le système de détection.

Comportement des tricheurs

Nous avons défini plusieurs stratégies possibles de gestion des compteurs par un nœud malveillant suite à la destruction d'un paquet. Elles sont décrite dans le tableau 4.4. Par ailleurs, dans les scénarios qui mettent en œuvre notre méthode de notation, les tricheurs publient des degrés de suspicion à l'égard de leurs MPR-selectors de 1 000 avec des indices de confiance de 1, ce qui permet d'évaluer à quel point notre méthode est résistante à la diffamation.

Le moyen principal que nous avons choisi pour évaluer les performances de notre méthode est de mesurer le taux d'acheminement des paquets de données de leur source jusqu'à leur destination, tous paquets ou flux de données confondus. Nous pouvons ainsi prendre en compte tous les facteurs de perte de paquets pouvant apparaître dans un réseau ad hoc mobile (manque de route, expiration du TTL, perte au niveau des couches inférieures, pertes intentionnelles). Comme l'illustre par ailleurs la figure 4.4, même en l'absence de tricheurs, le taux d'acheminement est en général inférieur à 1. Pour apprécier les taux d'acheminement atteints avec notre solution, nous proposons de comparer ceux-ci à ceux de plusieurs simulations-témoins.

TAB. 4.4: Stratégies de triche : manipulation des compteurs en cas de destruction de paquet.

FD	Incrémenter les compteurs comme si le paquet avait été effectivement retransmis.
BD	N'incrémenter aucun compteur, comme si le paquet n'avait jamais été reçu depuis le nœud précédent.
SD	Incrémenter les compteurs sur tous les liens de telle façon que la somme rajoutée totale soit égale à la taille du paquet détruit.
ND	Ne pas incrémenter les compteurs sur le lien sortant.
AD	Appliquer **FD**, **BD**, **SD** ou **ND** aléatoirement.

Simulations-témoins

Tout d'abord, il est essentiel de comparer le taux d'acheminement d'une simulation à celui atteint par l'utilisation du protocole OLSR classique, à conditions de vitesse maximale et densité des nœuds égales, en l'absence de tricheurs. C'est le témoin qui fixe la borne supérieure absolue du taux d'acheminement dans ces conditions.

Il est ensuite intéressant de comparer ce taux d'acheminement à celui atteint par l'utilisation du protocole OLSR classique en présence des mêmes tricheurs, à conditions de vitesse maximale et densité des nœuds égales. En un sens, cela donne une idée de la borne inférieure du taux d'acheminement qu'on ne devrait absolument jamais dépasser, autrement l'application de la solution est clairement à proscrire.

Enfin, comme le taux d'acheminement dépend aussi de la topologie du réseau, celle-ci ne permettant parfois pas de trouver des chemins qui évitent les tricheurs, on veut pouvoir le comparer à celui atteint par l'utilisation du protocole OLSR classique auquel on rajoute un oracle qui permet de savoir si un nœud est tricheur. La connaissance des tricheurs par l'oracle permet de modifier les poids sur le graphe topologique afin de trouver des chemin qui contournent les tricheurs quand cela est possible. On peut considérer la performance de ce témoin comme un objectif dont l'application de notre solution cherche à se rapprocher au maximum. En effet, si la solution permettait de détecter parfaitement les tricheurs et seulement eux, ceci en ne gaspillant jamais un paquet de données, alors c'est le taux d'acheminement qu'elle permettrait d'atteindre.

Le tableau 4.5 résume les paramètres des simulations-témoins.

4.3 Considérations générales

Pour extraire par simulation des données permettant d'avoir une vision générale de ce qui se passe, c'est à dire observer l'influence de certains paramètres en moyenne, tous autres paramètres confondus, on doit nécessairement éviter ou minimiser l'impact des cas particuliers. Si par exemple on désire observer le comportement d'une variable en moyenne selon un certain paramètre et que cette variable dépend fortement de la topologie instantanée, il faut s'assurer que

TAB. 4.5: Simulations-témoins

	Présence de tricheurs	**Routage**
NC	non	OLSR classique
C	oui	OLSR classique
K	oui	OLSR muni d'un oracle

la topologie aura suffisamment varié durant la simulation. Autrement dit, il faut s'assurer que les résultats obtenus sont stables quelle que soit la graine du générateur de nombres aléatoires. Or pour que la topologie d'un réseau ad hoc mobile ait beaucoup varié durant la simulation, il faut que les nœuds se soient eux-mêmes beaucoup déplacé. On doit donc avoir des nœuds se déplaçant à une vitesse suffisamment importante ou, à défaut, des durées simulées suffisamment longues. Seulement comme nous allons le voir dans le paragraphe suivant, une vitesse importante des nœuds est indésirable.

4.3.1 Vitesse maximale des nœuds

Nous avons constaté que la vitesse maximale des nœuds est un facteur très important du taux d'acheminement des paquets de données dans un réseau ad hoc dense fonctionnant avec le protocole OLSR. Il est apparu que dans un réseau dense, plus la vitesse maximale des nœuds est grande, plus les pertes de données sont importantes.

En fait ce phénomène est dû à la façon dont les nœuds sélectionnent leurs MPRs, selon l'heuristique décrite dans la spécification [3], paragraphe 8.3.1 :

```
The proposed heuristic is as follows:

  1    Start with an MPR set made of all members of N with
       N_willingness equal to WILL_ALWAYS

  2    Calculate D(y), where y is a member of N, for all nodes in N.

  3    Add to the MPR set those nodes in N, which are the *only*
       nodes to provide reachability to a node in N2.  For example,
       if node b in N2 can be reached only through a symmetric link
       to node a in N, then add node a to the MPR set.  Remove the
       nodes from N2 which are now covered by a node in the MPR set.

  4    While there exist nodes in N2 which are not covered by at
       least one node in the MPR set:

       4.1  For each node in N, calculate the reachability, i.e., the
            number of nodes in N2 which are not yet covered by at
            least one node in the MPR set, and which are reachable
            through this 1-hop neighbor;

       4.2  Select as a MPR the node with highest N_willingness among
            the nodes in N with non-zero reachability.  In case of
            multiple choice select the node which provides
            reachability to the maximum number of nodes in N2.  In
            case of multiple nodes providing the same amount of
            reachability, select the node as MPR whose D(y) is
            greater.  Remove the nodes from N2 which are now covered
```

```
            by a node in the MPR set.

5     A node's MPR set is generated from the union of the MPR sets
      for each interface.  As an optimization, process each node, y,
      in the MPR set in increasing order of N_willingness.  If all
      nodes in N2 are still covered by at least one node in the MPR
      set excluding node y, and if N_willingness of node y is
      smaller than WILL_ALWAYS, then node y MAY be removed from the
      MPR set.
```

Si on considère un réseau dense homogène sur une surface plane et que tous les nœuds ont le même rayon R de portée de réception, alors les nœuds ont tendance à sélectionner comme MPRs des voisins qui sont à distance proche de R. En effet, même si l'information sur la qualité (ou la puissance) du signal reçu est prise en compte pour préférer des MPRs plus proches, les étapes 3 et 5 impliquent la sélection de voisins parmi les plus éloignés. L'étape 3 stipule que les voisins étant les seuls à pouvoir joindre certains voisins à deux sauts doivent être sélectionnés en priorité, or dans un réseau dense, les voisins à deux sauts joignables par un seul voisin à un saut sont à distance proche de $2 \cdot R$. Tandis qu'à l'étape 5, il est suggéré qu'une optimisation de l'ensemble sélectionné doit être effectuée afin d'éliminer les MPRs redondants avec d'autres. Alors les MPRs les plus éloignés sont bien moins susceptibles d'être éliminés par cette optimisation. Sachant que le temps maximal de détection de la perte d'un voisin ne dépend que des temps de validité des informations et de la fréquence d'émission des messages HELLO, la probabilité qu'un voisin soit perdu augmente avec la vitesse des nœuds et la distance du voisin au moment de sa dernière détection. Si on a tendance à sélectionner des MPRs qui sont parmi les voisins les plus éloignés, alors l'augmentation de la vitesse fait augmenter considérablement la probabilité qu'un instant plus tard ce MPR sera perdu. Par conséquent il est clair que l'augmentation de la vitesse a de fortes conséquences sur la probabilité de perte des paquets de données.

Pour illustrer ce phénomène, nous avons effectué une série de simulations en faisant varier le nombre de nœuds sur la surface et leur vitesse maximale. Les résultats sont présentés sur les figures 4.3 et 4.4.

Chaque sommet de la grille correspond à la moyenne sur cinq simulations aux graines différentes. On constate bien que plus la vitesse augmente, plus le taux de transmission d'un nœud à un voisin diminue. Par ailleurs, on remarque une légère augmentation du taux de transmission réussie lorsque la densité diminue. Le taux d'acheminement quant à lui amplifie considérablement ces tendances puisque la probabilité d'acheminement d'un paquet est le produit des probabilités de transmission sur les liens qui composent son chemin. Ces derniers sont de plus en plus courts dans les réseaux peu denses du fait du partitionnement du réseau en composantes disjointes (les paquets à destination de nœuds dont la source ne dispose pas de route ne sont pas

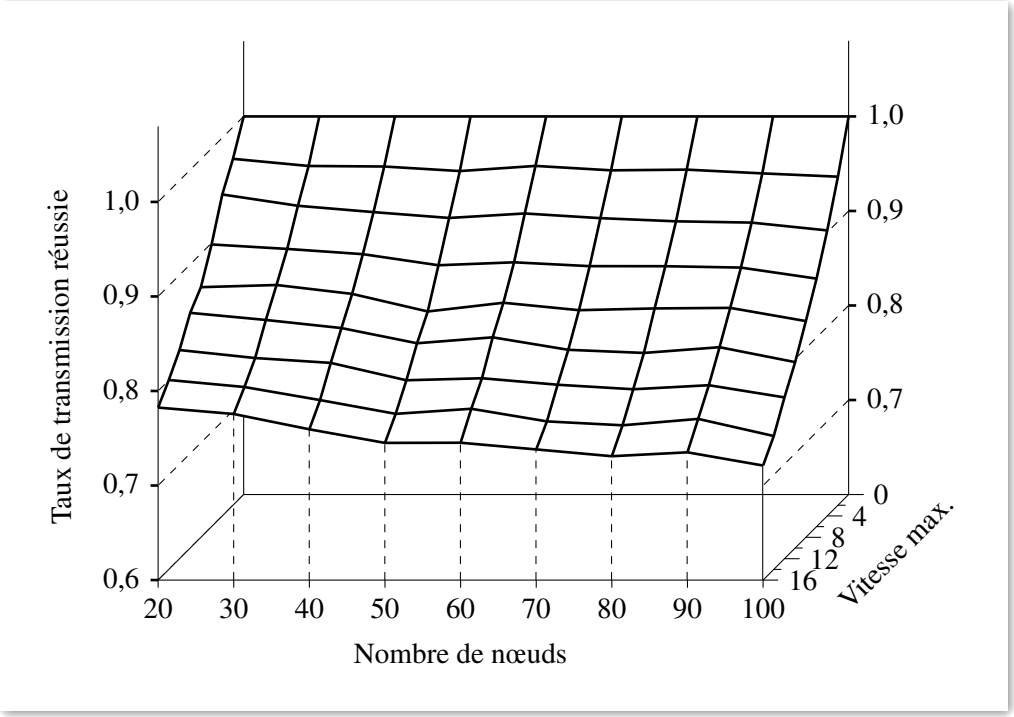


FIG. 4.3: Taux de transmission réussie dans un réseau en fonction de la densité de et la vitesse des nœuds, avec OLSR classique, 100 nœuds et aucun tricheur

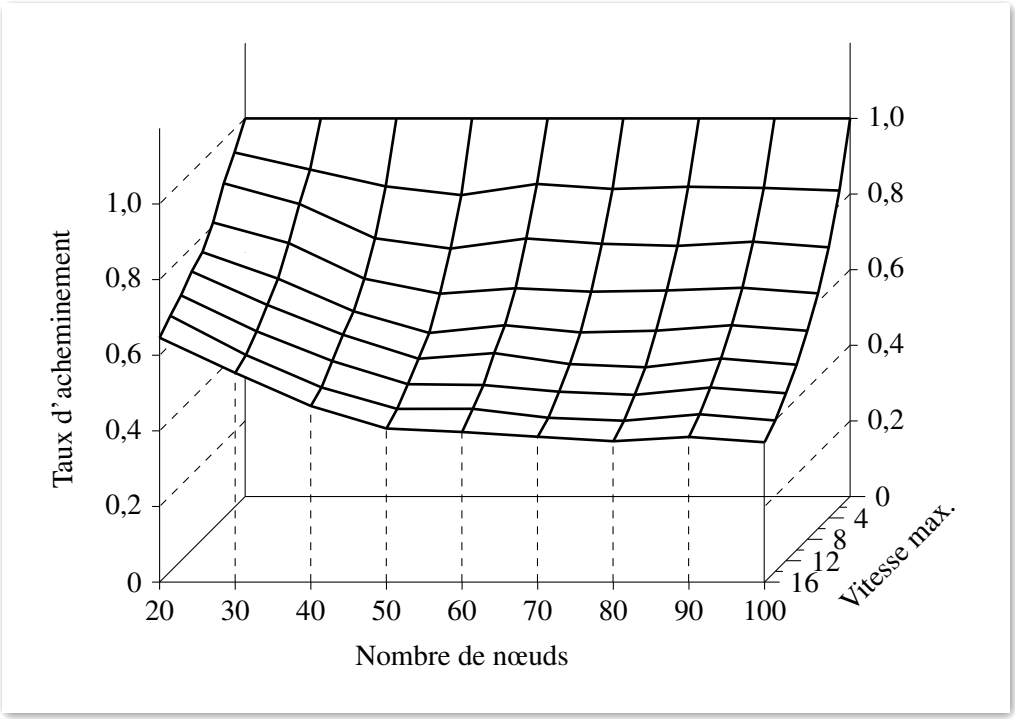


FIG. 4.4: Taux d'acheminement dans un réseau en fonction de la densité de et la vitesse des nœuds, avec OLSR classique, 100 nœuds et aucun tricheur

comptabilisés).

Pour éviter de devoir lancer des simulations sur des temps ridiculement grands juste pour que les nœuds aient le temps de faire suffisamment varier la topologie, nous nous sommes résolus à lancer plusieurs simulations plus courtes aux graines différentes et à considérer la moyenne des résultats. Cette approche est autant valide, puisqu'alors la variation de cette moyenne des résultats est nécessairement faible quand on change les graines des simulations (sans en changer le nombre).

Par conséquent, pour certaines séries de simulations, cinq graines différentes ont suffit, alors que pour d'autres, nous avons dû lancer jusqu'à 15 graines différentes.

4.3.2 États transitoires

Les nœuds participant à un réseau ad hoc utilisant un protocole de routage proactif sont en général dans un état transitoire suivant leur apparition. Cette phase de mise en place pendant laquelle le nœud acquiert les informations sur la topologie peut durer un certain temps, selon le protocole utilisé. Pendant cette phase, il n'est pas certain d'une part qu'une route vers une destination soit disponible (bien que la destination soit physiquement atteignable par le réseau) et d'autre part les messages de contrôle ne comportent pas encore toutes les informations sur le voisinage. Par conséquent, les mesures concernant les paquets de données et les messages de contrôle ne devraient pas être effectuées durant cette période.

Dans le cas d'OLSR, il est prudent de supposer qu'au-delà des trente premières secondes de fonctionnement, les nœuds ont acquis les informations de topologie et que la phase transitoire de démarrage est passée. Dans toutes nos simulations, les mesures influencées par la taille (et la fréquence) des messages de contrôle ne portent pas sur les trente premières secondes de l'expérience. En ce qui concerne les paquets de données, ils ne sont simplement pas générés pendant cette phase initiale.

Lorsque la simulation se termine, tous les paquets en cours de transmission sont perdus. Ici encore, pour ne pas biaiser les mesures sur les paquets de données, ces derniers ne sont plus générés pendant les trente dernières secondes de l'expérience, afin de minimiser les chances pour que l'un d'eux ne soit pas encore arrivé à destination (ou ne soit pas détruit pour toute autre raison) au moment de la fin de la simulation.

4.4 Stabilité des résultats

Pour déterminer la durée nécessaire pour que la solution atteigne un niveau stationnaire, nous avons effectué une série de simulations d'une durée d'une heure. Nous nous sommes in-

téressés non pas au taux d'acheminement moyen global (rapport entre le nombre de paquets de données émis par les sources et le nombre de paquets reçus par les destinations, depuis le début de l'expérience), ni à son évolution au cours du temps, mais au taux d'acheminement moyen quasi-instantané. Nous avons dont généré des traces de tous les instants où un paquet arrive à destination ou est perdu et calculé une moyenne par morceaux de 30 secondes. La figure 4.5 présente les courbes obtenues pour les simulations-témoins.

On constate qu'à cette densité, le témoin **K** est pratiquement indiscernable de **NC**, ce qui montre bien que le contournement des tricheurs est efficace. Il apparaît que pour **NC**, comme évoqué au paragraphe 4.2.4, l'écrasante majorité des pertes de paquets de données est due à des nœuds destinataires de la trame unicast trop éloignés. Par conséquent, que **K** puisse localement dépasser **NC** par moments n'est pas surprenant, dans la mesure où parfois le fait d'éviter des tricheurs en empruntant un chemin possiblement plus long permet de solliciter des nœuds intermédiaires plus proches.

L'instabilité apparente de **C**, par rapport à **NC** et **K** est due au fait que l'action des tricheurs dépend des aléas de la topologie dynamique : un tricheur n'est pas toujours bien placé pour être compris dans un chemin.

Pour apprécier la stabilité des différentes simulations de notre méthode, il n'est plus nécessaire de comparer les courbes à **NC**, mais seulement à **K** et **C**. Les courbes de la figure 4.6 présentent les résultats obtenus pour les différentes stratégies des tricheurs telles que décrites dans le tableau 4.4.

Ces résultats illustrent bien comment la solution proposée permet d'atteindre un état stable en moins de 300 secondes. On remarque aussi que lorsque la stratégie **ND** est appliquée, la stabilité est atteinte encore plus rapidement, ce qui est dû au fait qu'alors les degrés de suspicion des tricheurs augmentent bien plus vite que ceux des autres.

Pour une vue plus générale des taux d'acheminements atteints par les simulations-témoins, la figure 4.7 présente la moyenne sur cinq simulations (aux graines différentes) de l'évolution, au cours des 15 premières minutes, du taux d'acheminement global. Les courbes analogues pour les différentes stratégies des tricheurs sont présentées sur la figure 4.8.

À cette densité de nœuds, on peut noter qu'au bout de 15 minutes, les résultats sont suffisamment stables pour être relevés. Les courbes de la figure 4.8 permettent de mieux voir comment le fait que les tricheurs adoptent la stratégie **ND** au moins de temps en temps accélère la convergence de notre méthode et, dans le cas de la courbe **ND** seule, les résultats sont très proches de **K**.

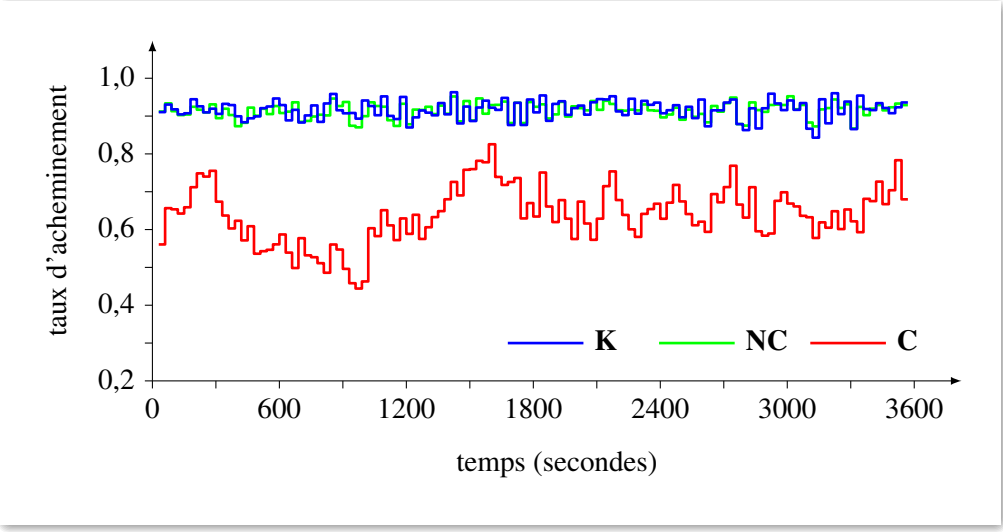


FIG. 4.5: Taux d'acheminement instantané des simulations-témoins

4.5 Influence de la densité

L'efficacité d'une méthode, comme la nôtre, basée sur l'évitement et non l'exclusion des nœuds suspects est fortement dépendante de l'existence de chemins alternatifs. Pour que cette dernière soit assurée, il semble essentiel que le réseau soit suffisamment dense. Ainsi il est intéressant d'observer l'évolution du taux d'acheminement des différents témoins et simulations selon la densité du réseau. Pour varier la densité du réseau, nous avons simplement varié le nombre de nœuds total, tout en maintenant une proportion égale (à 10%) de tricheurs. Cette approche simple pose toutefois un problème par rapport à la connexité du réseau qui a d'autant plus de chances de ne pas être assurée que le nombre total de nœuds est petit. Pour que les taux d'acheminement ne soient pas exagérément faibles du fait que la probabilité de tirer une source et une destination (dans le modèle de la couche applicative) qui appartiennent à des composantes disjointes, nous ne comptabilisons pas les paquets dont la source ne dispose pas de route vers la destination. Nous comptons cependant comme perdus les paquets pour lesquels un nœud intermédiaire ne dispose pas de route. Les figures 4.9 et 4.10 présentent les résultats pour un nombre total de nœuds variant entre 20 et 100, en moyenne sur 15 graines différentes. En effet, à faible densité, l'impact de la topologie s'est avéré très important et a nécessité plus de simulations pour obtenir des résultats suffisamment stables. Ici les courbes **FD**, **BD**, **SD**, **ND** et **AD** ne sont pas distinguées et sont désignées par ***D**.

On constate que les courbes ***D** sont très proches les unes des autres et qu'on pourrait aussi bien considérer leur moyenne sans véritablement réduire la lisibilité des résultats. À 20 nœuds,

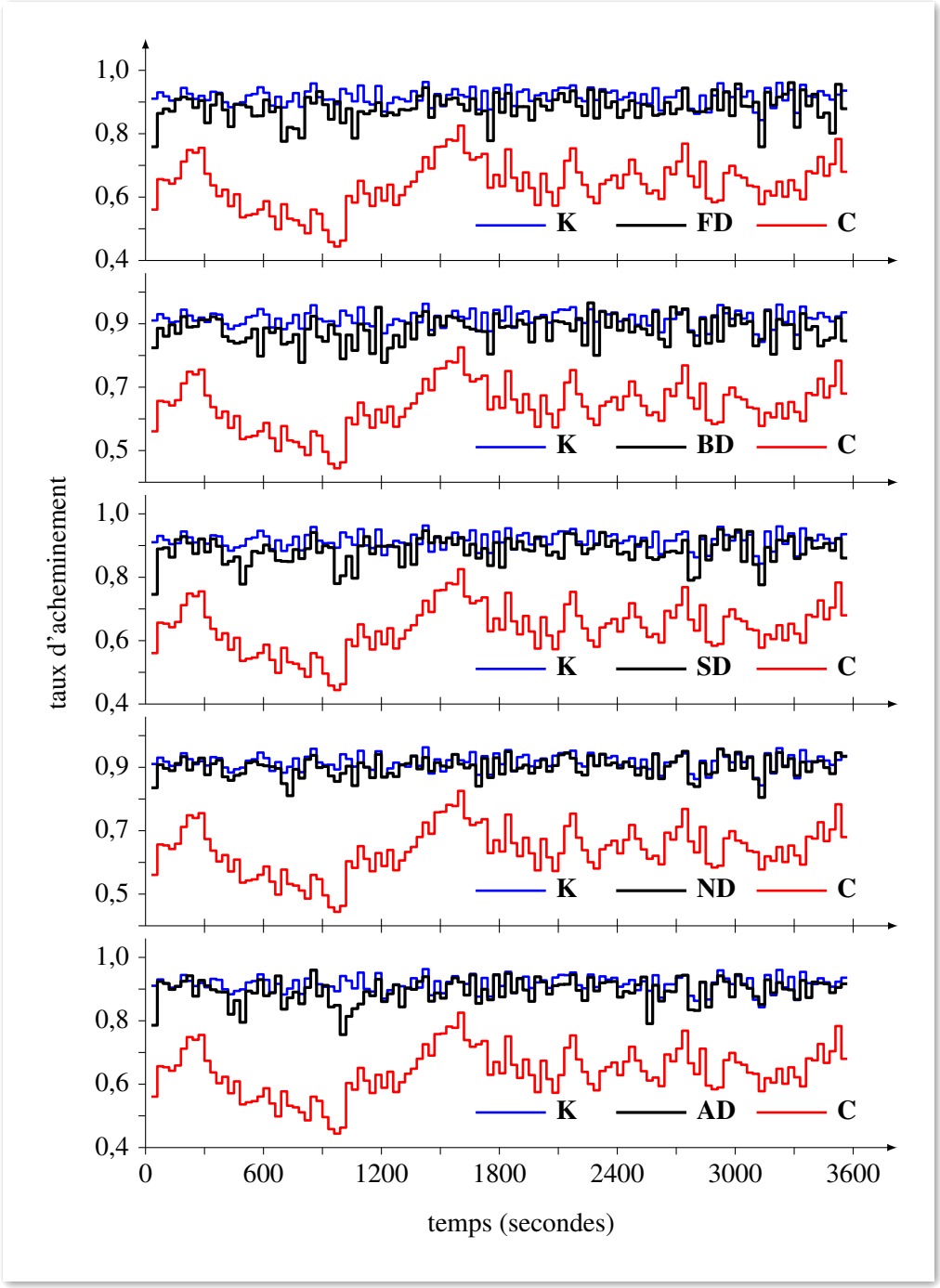


FIG. 4.6: Taux d'acheminement instantanés des différentes stratégies de triche

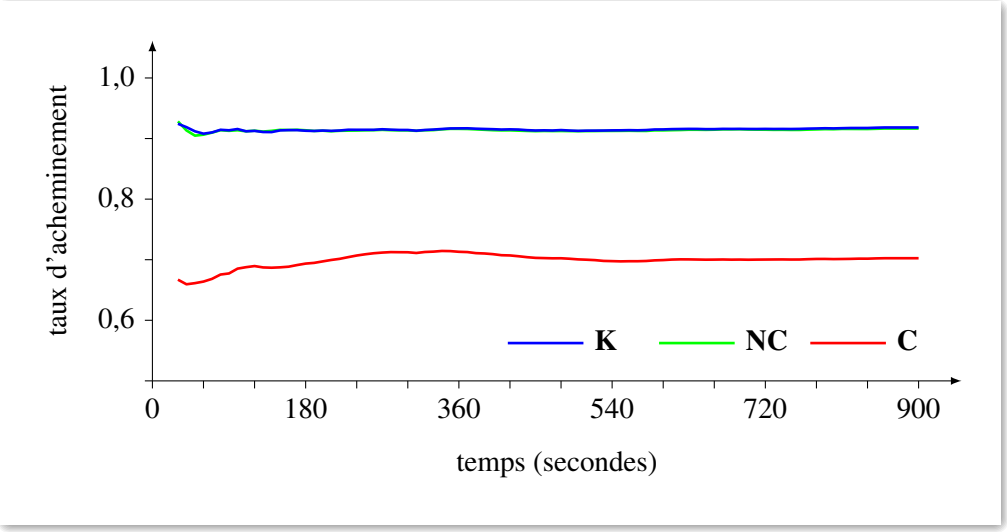


FIG. 4.7: Taux d'acheminement global des simulations-témoins

les courbes **K** et ***D** sont nettement superposées à **C**, ce qui indique que notre méthode, tout comme celle d'OLSR muni d'un oracle, n'apporte aucun avantage par rapport à OLSR classique. Ce phénomène s'explique simplement par la rareté des chemins alternatifs permettant de contourner un tricheur. On remarque aussi que l'impact des tricheurs semble alors globalement faible, avec des taux d'acheminement dépassant 90%, mais ceci est juste l'effet des chemins plutôt courts que parcourent les paquets de données. En effet, à une densité aussi faible, on peut s'attendre à l'apparition de plusieurs ensembles de nœuds disjoints dans le réseau. Plus la densité est faible, plus le nombre de ces ensembles disjoints peut être grand. Or un paquet de données reste confiné à l'ensemble connexe auquel appartient sa source et plus il y a d'ensembles disjoints, plus la probabilité que le chemin parcouru passe par un tricheur est faible.

C'est à 30 nœuds qu'on commence à disposer de chemins alternatifs. En effet, la courbe **K** est supérieure de 5 points environ à la courbe **C**. Ici on peut voir que ***D** est plus proche de **K** que de **C**, ce qui montre que l'avantage de la méthode commence à se faire sentir.

À 40 et 50 nœuds, l'augmentation du nombre de chemins alternatifs se confirme, puisque **K** semble se détacher de **C** et se rapprocher progressivement de **NC**. Les courbes ***D** suivent un peu moins bien, mais restent dans la limite des 5 points en dessous de **K**.

À partir de 60 nœuds, les courbes se stabilisent davantage. Le nombre de chemins alternatifs est tel que la courbe **K** se rapproche de plus en plus de **NC** au point de se superposer à cette dernière à partir de 90 nœuds. Les courbes ***D** deviennent stables de plus en plus rapidement et restent assez proches à la fois les unes des autres (leur écart ne dépasse jamais 5 points) et de **K**.

La figure 4.11 résume ces résultats en laissant apparaître la tendance générale de la méthode

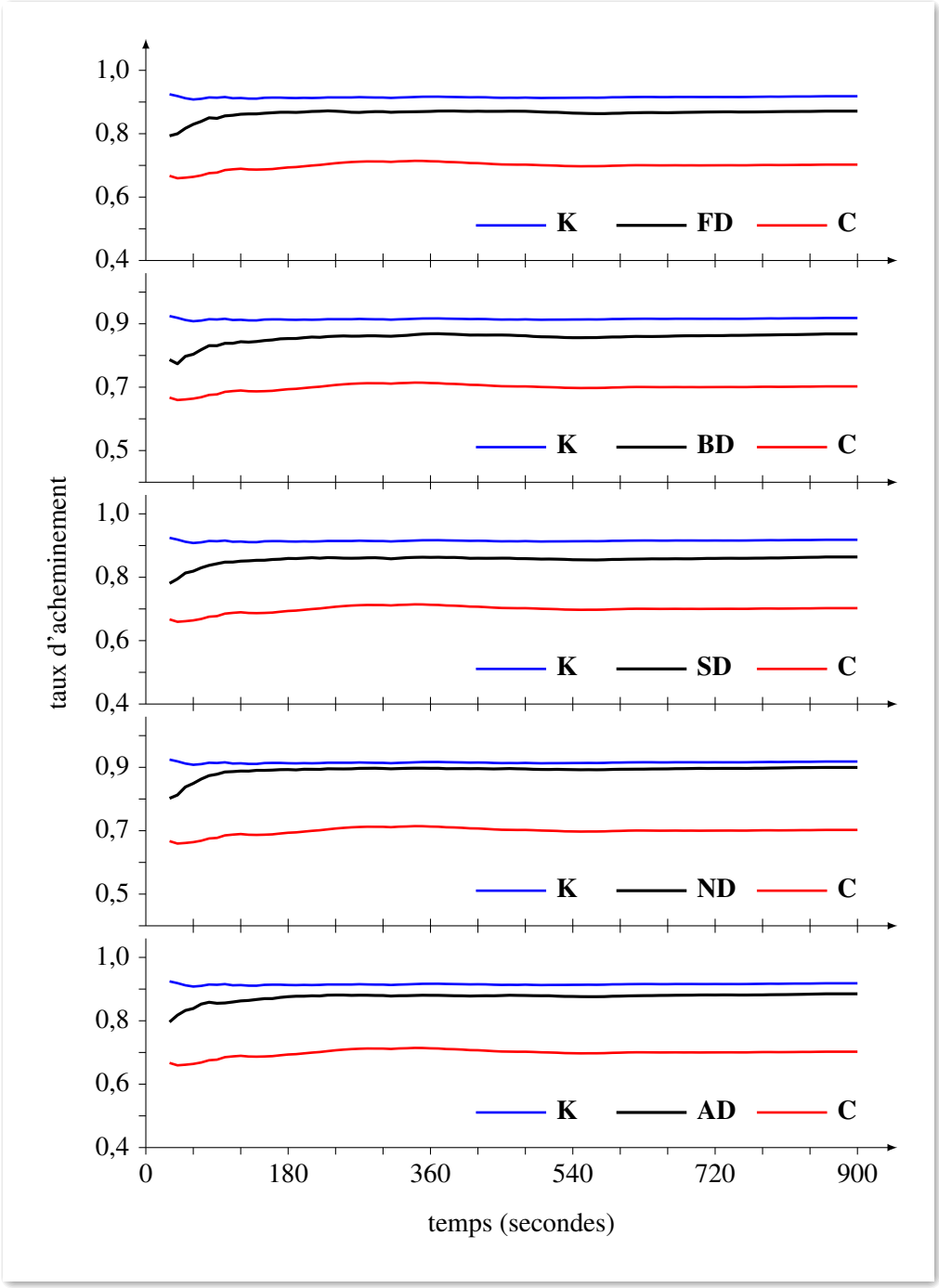


FIG. 4.8: Taux d'acheminement globaux des différentes stratégies de triche

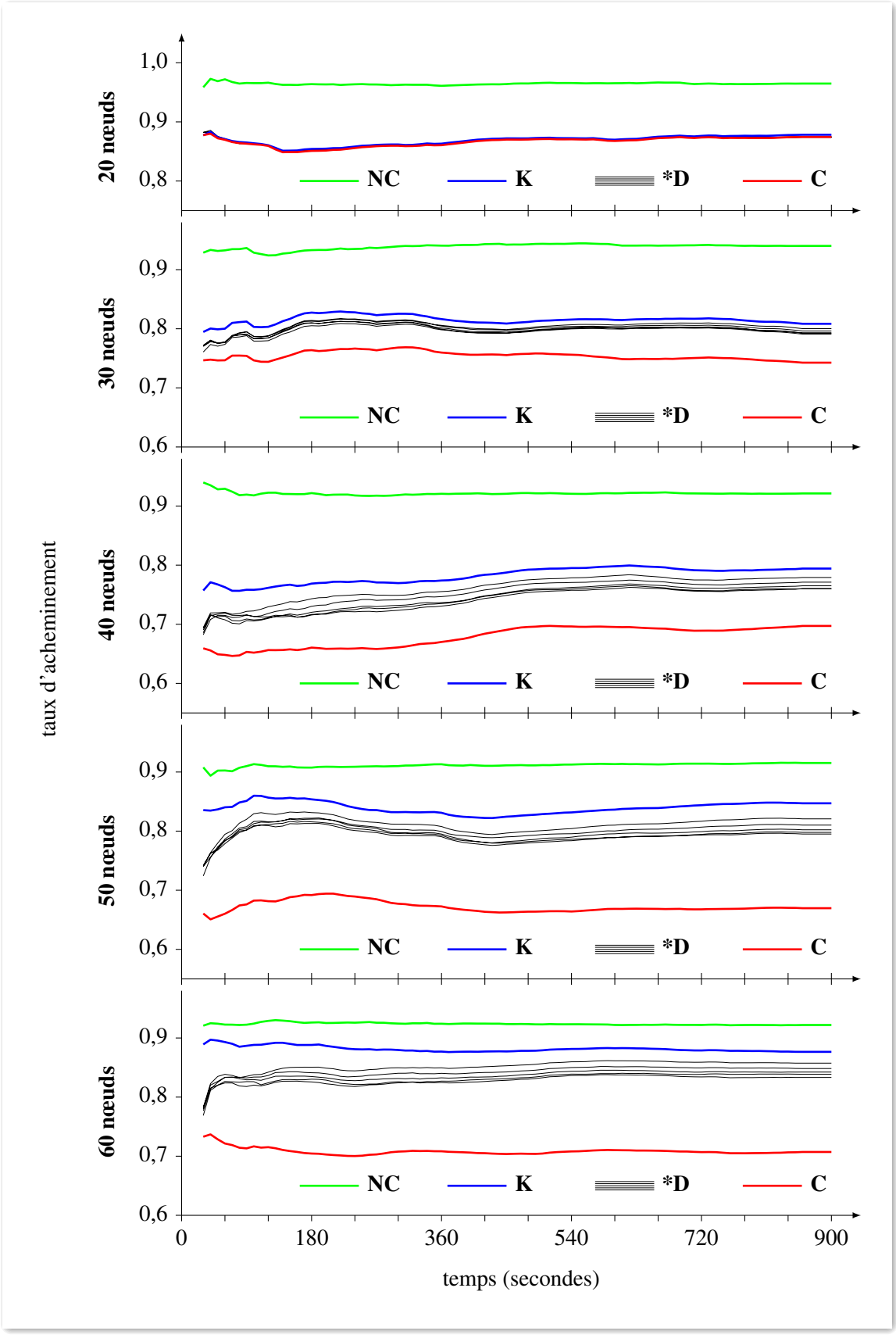


FIG. 4.9: Taux d'acheminement global, 20 à 60 nœuds dont 10% de tricheurs

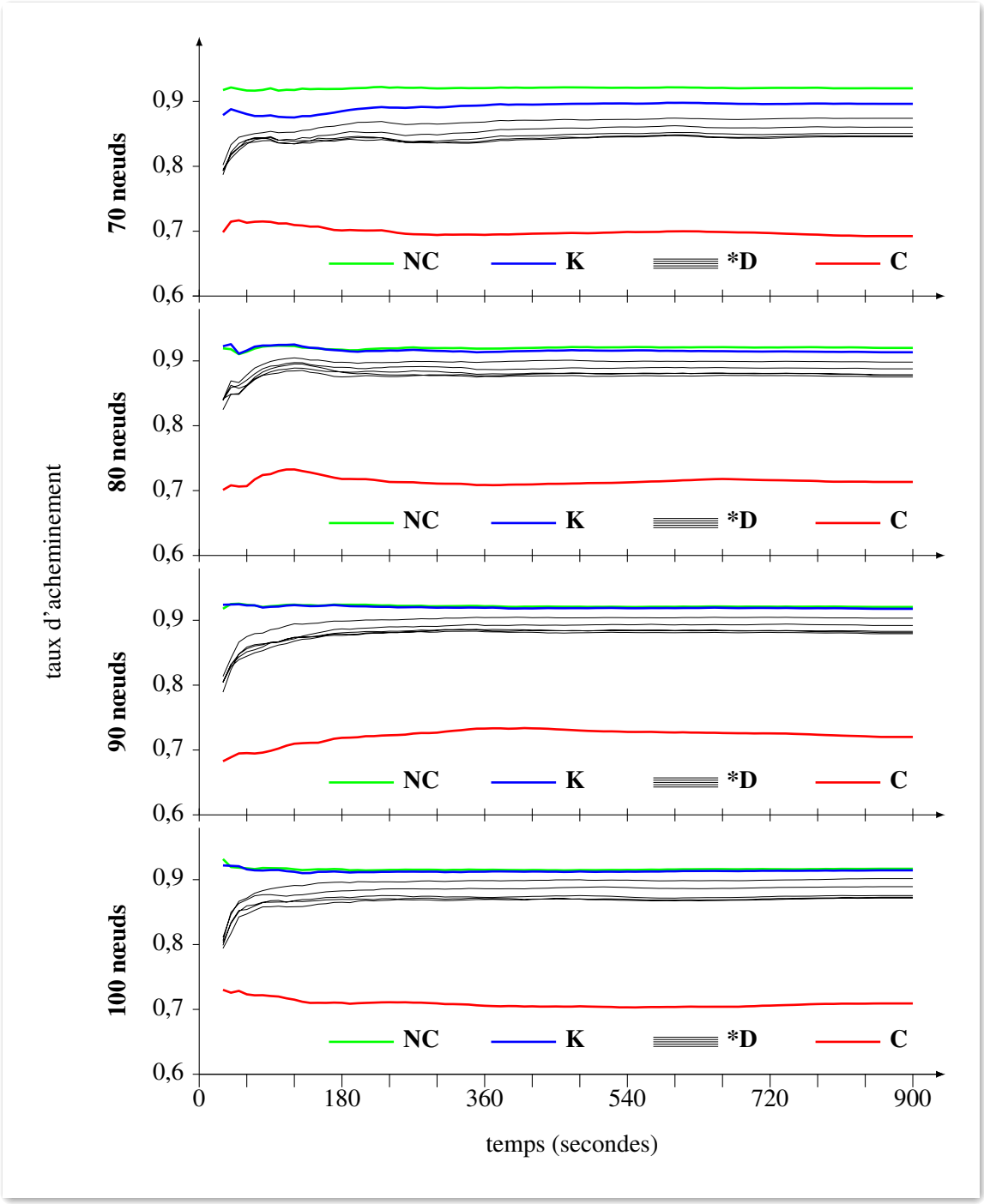


FIG. 4.10: Taux d'acheminement global, 70 à 100 nœuds dont 10% de tricheurs

par rapport aux simulations-témoins en fonction de la densité des nœuds. Les intervalles de confiance à 95% sont indiqués en chaque point. On remarque que tant que la densité est faible, la tendance générale du taux d'acheminement est à la baisse avec l'augmentation du nombre de nœuds. Lorsque la densité augmente (à partir de 40 nœuds), la diversité croissante des chemins inverse cette tendance et le taux d'acheminement remonte rapidement. À partir de 50 nœuds, le taux d'acheminement est plus proche de **NC** que de **C**.

4.6 Influence du taux de perte intentionnelle

Il est aussi intéressant de voir l'évolution des performances de la méthode par rapport au taux de perte intentionnelle des tricheurs. Les résultats avec 10% de tricheurs sont présentés sur les figures 4.12, 4.13 et 4.14 pour respectivement 100, 60 et 40 nœuds. Les intervalles de confiance à 95% sont indiqués en chaque point. Dans les trois cas, la courbe **C** est nettement affectée par le taux de perte et ses performances décroissent proportionnellement à celui-ci. La courbe **K**, quant à elle, se rapproche de **NC** plus la densité est grande. Il apparaît clairement que notre méthode ne requiert pas un fort taux de perte pour fonctionner convenablement et ses performances restent proches de celles du témoin **K**. Le fait que la courbe ***D** s'éloigne quelque peu de **K** avec l'augmentation du taux de perte semble indiquer simplement que la méthode doit tester en quelque sorte de temps en temps les tricheurs, afin que leur degré de suspicion demeure élevé, contrairement à l'utilisation de l'oracle qui se fait sans coût.

4.7 Surcoût de contrôle

La solution proposée reposant sur l'extension du protocole de routage existant, donc sur l'échange d'informations supplémentaires entre les nœuds du réseau, il est légitime de se demander quel est l'impact sur le fonctionnement général de ce surcoût de contrôle.

On s'attend bien sûr à ce que le surcoût sur les messages HELLO soit considérable, puisque pour chaque adresse de voisin publiée, on rajoute un ensemble de valeurs de compteurs. Nous avons comparé la taille des messages HELLO dans la stratégie **NC** (donc avec protocole OLSR classique) et dans une des stratégies ***D**. Les courbes de la figure 4.15 présentent les résultats de ces mesures pour un nombre de nœuds total variant de 20 à 100, sur 15 graines différentes et pour des durées totales de 300 secondes. Les intervalles de confiance à 95% sont tellement serrés qu'ils ne sont visibles que sur la courbe ***D** pour les grands nombres de nœuds.

La quantité de trafic de messages HELLO généré dans le cas de ***D** est considérablement plus grande que dans le cas de **NC**, ce qui était à prévoir, connaissant la quantité de données supplémentaires incluse. Pour autant, nous avons observé que le nombre moyen de voisins publiés

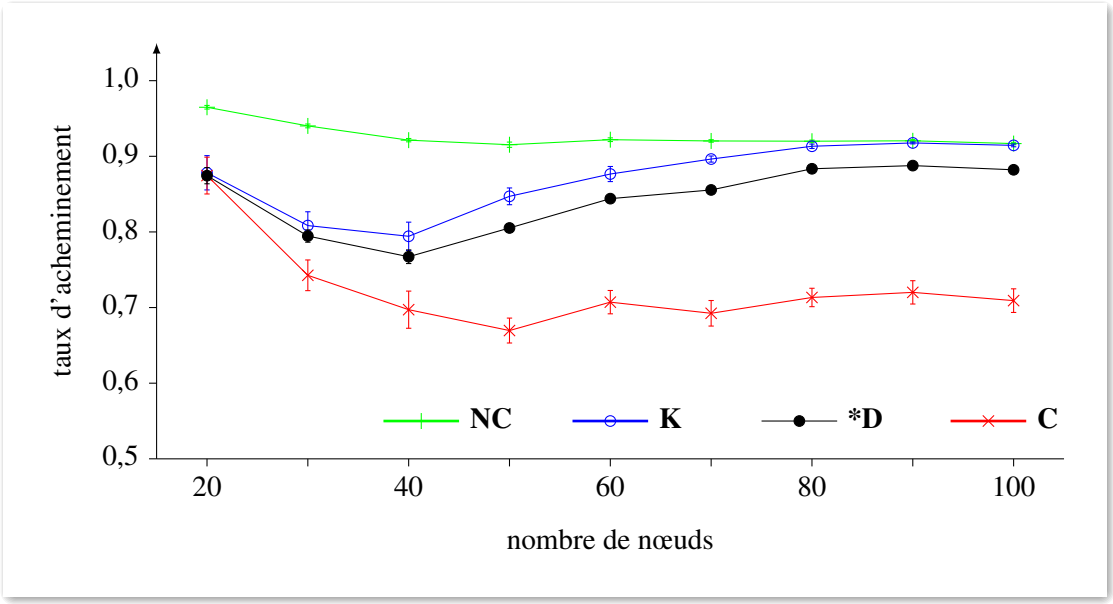


FIG. 4.11: Taux d'acheminement global en fonction de la densité

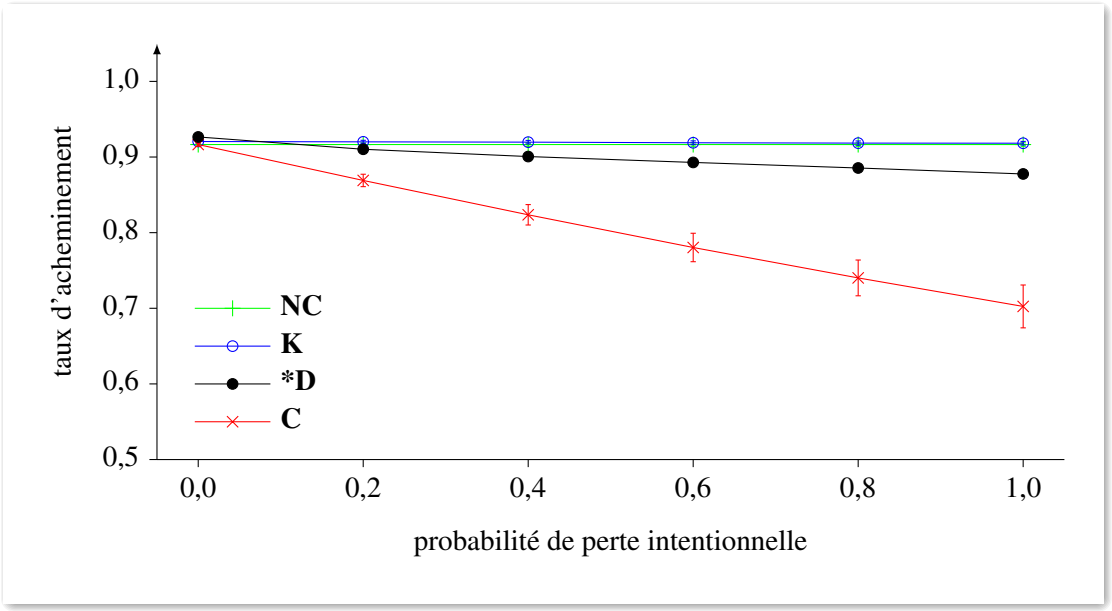


FIG. 4.12: Taux d'acheminement global en fonction de la probabilité de triche, à 100 nœuds

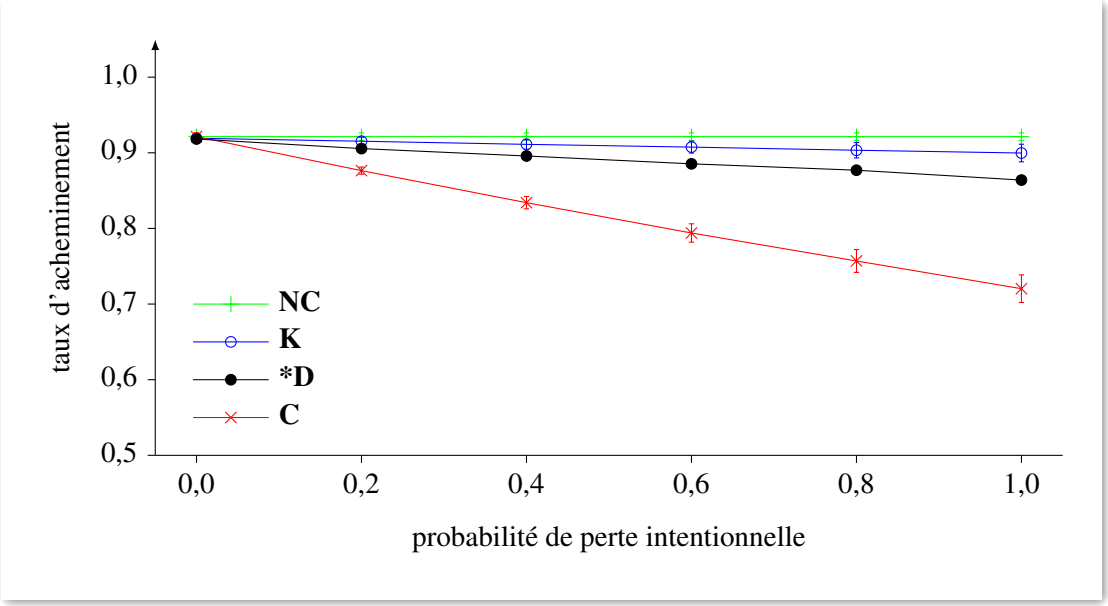


FIG. 4.13: Taux d'acheminement global en fonction de la probabilité de triche, à 60 nœuds

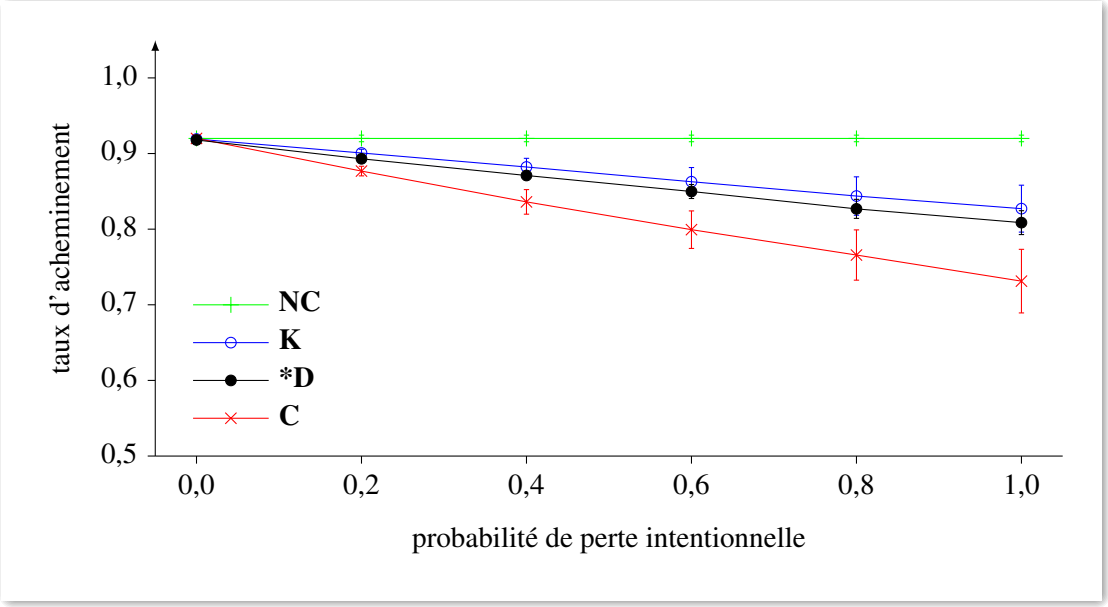


FIG. 4.14: Taux d'acheminement global en fonction de la probabilité de triche, à 40 nœuds

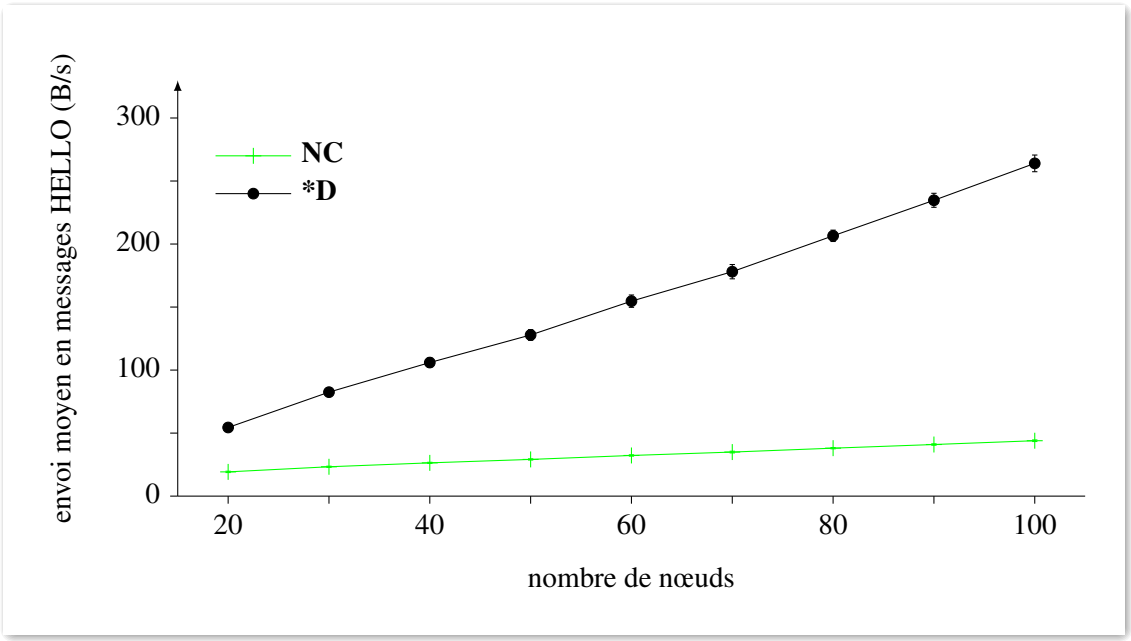


FIG. 4.15: Quantité de données envoyée en moyenne par chaque nœud et par seconde dans les messages HELLO en fonction de la densité

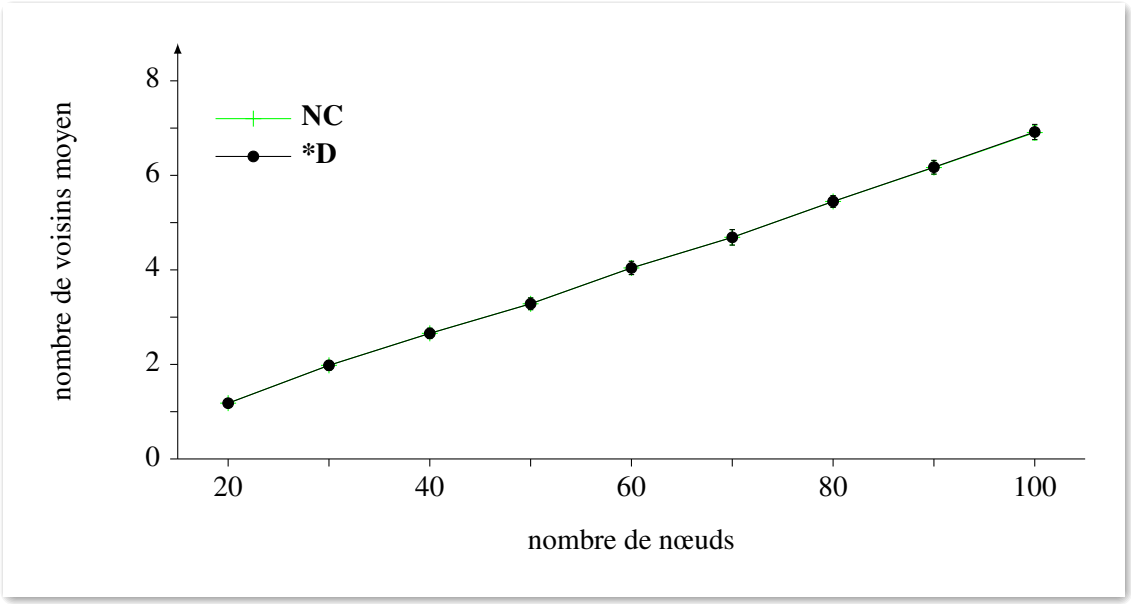


FIG. 4.16: Quantité de voisins publiés en moyenne par chaque nœud et par seconde dans les messages HELLO en fonction de la densité

par nœud et par seconde ne varie pas significativement entre **NC** et ***D** (courbes présentées en figure 4.16).

Concernant les messages TC, nous nous sommes intéressés à la quantité totale de trafic qu'ils engendrent, soit la taille de chacun multipliée par le nombre de (re)transmissions. Cette quantité est considérée en moyenne par nœud et par seconde. Les courbes de la figure 4.17 présentent les résultats qui sont bien moins contrastés que dans le cas des messages HELLO puisqu'ici la quantité de données supplémentaires est plus limitée (degré de suspicion et indice de confiance).

Enfin pour avoir une vision globale de l'impact de l'utilisation de notre méthode, nous nous sommes intéressés au taux de médium libre au niveau de la couche MAC. Il s'agit de la mesure du temps, au niveau d'un nœud, pendant lequel celui-ci n'est ni en train de transmettre, ni en train d'attendre un IFS, ni en train d'attendre la libération du médium (que son drapeau Clear Channel Assessment (CCA) soit levé), ni bloqué par son NAV. En somme, il s'agit de la mesure de tous les instants pendant lesquels un nœud pourrait transmettre immédiatement.

L'augmentation du trafic de contrôle a forcément pour effet de diminuer le temps de médium libre des nœuds. Pour avoir une idée de l'importance de cet effet, nous avons mesuré le taux de temps de médium libre (la portion de temps de simulation pendant laquelle le médium est considéré comme libre) moyen par nœud, pour **NC** et pour ***D**. Les résultats sont présentés sur la figure 4.18, en fonction de la densité du réseau. On constate que l'effet de l'augmentation (quoique considérable) du trafic de contrôle n'a que relativement peu d'effet sur la capacité du réseau (la diminution est inférieure à 5%).

4.8 Conclusion

Dans ce chapitre, nous avons présenté une étude de performances par simulation de notre méthode appliquée au protocole OLSR. Après avoir détaillé le modèle utilisé ainsi que ses divers paramètres, nous avons analysé l'influence de différents facteurs sur le fonctionnement de notre solution.

Nous avons d'abord montré que la vitesse maximale des nœuds dans le modèle de mobilité est un facteur déterminant pour le taux de succès dans l'acheminement des paquets de données de leur source à leur destination. Nous avons expliqué que ce phénomène concerne de façon plus générale OLSR et l'heuristique de sélection des MPRs en particulier.

Ensuite, nous avons vu que le système converge rapidement quelle que soit la stratégie de triche employée par les tricheurs (parmi celles décrites précédemment).

Nous avons étudié l'impact sur le taux d'acheminement des paquets de données des paramètres de densité du réseau et de probabilité de triche. Nous avons constaté que dans tous les cas notre solution permet d'atteindre des performances très proches de l'optimum.

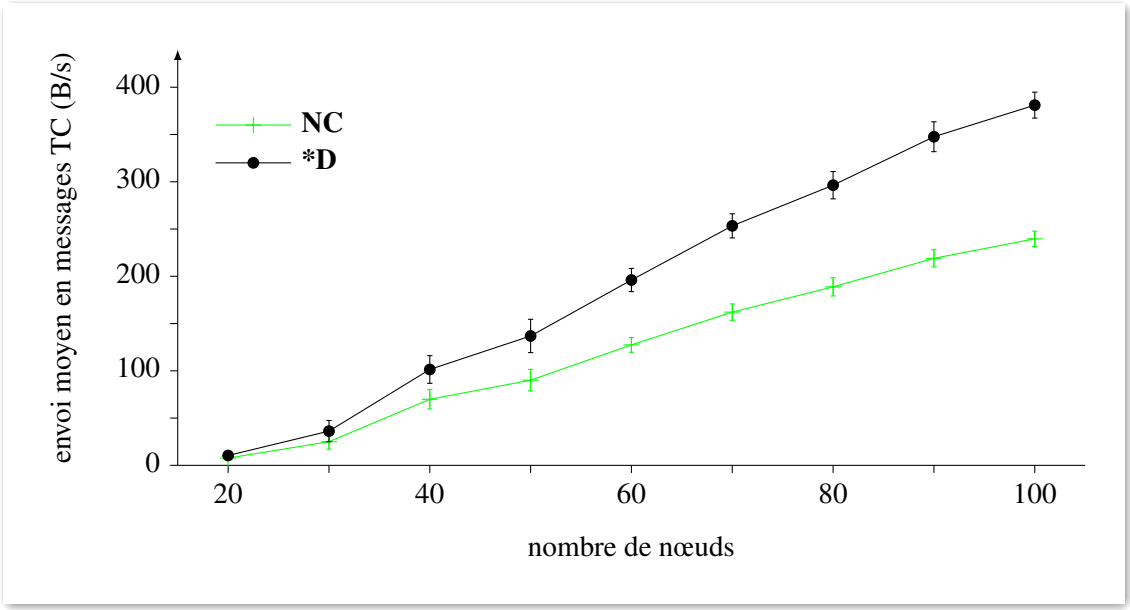


FIG. 4.17: Quantité de données envoyée en moyenne par chaque nœud et par seconde dans les messages TC en fonction de la densité

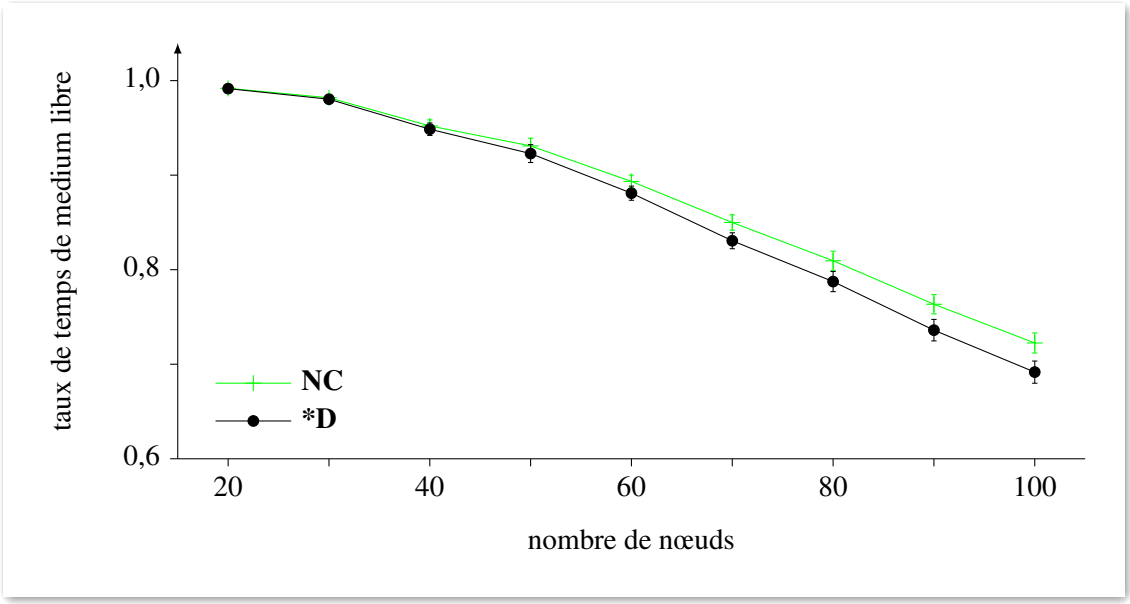


FIG. 4.18: Taux de temps de médium libre en fonction de la densité

Enfin nous avons étudié l'impact de l'utilisation de notre méthode en termes de surcoût de contrôle. Nous avons pu voir que ce surcoût reste faible par comparaison avec le fonctionnement du protocole OLSR classique, ce qui est très encourageant.

Nous sommes bien conscients que l'étude d'une solution dans un simulateur n'a pas d'autre intérêt que l'analyse comparative. C'est pour cette raison que nous avons toujours présenté nos résultats par comparaison à des scénarios de référence. Pour véritablement éprouver les performances d'une solution concernant les réseaux, il est inévitable de la mettre en œuvre sur une installation avec des machines et un environnement réels. C'est une habitude prise dans notre équipe et un requis de plus en plus fréquent des grandes conférences du domaine. Notre expérience nous a montré plus d'une fois que certains problèmes ne font surface qu'au moment de la mise en œuvre d'une solution sur de réelles machines. C'est pour ces raisons que nous mettons en œuvre nos solutions dans notre plateforme expérimentale, dont il est question dans le chapitre suivant.

Chapitre 5

Plateforme expérimentale

Avant même le début de cette thèse, l'équipe Réseaux Mobiles du LRI était impliquée dans des projets concernant le protocole OLSR et la qualité de service. Il est donc arrivé un moment où le besoin de mettre en œuvre les résultats de recherche de l'équipe dans un logiciel exploitant de réelles interfaces réseau s'est fait sentir. C'est ainsi que nous avons pris la décision d'entreprendre le développement de la plateforme Qolyester [H], dans le cadre d'un projet que nous avons baptisé QOLSR.

Je me suis chargé alors de la mise en œuvre partie de zéro du protocole OLSR, tel que décrit dans le draft 10, puis 11 et enfin dans le RFC 3626. Nous avons décidé d'utiliser le langage C++ [59] puisque celui-ci offre la possibilité d'appliquer divers paradigmes de programmation à la fois (impératif, orienté objet, fonctionnel, généricité statique, etc), tout en permettant l'utilisation directe des appels systèmes du système d'exploitation et la bibliothèque standard du C (ainsi que la plupart des bibliothèques C). C'était donc la réponse la plus adaptée au besoin de programmer rapidement et de façon modulaire des algorithmes complexes de haut niveau et d'accéder aux fonctionnalités réseau de bas niveau.

Au cours de cette thèse, j'ai supervisé la poursuite du développement de Qolyester qui avait pour objectif de mettre en œuvre notre proposition pour l'évitement des nœuds malveillants. En effet, il apparaît aujourd'hui essentiel de disposer de résultats de mise en œuvre réelle d'une solution vis-à-vis des grandes conférences du domaine. Les simulations seules ne peuvent attester de la validité absolue d'une solution, tant les modèles, notamment de couche physique, sont des approximations grossières de la réalité. Il est indispensable de valider notre solution par une mise en place expérimentale sur des machines et dans un environnement réels. Ce n'est que par manque de temps que cette mise en œuvre n'a pu être terminée à ce jour et reste dans les perspectives à court terme.

Après une phase initiale, une fois le protocole OLSR classique fonctionnel, le logiciel a été

mis à disposition du public sous licence libre GPL et empaqueté selon les règles en vigueur dans le monde du logiciel libre. Durant les mois qui ont suivi, une branche de développement a été poursuivie en interne qui avait pour but de mettre en œuvre les algorithmes de qualité de service. Ces divers développement ont permis de faire remonter des problèmes théoriques intéressant qui n'avaient pas été prévus et dont la résolution a donné lieu à plusieurs articles présentés à des conférences internationales.

Dans ce chapitre, nous présentons l'architecture générale de la plateforme ainsi que les techniques de programmation générique utilisées pour sa mise en œuvre. Puis nous exposons les avancements successifs dans le développement, ainsi que les problèmes techniques rencontrés. Enfin nous présentons les perspectives de développement à venir.

5.1 Architecture générale

L'architecture générale du projet, illustrée par la figure 5.1, a été conçue de façon à être facilement extensible avec de nouveaux messages, de nouveaux ensembles y correspondant, de nouveaux algorithmes de routage ou encore de nouvelles interfaces avec l'extérieur (applications, noyau, etc). Elle consiste en plusieurs composants distincts dont le fonctionnement est indépendant des détails de mise en œuvre du reste du projet.

5.1.1 L'ordonnanceur et les évènements

Le composant principal, autour duquel tout le reste du code s'articule est un ordonnanceur dont le rôle est de gérer le déclenchement d'évènements temporisés et d'entrée-sortie. Un évènement est simplement une procédure, ou un ensemble d'opérations à effectuer. Par exemple l'arrivée d'un paquet sur le réseau est géré par un évènement d'entrée-sortie qui se charge de le

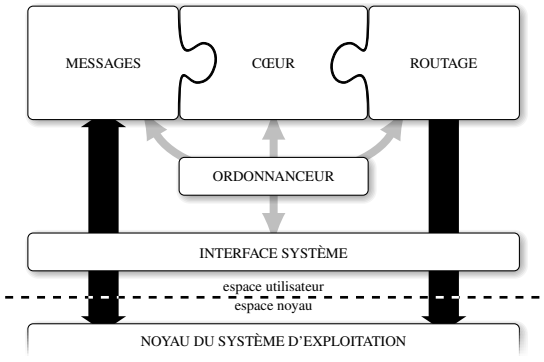


FIG. 5.1: Architecture générale du système

récupérer, le décoder et d'agir en conséquence sur les données en mémoire. Un autre exemple, l'envoi de messages HELLO, est pris en charge par un évènement temporisé périodique qui se déclenche aux bons moments et qui génère le message et le met dans la file d'attente associée à un évènement d'entrée-sortie qui l'enverra dans un paquet sur le réseau.

Les évènements sont des instances issues de la hiérarchie présentée sur la figure 5.2. Cette hiérarchie a été conçue dans l'optique d'être facilement extensible et donc de fournir un socle suffisamment général pour un grand nombre d'ajouts ultérieurs. L'ordonnanceur tel qu'il est conçu, manipule séparément tous les objets de la hiérarchie des évènements d'entrée/sortie (***IOEvent***) d'une part et tous les objets de la hiérarchie des évènements temporisés (***TimedEvent***) d'autre part. Il est toujours possible de rajouter d'autres types d'évènements (héritant de la classe ***Event***) mais alors le code de l'ordonnanceur doit être modifié. Tout autre partie de la hiérarchie peut être enrichie indépendemment de l'ordonnanceur.

Tout évènement concret dispose d'une mise en œuvre de la méthode abstraite *handle()* qui est appelée par l'ordonnanceur pour « gérer » l'évènement en question.

Les évènements d'entrée/sortie exposent en plus en interne des informations de bas niveau qui servent à indiquer au système d'exploitation quelles sont les ressources d'entrée/sortie auxquelles ils se rattachent. Ainsi l'ordonnanceur peut demander au noyau du système de scruter ces ressources afin de savoir à quel moment déclencher quel évènement.

Les évènements temporisés disposent simplement d'un attribut *next* indiquant à quel instant ils doivent se déclencher. En tant que tel, un évènement temporisé est en quelque sorte « jetable », au sens où une fois déclenché, il est en principe détruit. Les seuls qui dérogent à cette règle sont les évènements périodiques (**PeriodicEvent**) qui, une fois déclenchés, se réenregistrent auprès de l'ordonnanceur avec une nouvelle valeur d'attribut *next* au lieu de se détruire. Signalons ici que **PeriodicEvent** est une méta-classe paramétrée par la classe des foncteurs d'incrémentation, dont le rôle est de modifier l'attribut *next* avant le réenregistrement de l'évènement. Ainsi on peut facilement définir des classes d'évènement approximativement périodiques qui modifient leur période d'une certaine manière à chaque déclenchement, au lieu de garder une période fixe. Un exemple d'une telle classe est **JitteredEvent**, dont la date de déclenchement suivant est tirée uniformément entre la période ôtée d'une valeur de gigue et la période. La génération de messages HELLO est un exemple typique d'un tel évènement. La classe **StatePrinter** est une instance de la méta-classe **PeriodicEvent** où le foncteur d'incrémentation est **TrivialIncrement** qui ajoute à **next** la valeur de la période. Son rôle est d'afficher toutes les secondes l'état des ensembles du cœur.

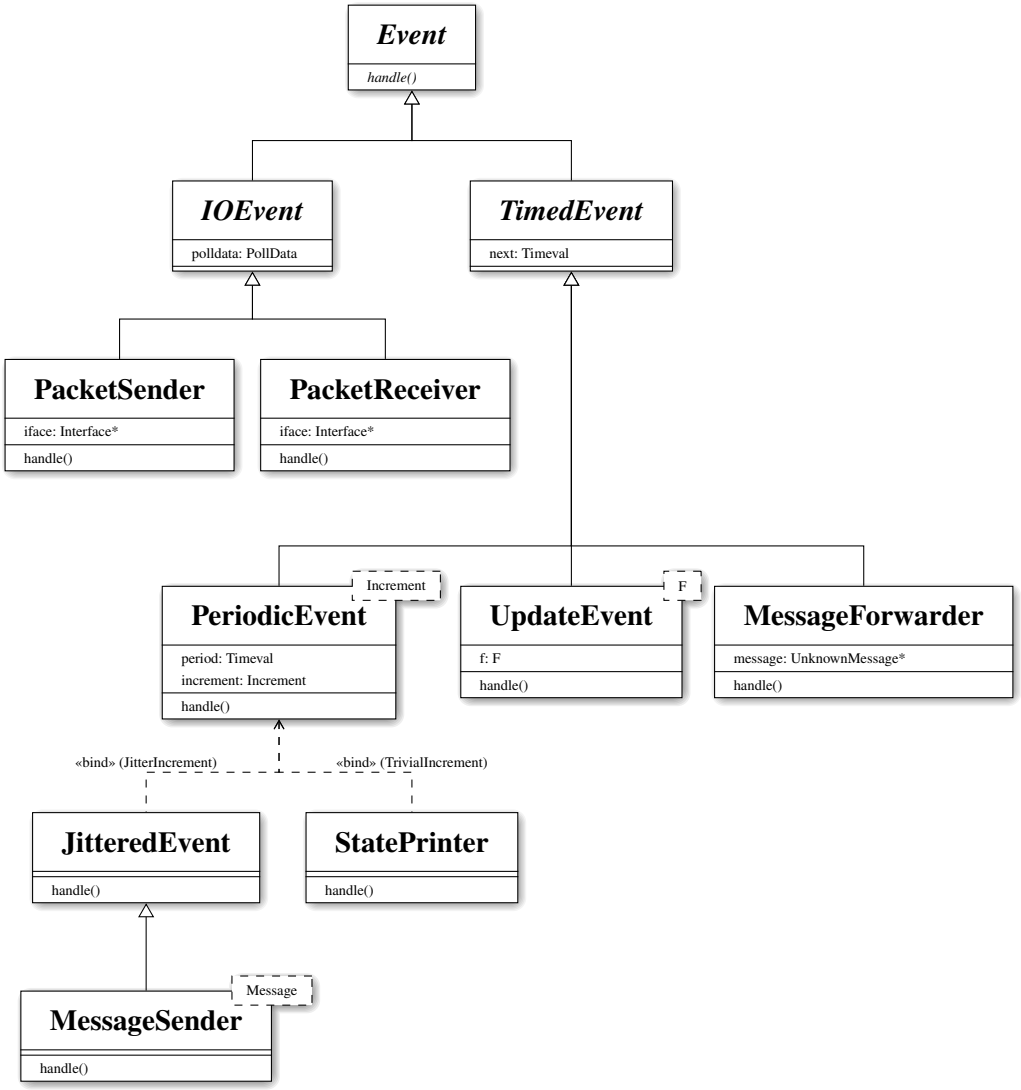


FIG. 5.2: Diagramme UML simplifié de la hiérarchie des évènements

5.1.2 Le gestionnaire de paquets et de messages

Le composant qui se charge de décoder les paquets puis les messages et de générer le contenu des nouveaux messages est le gestionnaire de messages. Ses procédures sont appelées depuis des évènements de l'ordonnanceur (pour les messages entrants et les messages sortants périodiques) mais aussi depuis d'autres composants qui peuvent déclencher la génération d'un message.

La hiérarchie des messages est présentée sur la figure 5.3. Chaque classe fournit sa propre fonction d'interprétation d'un message et sa méthode de génération. La fonction de retransmission, fournie par la classe abstraite ***Message*** est utilisée par toutes les sous-classes, bien qu'il soit possible à l'avenir de la spécialiser pour certains types de messages.

La méthode *dump()* est appelée au moment de la génération d'un message dans un paquet, ce qui permet de détacher l'instanciation d'un message de sa génération proprement dite. Cette approche a le double avantage de permettre de générer le contenu des messages le plus tard possible avant leur envoi et de connaître par conséquent la capacité restante dans le paquet et d'adapter le contenu en conséquence (ce qui permet notamment la génération de messages partiels avec plus de souplesse).

Concernant le décodage des messages, celui-ci est effectué par les fonctions de décodage de chaque message. Une fois le paquet reçu depuis l'interface système, il est pris en charge par la fonction de décodage d'un paquet. Cette fonction identifie les messages qui y sont contenus et appelle les fonctions de décodage, puis les fonctions de retransmission correspondantes. Les données du paquet ne sont pas recopiées en interne lorsqu'un message est programmé pour une retransmission mais sont encapsulées dans une instance de classe qui se charge notamment du comptage de références. Ce n'est qu'au moment de la génération du paquet le contenant que la copie des données a lieu. Ainsi l'espace mémoire alloué à la réception d'un paquet n'est libéré que lorsque la dernière référence à une partie de ses données est détruite.

5.1.3 Le cœur et les ensembles abstraits

Au cœur du système se trouvent les ensembles abstraits nécessaires au maintien d'informations relatives aux liens, aux voisins et à la topologie du réseau. Ces ensembles sont manipulés par le gestionnaire de messages qui répercute sur eux les informations reçues dans les messages et par l'ordonnanceur qui se charge de détruire les informations périmées.

Les éléments périssables

Les classes de tous les éléments qui sont périssables (c'est-à-dire qu'on leur associe une date au-delà de laquelle ils doivent être détruits) héritent d'une classe abstraite ***Updatable*** qui se

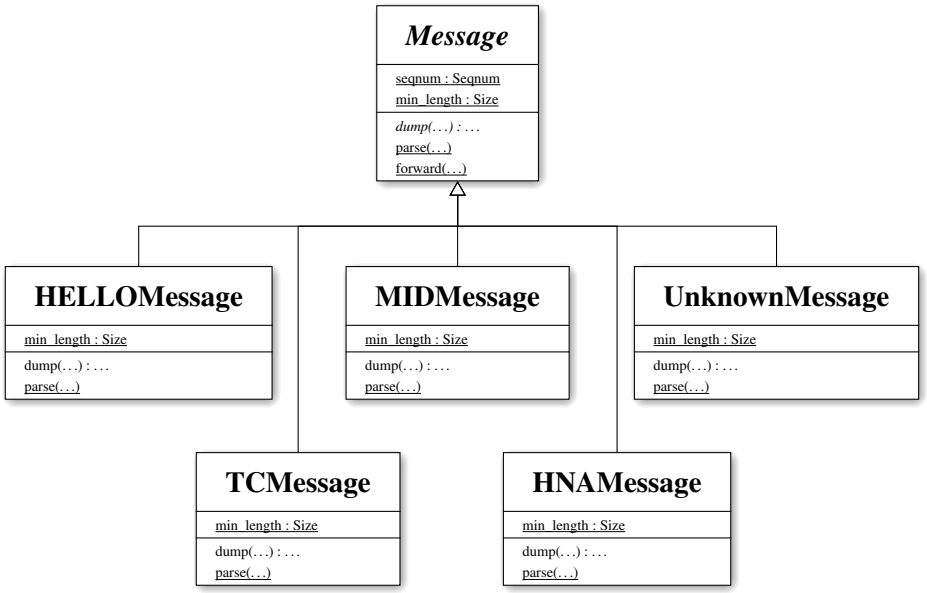


FIG. 5.3: Diagramme UML simplifié de la hiérarchie des messages

charge de gérer les évènements temporisés de type ***UpdateEvent*** associés à l'élément. Ainsi la mise en place de la destruction automatique est transparente du point de vue de l'utilisation de ces éléments. Seules les modifications de date de péremption doivent être accompagnées d'une gestion explicite des évènements.

Cohérence entre liens et voisins

Les ensembles de liens et de voisins ne sont pas indépendants, puisque l'existence d'un voisin est conditionnée par l'existence d'au moins un lien vers celui-ci. De plus, l'état asymétrique, symétrique ou perdu d'un voisin est aussi fonction de ses liens. Par conséquent, les ensembles de liens et de voisins ne sont pas utilisés directement, mais leur accès passe par une classe proxy qui assure la cohérence entre eux.

De façon générale, les ensembles sont mis en œuvre par l'utilisation des conteneurs et des mécanismes associés de la Standard Template Library (STL) [60]. Cette bibliothèque standardisée avec le langage C++ permet l'utilisation facile et efficace de toute une variété de structures de données et d'algorithmes, exploitant le mécanisme des templates qui permettent la spécialisation statique du code par le compilateur.

Sous-ensembles de voisins

Les éléments de l'ensemble des voisins ne sont pas toujours ceux que l'on a besoin de prendre en compte dans les diverses parties du projet. En effet, les voisins peuvent être symétriques, asymétriques ou perdus, alors que pour la génération des messages TC par exemple seuls les voisins symétriques sont nécessaires. Pour ne pas avoir à gérer d'ensembles distincts pour ces différents voisins (avec toutes les complications que ça comporte), sans toutefois devoir distinguer les différents états d'éléments dans tous les parcours de l'ensemble, nous avons appliqué un design pattern particulier, qui consiste à encapsuler cette distinction dans un nouveau type itérateur de l'ensemble. Ainsi c'est au moment du parcours (ou encore de la recherche) de l'ensemble et plus précisément lors de l'incrémentation de cet itérateur que l'état de l'élément est considéré et l'élément possiblement sauté. Cette solution permet d'économiser la mémoire tout en masquant les détails des distinctions dans un conteneur qui présente tous les aspects d'un conteneur STL classique.

5.1.4 Le routage

Pour faciliter le calcul des tables de routage, la topologie du réseau est maintenue sous la forme d'un graphe qui est formé sur la base des informations dans les ensembles topologique (manipulé à la réception de messages TC), de liens et de voisins à deux sauts (manipulés à la réceptions de messages HELLO). Les sommets du graphe représentent les nœuds du réseau et les arcs les liens entre ceux-ci. Des poids sur les arcs permettent de distinguer les arcs créés par des informations de l'ensemble topologique de ceux créés par des informations de l'ensemble des liens ou des voisins à deux sauts. Le calcul des chemins est effectué par l'application d'un algorithme de Dijkstra [61] sur le graphe topologique et l'utilisation directe des informations de voisinage et d'interfaces multiples.

Le choix de maintenir un graphe topologique séparé a été fait dans la perspective de rajouter des informations ultérieurement pour le support de différentes métriques de qualité de service.

5.1.5 Interface système

Les différents composants interagissent avec le système d'exploitation sous-jacent par l'intermédiaire d'une interface qui permet d'abstraire les fonctionnalités offertes par celui-ci. Ainsi cette interface fournit un moyen de manipuler les tables de routage et de récupérer des informations relatives aux interfaces réseau disponibles.

Les informations nécessaires sur les interfaces réseau à utiliser sont essentiellement la taille du MTU et les adresses associées (adresses unicast et broadcast ainsi que la longueur du préfixe).

La façon de récupérer ces informations est fortement dépendante du système d'exploitation et de ce fait cette tâche est déléguée à une fonction séparée du reste. Comme dans tous les cas où des fonctions dépendent de l'interface avec le noyau, elles sont séparées du reste pour être plus facilement portées ou changées, sans impliquer une refonte plus importante du code.

Les tables de routage ne sont pas lues depuis le noyau, mais seulement inscrites. Pour ce faire, une copie abstraite de ces tables est maintenue en interne et les changements sont répercutés sur le noyau. Cette approche a deux avantages : d'une part elle permet de maintenir dans les tables abstraites des informations qui ne seront pas nécessairement présentes dans les tables du noyau et d'autre part elle offre la possibilité de déporter les détails de gestion des tables du noyau. En effet, ce dernier point est important puisque différentes règles doivent être respectées lors de la manipulation des tables, qui ne sont certainement pas les mêmes selon les noyaux, ni selon que l'on fonctionne en mode IPv4 ou IPv6.

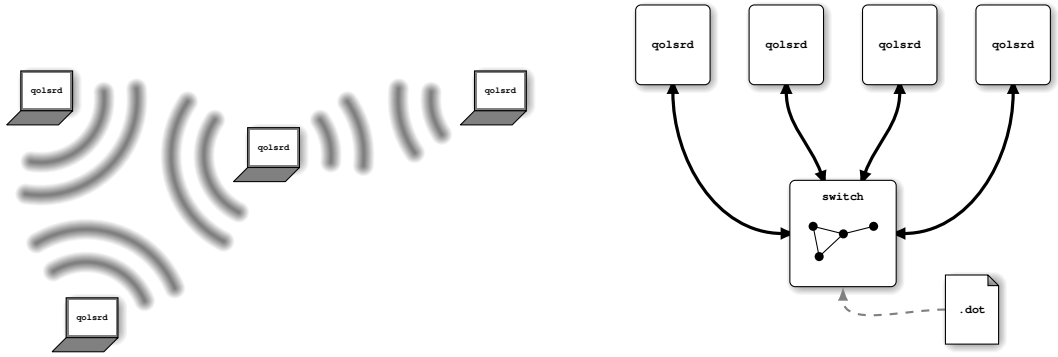
5.1.6 Interfaces virtuelles et simulateur de topologie

Assez tôt dans le développement de Qolyester, nous avons senti le besoin de pouvoir tester la plateforme dans des configurations topologiques diverses. Dans la phase de débogage intense, certains problèmes n'apparaissaient que dans des configurations topologiques particulières, qui n'étaient pas toujours faciles à reproduire dans la réalité. Parmi les problèmes que nous avions à résoudre, on peut distinguer deux catégories : ceux pour lesquels nous suspections un problème dans l'interface système et ceux pour lesquels le problème se situait ailleurs dans l'architecture du projet.

La nature modulaire du code nous a rapidement incité à traiter les problèmes de la seconde catégorie de façon différente. Au lieu de faire fonctionner le code sur des machines physiques différentes et de tenter de reproduire une topologie souhaitée, nous avons mis en œuvre une interface système différente qui simule la(les) interface(s) réseau. Sans changer le reste du code, il nous suffit de demander à la compilation d'utiliser le mode virtuel au lieu du mode réel par défaut. Pour remplacer le médium physique, cette interface système communique (localement sur la même machine physique) avec un processus particulier que nous avons appelé le *switch* et dont le rôle est de transmettre les messages entre les interfaces virtuelles des différentes instances de Qolyester. Ainsi il suffit de fournir au switch une description de la topologie dans un fichier de configuration et de lancer autant d'instances du démon que nécessaire pour mettre le code dans la situation précise où apparaît le problème à résoudre.

Cette fonctionnalité s'est avérée extrêmement utile pour la suite du développement, notamment en ce qui concerne la phase de mise en œuvre des algorithmes de qualité de service.

Avec le temps, les fonctionnalités offertes par le switch ont été enrichies, pour passer d'un



(a) (b)

FIG. 5.4: Interfaces réelles (a) et virtuelles (b) : on obtient l'équivalent d'une topologie réelle en connectant les instances de Qolyester à son switch, qui permet de simuler une connectivité sans changer le cœur du code.

switch à la topologie statique qui ne permettait que de transmettre des paquets de contrôle OLSR à un switch à la topologie variable qui permet de simuler également des paquets de données (par exemple pour simuler un *ping* ou un *traceroute*).

5.1.7 Diffusion du code et logiciel libre

Qolyester a été développé sous forme de démon pour le système d'exploitation GNU/Linux et mis sous licence GNU General Public Licence [62] ce qui assure au code d'être disponible pour tout un chacun tout en restant la propriété intellectuelle du LRI. Le code source de la version d'OLSR classique est diffusé sur le site du projet QOLSR [63] qui est mis à jour à chaque modification fonctionnelle du code.

Compilation du code et Autotools

Comme dans l'écrasante majorité des cas de projets de logiciel libre, nous avons utilisé la suite de logiciels Autotools [64] pour faciliter la compilation. Il s'agit principalement des outils Autoconf et Automake dont le rôle est respectivement de générer et maintenir aisément un script shell de configuration des sources avant la compilation (choix des différentes options de compilation ou cross-compilation, vérification de la disponibilité des diverses bibliothèques requises, etc) et de générer et maintenir aisément une hiérarchie de fichiers Makefile qui permettent la compilation (et la recompilation éventuelle dans les phases de développement) automatique des composants qui le nécessitent.

Génération de la documentation en ligne avec Doxygen

Doxygen [65] est un outil de génération automatique de documentation de code, qui se base sur des commentaires particuliers placés dans le code lui-même. Même lorsqu'aucun commentaire n'est rajouté, les outils de Doxygen permettent déjà de créer un certain nombre de pages qui facilitent la navigation dans le code. Le format de sortie peut être soit destiné à l'impression paginée (format PDF) soit à la consultation en ligne (format HTML).

5.2 Expérimentations en environnement réel

À plusieurs reprises durant le développement de Qolyester, nous avons eu l'opportunité de tester son fonctionnement dans un environnement réel. Dans un premier temps, aussitôt qu'une version initiale était disponible, nous avons procédé à des tests réels dans les locaux du LRI. Ensuite, à trois reprises, nous avons eu l'occasion de tester l'inter-opérabilité de Qolyester avec les autres mises en œuvres du protocole OLSR lors de journées OLSR Interop Workshop, ayant eu lieu à San Diego en 2004, à Paris en 2005 et à Tokyo et Niigata en 2006. Enfin des expérimentations dans la gare Montparnasse à Paris ont été menées dans le cadre du projet RNRT SAFARI.

5.2.1 Plateforme matérielle du LRI

Outre les différents postes de bureau équipés de cartes PCMCIA via bridge PCI, de marque Lucent (chipset Orinoco), l'équipe Réseaux Mobiles du LRI dispose de six PDA de type iPAQ, fonctionnant sous noyau Linux (distribution OPIE), eux aussi équipés des mêmes cartes PCMCIA.

Cet ensemble de machines nous a permis d'effectuer des tests de fonctionnement du protocole dans les couloirs du LRI et d'établir ainsi de petites topologies à plusieurs sauts. L'intérêt de ces expériences était avant tout d'identifier les différents problèmes liés à la configuration du matériel qui pouvait être en partie prise en charge par Qolyester (par exemple désactiver les envois de paquets ICMP Retransmit lorsqu'un paquet arrive et dont la destination est sur la même interface). Par ailleurs, l'impératif de cross-compiler le logiciel pour l'architecture StrongARM a pu mettre en évidence certaines erreurs dans le code, dues à des suppositions de bas-niveau propres aux architectures Intel 32 bits (x86).

5.2.2 Tests d'interopérabilité

Les différentes mises en places de tests d'interopérabilité lors des journées OLSR Interop ont permis d'identifier à la fois des problèmes d'interprétation de la spécification du protocole et des erreurs de programmation qui n'apparaissaient qu'en environnement hétérogène. Les participants à ces tests sont des personnes, des équipes ou des organismes en charge de produire une réalisation logicielle du protocole. Les origines géographiques diverses, les différents langages de programmations et les différents matériels utilisés ainsi que les intentions d'applications allant de la fourniture gratuite d'accès Internet en milieu urbain jusqu'aux applications militaires aux réseaux de capteurs ont permis de tester une grande variété de fonctionnalités.

En outre, ces tests de réseaux de plus grande taille (ici on pouvait compter au moins une quinzaine de nœuds et parfois une dizaine de sauts) ont conduit à identifier des problèmes liés à leur mise en place sur du matériel conforme au standard IEEE 802.11b. En effet il est apparu que le mode ad hoc de ce standard n'est pas bien adapté à la réalisation de réseaux ad hoc mobiles multisauts (notamment à cause du choix du champ BSSID lors de la mise en place de nouvelles « cellules », empêchant deux réseaux formés séparément de fusionner ultérieurement). Nous nous sommes donc résolus à l'utilisation du mode « Ad Hoc Demo » disponible sur certains chipsets, qui est un mode ad hoc simplifié et qui fonctionne bien mieux en environnement multisaut.

5.2.3 Test avec applications en environnement public

Les tests dans la gare Montparnasse consistaient en cinq machines fixes placées à des endroits stratégiques de la gare (en hauteur pour ne pas gêner ni être gêné par les usagers) et un ensemble de machines mobiles sous la forme de PDAs iPAQ et ordinateurs portables.

Lors de tests de trafic applicatif (transfert de fichier par FTP ou SSH) sur le réseau ad hoc, un problème important a pu être identifié ayant trait aux transmissions des messages HELLO par des trames broadcast de la couche MAC IEEE 802.11. La figure 5.5 présente la situation qui a posé problème : quatre nœuds A, B, C et D placés de telle sorte que les nœuds A et C ne détectent pas le signal en provenance de D, puisque celui-ci est stoppé par des obstacles symbolisés par les zones hachurées. Lorsqu'un trafic intense a lieu entre les nœuds A et C en passant par B, alors le lien entre B et D est rapidement perdu. En analysant plus en détails ce cas particulier, nous sommes arrivés à la conclusion qu'il s'agit là d'un problème lié à la couche MAC IEEE 802.11 utilisée. En effet, lorsqu'un trafic TCP soutenu, qui consiste uniquement en des trames unicast, passe du nœud A au nœud B, puis du nœud B au nœud C, alors les trames broadcast contenant les messages de contrôle en provenance de D ont de fortes chances d'entrer en collision avec des trames unicast provenant de A ou de C au niveau de B. Le nœud B est pratiquement tout le temps

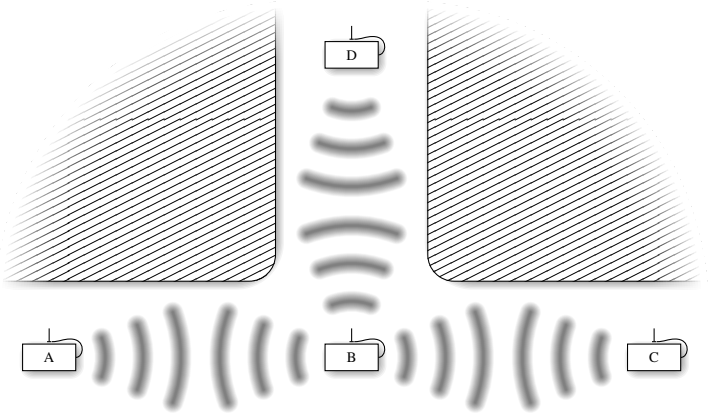


FIG. 5.5: Mise en place à la gare Montparnasse

soit en train d'écouter une trame (qu'elle lui soit destinée ou non), soit en train d'en transmettre une. Étant donné que ni A ni C ne peuvent détecter une transmission de D et vice-versa, lorsque D détecte un médium libre et transmet sa trame, il se peut fortement que B soit en fait en train d'en écouter une autre. La transmission de la trame de D échoue mais D n'a aucun moyen de le savoir puisque la perte d'une trame broadcast n'est pas détectable. Si B est aussi la destination de la trame unicast, la transmission de celle-ci échoue également, mais la perte est détectée (par l'absence de trame ACK de la part de B) et la transmission est retentée. Si jamais D commence à transmettre sa trame broadcast alors que B n'est pas déjà en train d'écouter une trame unicast (par exemple lorsque A et C sont en train d'effectuer un backoff), il est fort probable que A ou C commence à transmettre une trame unicast qui entre alors en collision. L'utilisation du contrôle de flux (RTS/CTS) n'arrange ici en rien la situation puisque, d'une part, celui-ci ne peut être utilisé qu'avec des trames unicast, ce qui ne permet pas à A et C de savoir quand D est en train de transmettre une trame broadcast et, d'autre part, un seul envoi de trame RTS par A ou C pendant que D est en train de transmettre suffit à provoquer une collision. À force de ne plus recevoir les messages HELLO en provenance de D, B annonce son lien avec celui-ci comme perdu. Bien que D reçoive toujours les messages de B, le lien n'est plus symétrique.

Une idée de solution partielle pour pallier à ce problème serait de permettre à des nœuds de transmettre leurs messages HELLO dans des paquets unicast à destination des voisins qui commencent à ne plus recevoir leurs messages, en plus des envois en broadcast classiques. Il faudrait donc enrichir le protocole pour permettre à un nœud d'annoncer des liens dont la rupture est imminente. Dans la pratique, cette approche nécessiterait la connaissance de l'adresse MAC du voisin en question, puisqu'autrement il pourrait arriver que l'envoi du paquet unicast déclenche une résolution ARP (ou Neighbor Discovery pour IPv6) qui se déroule en broadcast

(et qui n'aurait pas plus de chances d'aboutir que les messages HELLO que l'on cherche à faire passer). Ces adresses pourraient être extraites de l'en-tête MAC de la trame contenant un paquet OLSR et ainsi le démon de routage pourrait être responsable du remplissage de la table des voisins dans le noyau (ou « table ARP » pour IPv4).

Cette solution n'est pas totale, puisque dans le cas où le nœud dont les messages HELLO sont perdus n'est pas connu du nœud qui les perd, alors ce dernier n'a aucun moyen de faire savoir au premier qu'il ne reçoit pas ses messages.

5.3 Développements expérimentaux

Une fois qu'une version supportant le protocole OLSR classique était en étant de marche, nous avons créé une nouvelle branche de développement dans l'optique de mettre en œuvre les résultats théoriques de QOLSR. C'est ainsi en collaboration successive avec Thomas Claveirole, puis Johan Oudinet et enfin Simon Odou, alors ingénieurs-stagiaires dans l'équipe, que ces développements ont été réalisés

5.3.1 Fonctionnalités de qualité de service

Pour la mise en œuvre de la prise en compte de la qualité de service dans le routage, les utilisations de graphes en interne se sont multipliées et c'est pourquoi nous avons fait le choix d'utiliser la bibliothèque Boost [66] et plus particulièrement la Boost Graph Library (BGL) [67] qui jouit d'un solide support d'une communauté de développeurs très active.

Les résultats des travaux de l'équipe en matière de routage OLSR avec qualité de service [68] ont été mis en œuvre et des problèmes théoriques importants ont pu être révélés, alors que les travaux antérieurs basés sur des simulations ne les avaient pas identifiés. Ainsi, les problèmes de collision et d'oscillation de flux à qualité de service en absence de réservation se sont avérés intéressants et ont mené à une présentation à une conférence internationale [69].

Parmi les fonctionnalités développées, on peut citer la mesure du délai aller-retour sur les liens à l'aide des messages HELLO, la recherche de chemins avec contraintes de délai, bande passante, taux de perte et nombre de sauts et l'estimation de la capacité restante sur les liens. Ce dernier point a nécessité le développement de fonctions supplémentaires dans l'interface système, puisqu'il requiert l'utilisation de compteurs de paquets au niveau des interfaces, avec la distinction des adresses MAC source et destination.

5.3.2 Signature des messages TC

Une autre branche de développement expérimentale, réalisée en collaboration avec Thomas Claveirole, a consisté à réaliser la solution proposée par Fourati et al. [70, 40]. Il s'agissait de mettre en œuvre tout le code nécessaire à l'utilisation des signatures partielles et des secrets partagés, en utilisation les fonctions de la bibliothèque Crypto++ [71] pour assurer que le contenu des messages TC émis par un MPR est validé et signé par l'ensemble de ses voisins.

5.4 Mise en œuvre de l'évitement des tricheurs

La réalisation de la solution théorique proposée dans cette thèse et appliquée au protocole OLSR n'a pu être terminée faute de temps. Elle est néanmoins bien avancée, puisque nous disposons déjà d'une grande partie des fonctionnalités requises dans le code de Qolyester. L'aboutissement de cette réalisation est donc laissée en perspective à très court terme.

5.4.1 Protocole OLSR et routage avec qualité de service

Le routage avec qualité de service dans OLSR a été développé de façon générique et l'ajout d'une métrique de fiabilité sur les nœuds ainsi que la recherche de chemins optimisant celle-ci ne pose pas de problème particulier. Seule demeure nécessaire la mise en place de la combinaison par médiane pondérée des degrés de suspicion locaux en degré de suspicion global. Cette partie du développement est aujourd'hui grandement simplifiée grâce à l'utilisation des interfaces virtuelles qui permettent de tester facilement cette partie du code.

5.4.2 Sécurisation des messages OLSR

Le prérequis de protéger les messages de contrôle du protocole est en partie satisfait par la signature distribuée des messages TC, déjà mise en place à l'heure actuelle dans Qolyester. Quant à l'application de la proposition de Raffo et al. [39], elle ne l'est pas encore.

5.4.3 Comptabilisation des paquets

L'infrastructure de bas-niveau nécessaire pour l'utilisation des compteurs est déjà présente dans le code, puisqu'elle est requise pour l'estimation de la capacité des liens. Il ne suffit plus que d'ajouter les quelques détails nécessaires à la solution présentée au chapitre 3.

5.4.4 Publications partielles et retransmissions

Le code nécessaire à la construction des publications partielles ainsi qu'à leur retransmission, tel qu'il est aujourd'hui disponible dans le modèle de simulation développé au cours de cette thèse, est portable de façon directe dans le code de Qolyester. En effet, aucune simplification irréaliste n'a été introduite lors de sa conception et ici le modèle de simulation a fait office de plateforme de développement expérimentale.

5.4.5 Signature et authentification des publications

La mise à disposition d'outils cryptographiques dans Qolyester a été requise par la mise en œuvre des signatures distribuées des messages TC. Ces outils sont donc disponibles pour réaliser la signature des publications ainsi que la vérification de leur authenticité et de leur intégrité.

5.5 Autres développements à venir

Pendant le développement des fonctionnalités de qualité de service dans Qolyester, de nombreux problèmes sont apparus et ont nécessité des solutions qui n'ont pas toutes été encore mises en œuvre. Ainsi pour réaliser le routage avec qualité de service, une fois le chemin trouvé, il ne suffit pas de rajouter les informations le concernant dans les tables de routage pour qu'il soit utilisable par une application. En effet, il faut s'assurer qu'en chaque nœud ayant effectué une réservation pour ce flux, le trafic sortant est correctement géré, pour que les paquets *Best Effort* n'interfèrent pas avec les flux avec qualité de service. Il faut donc mettre en place en chaque nœud un classificateur et des ordonnanceurs de paquets (de type *token bucket*). Ces outils sont déjà rendus disponibles par le noyau Linux et ne demandent qu'à être correctement interfacés avec le démon de routage.

Par ailleurs, il est clair que les applications nécessitant la recherche d'un chemin satisfaisant des contraintes de qualité de service doivent communiquer celles-ci au démon de routage. Il est donc essentiel de définir un protocole de communication entre le démon et les applications. De plus, cette voie de communication, qui peut être mise en œuvre par des sockets UNIX (comme c'est le cas pour les communications entre les applications graphiques et le serveur X11, par exemple), peut aussi servir dans d'autres cas où le démon doit échanger des informations avec d'autres processus. On peut imaginer par exemple une application graphique qui affiche l'état de la topologie OLSR en temps réel, fournie par le démon de routage.

5.6 Conclusion

Dans un domaine de recherche appliquée comme celui des réseaux mobiles, il est à notre sens essentiel de mettre en œuvre les solutions étudiées dans un logiciel fonctionnant sur de réelles machines. Non seulement sans application la recherche appliquée perd son intérêt essentiel, mais il est aussi important que les chercheurs eux-mêmes soient en prise avec les applications de leur travaux, au moins dans un cadre expérimental. Une telle démarche peut garantir la qualité des résultats, puisque ceux-ci sont confrontés au test impitoyable de la réalité.

Dans notre propre expérience, nous avons à maintes reprises constaté que seule la réalisation de certaines solutions a pu faire apparaître des problèmes jusque là omis ou négligés. Ces problèmes passent souvent inaperçus lors de la simulation, puisque les modèles, aussi réalistes soient ils, ne peuvent jamais prendre en compte l'extrême complexité d'un environnement réel. Par ailleurs, le but de la simulation n'étant pas la vérification de la correction d'un modèle, certains raccourcis dans le code, tout à fait licites du point de vue théorique, n'ont pas leur pareille dans une réalisation.

Enfin la mise à disposition du code source sous forme de logiciel libre est selon nous une démarche importante afin de mettre à disposition de tout un chacun le fruit du travail scientifique. Par ailleurs, cette démarche suit celle de toute activité scientifique, consistant à soumette nos résultats à l'avis de nos pairs. Ainsi nous espérons qu'à l'avenir nous parviendrons à mobiliser une communauté de gens motivés pour participer à ce projet, pour que celui-ci prenne suffisamment d'inertie pour se poursuive de lui-même.

Chapitre 6

Conclusion

Dans ce dernier chapitre, nous récapitulons les apports de cette thèse et les implications qu'ils entraînent. Nous reprenons les points essentiels de l'approche de notre solution pour rendre un réseau ad hoc sans fil résistant à la présence de nœuds malveillants puis présentons les perspectives ouvertes pour des travaux à venir.

6.1 Récapitulatif

Les problématiques liées aux réseaux ad hoc ont fait l'objet de recherches très actives ces dernières années. Ces travaux ont donné naissance à de nombreuses propositions notamment dans le domaine du routage. Les protocoles les plus mûrs issus de ces recherches ont été conçus dans l'hypothèse où tous les nœuds participant au réseau sont prêts à coopérer et il existe donc un certain nombre de moyens pour un ou plusieurs nœuds malveillants de nuire à son bon fonctionnement.

L'état de l'art présenté dans cette thèse montre que de nombreux travaux effectués dans le domaine de la sécurité des réseaux ad hoc dépendent de caractéristiques ou de fonctionnalités spécifiques offertes par le matériel sous-jacent utilisé pour communiquer, ce qui restreint les possibilités d'application de ces solutions. Par ailleurs, une bonne part des solutions impliquent une intervention au niveau des couches de bas-niveau prises en charge habituellement par le noyau du système d'exploitation. Cette implication est un frein supplémentaire au large déploiement de ces solutions dans des réseaux publiques. Pour ces raisons, nous nous sommes orientés vers des solutions protocolaires indépendantes du type de matériel utilisé et nécessitant aussi peu d'intervention de bas-niveau que possible, notamment en utilisant les fonctionnalités déjà bien répandues.

De l'état de l'art il nous est apparu clairement que les solutions proposées font largement

abstraction des résultats des travaux dans le domaine de la qualité de service. Or il nous a semblé naturel de considérer le problème de la présence de nœuds malveillants analogue à celui de l'existence de conditions variables sur les liens du réseau quant à la capacité, au délai, ou encore au taux de perte. Puisqu'il apparaît difficilement possible pour les nœuds d'un réseau ad hoc de prendre une décision juste, définitive et univoque concernant le caractère malintentionné de leurs voisins, une approche plus nuancée s'est imposée.

Nous avons donc proposé une méthode en plusieurs étapes relativement indépendantes, ce qui permet d'avoir une souplesse de fonctionnement et des possibilités d'exploitation des résultats accrues. Ces différentes parties se déroulent toutes en parallèle sur chaque nœud pendant tout le temps de son fonctionnement.

La première étape consiste à maintenir des compteurs de paquets sur chaque lien vers un voisin et d'effectuer régulièrement leur relevé. La seconde consiste à échanger les valeurs relevées dans les messages de contrôle périodiques. La troisième étape est l'application de vérification de cohérence sur ces valeurs en vue de vérifier la propriété de conservation de flot au niveau des nœuds voisins. En quatrième étape, les résultats de ces vérifications servent de base au maintien d'un degré de suspicion local de chaque nœud qui fait l'objet d'une vérification. La cinquième étape est la diffusion de ces degrés locaux à tous les nœuds du réseau alors que la sixième est leur récupération et leur combinaison en un degré de suspicion global uniforme de chaque nœud. Ce degré global sert enfin, en septième étape, comme métrique de qualité de service dans un protocole de routage sachant exploiter cette information pour éviter les nœuds les plus suspects.

6.1.1 Contributions

Nos contributions concernent chacune des trois étapes principales de la solution générale, dont nous présentons ici un résumé.

Utilisation des compteurs de paquets

Bien que l'utilisation de compteurs de paquets en soi n'ait rien de particulièrement difficile à mettre en œuvre, l'objectif de leur utilisation dans des voisinages dynamiques nous a conduit à proposer l'utilisation de valeurs dites différentielles. Nous avons aussi proposé une méthode de publication d'un grand nombre de voisins dans des messages de taille limitée qui permet toujours de conduire les vérifications de l'étape suivante en minimisant le retard d'obtention des résultats.

Vérification de la cohérence des compteurs

Pour réaliser la vérification du principe de conservation de flot dans tout le réseau sans connaissance instantanée de la quantité totale de trafic qui a circulé sur chaque lien, nous avons proposé une approche distribuée basée sur des relevés de compteurs locaux en chaque nœud pour chaque lien dont il est une extrémité. Cette méthode de vérification tient compte du fait que les compteurs sont diffusés de manière particulière dans la première étape. Chaque nœud réalise la vérification locale de la conservation de flot en tous ses voisins, mais l'existence de deux valeurs de compteurs pour chaque lien (orienté) entre deux nœuds implique une vérification en deux étapes. La première, la cohérence par nœud, est la vérification du principe de conservation de flot d'après les propres compteurs d'un nœud ; la seconde, la cohérence par lien, est la vérification que les compteurs pour un lien sont cohérents avec les compteurs de l'autre extrémité de ce même lien. Comme les valeurs des extrémités d'un lien sont désynchronisées (elles ne sont pas relevées aux mêmes instants), nous avons proposé une méthode approchée qui exploite au mieux les informations disponibles dans les compteurs eux-mêmes.

Degrés de suspicion locaux et globaux

Les résultats des vérifications de conservation de flot ne peuvent pas être utilisées comme des verdicts sur le caractère indésirable du nœud concerné. Les raisons en sont que des pertes de données ont toujours lieu dans un réseau ad hoc même en l'absence de nœuds malveillants et que des échecs de vérification de cohérence par lien n'indiquent pas laquelle des extrémités est fautive.

Par conséquent, on ne peut définir de critère de malveillance absolue concernant un nœud sur la base des résultats de ces vérifications. Ces résultats ne peuvent qu'influer sur une mesure relative de la suspicion qu'un nœud est malveillant. Nous avons donc défini le degré de suspicion local comme étant une valeur maintenue en fonction des résultats des vérifications dans le temps. L'idée étant qu'un nœud qui est impliqué plus souvent qu'un autre dans des échecs de vérification de cohérence sera plus suspect qu'un autre.

Ces degrés de suspicion locaux ne sont maintenus que sur la base de résultats de première main, autrement dit, il n'est pas certain que tous les autres voisins d'un nœud aient le même degré de suspicion en celui-ci. A fortiori, un nœud qui vient d'arriver dans le voisinage et qui n'a jamais été voisin du nœud en question n'a pas de raison d'avoir un autre degré de suspicion que 0. Ces degrés locaux ne peuvent donc servir qu'à prendre des décision locales, qui n'engagent que le nœud courant. Ils ne peuvent pas servir directement au routage, puisque celui-ci est effectué de façon répartie dans les protocoles par état de lien.

Nous avons défini le degré de suspicion global, dont la valeur est calculée à l'aide des degrés

de suspicion locaux diffusés par les nœuds dans le réseau. La méthode de calcul, basée sur la médiane pondérée, permet d'obtenir des valeurs uniformes et résistantes à la diffamation qui sont donc bien adaptées au calcul réparti des routes.

Performances

L'évaluation des performance par simulation a confirmé que cette approche est valide dans toutes les conditions où des chemins alternatifs existent permettant de contourner les nœuds malveillants. Elle a aussi montré que la convergence est suffisamment rapide pour identifier rapidement les nœuds à risque. Le passage à l'échelle, dans des réseaux à grande densité, est effectué grâce à la possibilité de ne publier qu'une partie des compteurs dans chaque message.

L'impact en termes de surcoût de contrôle a aussi été étudié en mesurant le taux de temps de médium libre par rapport à l'application du protocole OLSR classique. Nous avons pu observer que la différence est minime, tant les messages de contrôle n'occupent qu'une petite part du trafic dans un réseau ad hoc. Il est donc assez clair que l'impact de l'ajout des informations des compteurs dans les messages de contrôle du protocole n'ont pas rendu l'approche trop lourde.

6.2 Perspectives

La conduite des travaux durant cette thèse nous a amenés à considérer plusieurs points d'amélioration possible et/ou de prolongement. Nous allons passer en revue les différents points qui nous semblent aujourd'hui dignes d'intérêt.

6.2.1 Amélioration du fonctionnement d'OLSR

Durant toute la phase d'application de notre solution au protocole OLSR, telle que décrite au paragraphe 3.6, ainsi que lors de l'évaluation par simulation, sans parler des expériences mises en œuvres auparavant, nous avons constaté plusieurs points faibles dans OLSR qu'il est intéressant d'explorer en vue d'améliorer ce protocole. L'échange de messages HELLO est à la fois utilisé pour la découverte de voisinage et pour la mesure de l'état des liens par la suite. Ceci implique que lorsqu'un lien vers le successeur ou la destination d'un paquet est perdu, il existe un temps moyen avant le calcul des nouvelles tables de routage pendant lequel les paquets sont perdus. Plusieurs approches ont été proposées pour pallier à ce problème dans diverses situations où il devient très important. L'heuristique de sélection des MPRs est aussi en cause, comme nous l'avons vu au paragraphe 4.3.1. Au paragraphe 5.2.3, nous avons décrit un problème assez gênant qui apparaît lors de l'application d'OLSR sur un réseau à couche MAC de type IEEE 802.11.

Nous avons proposé une solution partielle à ce problème, mais une solution complète n'est sans doute pas évidente à trouver.

6.2.2 Amélioration de la méthode proposée

Plusieurs points concernant la solution proposée dans ce mémoire invitent à de futurs travaux.

Retransmission des publications

La méthode proposée pour détecter les pertes au niveau des nœuds et des liens nécessite la réception continue des publications de la part de tous les voisins. Or il est tout à fait possible et même fréquent que les messages de contrôle contenant ces publications soient perdus puisque la remise des trames broadcast qui les contiennent est en général non fiable. Pour cette raison nous avons dû utiliser un sous-protocole de retransmission de publications perdues. Cependant, le protocole que nous avons proposé peut diverger dans certains cas particuliers. Autrement dit, on peut trouver un scénario d'échanges et de pertes qui entraîne la retransmission d'un nombre croissant de publications. Bien que cette situation est sans doute rare en pratique, il serait préférable de disposer d'une méthode de retransmission qui ne diverge dans aucun cas.

Collusion des tricheurs

Deux nœuds malveillants en collusion peuvent contourner notre méthode de détection de pertes à condition qu'ils soient voisins directs. Il suffit pour cela que le nœud suivant déclare comme reçus pour lui (donc en tant que destinataire) depuis le précédent la quantité de trafic que ce dernier détruit (et possiblement vice-versa). Dans notre cas, cette vulnérabilité de la méthode vient du fait que les paquets sur un lien ne sont différenciés que par le fait que ce lien est le premier saut, le dernier saut ou un autre saut sur leur chemin. L'ajout de critères supplémentaires qui permettraient de distinguer des sauts supplémentaires entraînerait l'ajout de valeurs supplémentaires à publier et donc un surcoût de contrôle d'autant plus grand. Il est possible cependant que l'hypothèse selon laquelle un nœud donné ne retransmet les paquets que d'un nombre fini de flux limite le nombre total de différentes valeurs à publier.

Faux positifs dans la détection de perte

Lors de nos évaluations par simulation, nous avons observé l'apparition d'un nombre relativement négligeable de faux positifs lors de la détection de perte. La cause en était que la supposition que l'intervalle de temps écoulé entre l'envoi d'une publication par le démon de

routage à une extrémité d'un lien et la réception de celle-ci par le démon de routage à l'autre extrémité est négligeable. En fait il arrive que des publications se « croisent » en chemin et par conséquent les calculs de bilan par lien sont faussés. Une piste à explorer serait de marquer chaque publication avec le temps écoulé depuis la dernière publication et tenter de détecter les cas litigieux au niveau du récepteur de la publication, de façon à effectuer des vérifications moins strictes le cas échéant. Il faudrait cependant étudier dans quelle mesure une telle approche offrirait à un nœud malveillant un moyen de tromper ses voisins.

Vieillissement prématuré

Une des attaques possibles pour provoquer la perte de paquets de données au niveau d'un lien distant est la modification illicite du champ *time to live* (TTL) dans les en-têtes IP. En effet, ce champ devant être modifié par chaque nœud intermédiaire, son intégrité ne peut être protégée directement comme celle du reste du paquet. En décrémentant abusivement cette valeur, un nœud malveillant peut provoquer la destruction du paquet par un nœud intermédiaire avant son arrivée à la destination. Bien qu'il soit simple de prendre en compte les expirations de TTL dans la comptabilisation des paquets de façon ce qu'elles n'impliquent pas d'incohérences dans les compteurs, empêcher ce vieillissement prématuré semble impliquer des changements intrusifs dans la pile TCP/IP. Ce point reste à améliorer dans le cadre de notre solution protocolaire.

6.2.3 Intégration d'autres sources d'information

Il apparaît assez clairement que l'approche en couches indépendantes qui détache la détection de pertes de l'élaboration de la métrique de qualité de service invite à l'intégration de sources supplémentaires d'information sur le comportement des nœuds. On pourrait facilement ajouter d'autres méthodes de détection, comme par exemple la détection de mauvais routage ou de modification de paquet, pour affiner le calcul de la métrique de qualité de service. L'influence de chaque source d'information peut être modulée par la règle associée de modification du degré de suspicion local. Il serait donc possible de donner des priorités aux détections de façon à rendre le système plus sensible à certains comportements plutôt qu'à d'autres (selon que les résultats de détection sont univoques ou non par exemple).

6.2.4 Mise en œuvre

Comme déjà indiqué précédemment, il est essentiel que toutes ces propositions soient réalisées dans notre plateforme expérimentale Qolyester, afin de les tester dans des conditions d'utilisation réelles et éventuellement les rendre accessibles à tous lorsque le code atteint une maturité

suffisante. Cette partie du travail est bien souvent loin d'être triviale et fait toujours surgir des questions imprévues auxquelles des réponses doivent être apportées.

Dans le cas de la plateforme Qolyester, sa vocation expérimentale en fait en quelque sorte un chantier éternellement inachevé, auquel on peut toujours trouver des points d'amélioration possible.

Publication

[A] Ignacy Gawędzki and Khaldoun Al Agha. How to Avoid Packet Droppers with Proactive Routing Protocols for Ad Hoc Networks. *International Journal of Network Management*, number 2, volume 18, 2008.

Communications

Conférences internationales avec comité de lecture

[B] Ignacy Gawędzki and Khaldoun Al Agha. Scalable Exchange of Packet Counters in OLSR. *IFIP Med-Hoc-Net '08*. Palma de Mallorca, Spain, June 2008.

[C] Ignacy Gawędzki and Khaldoun Al Agha. Resilience to Dropping Nodes in Mobile Ad Hoc Networks with Link-State Routing. *7th International IFIP-TC6 Networking Conference*. Singapore, May 2008.

[D] Ignacy Gawędzki and Khaldoun Al Agha. Proactive Resilience to Dropping Nodes in Mobile Ad Hoc Networks. *Asian Internet Engineering Conference (AINTEC) 2006*. Bangkok, Thailand, November 2006.

[E] Hakim Badis, Ignacy Gawędzki and Khaldoun Al Agha. QoS Routing in Ad hoc Networks Using QOLSR with no Need of Explicit Reservation. *IEEE VTC'04-Fall: Vehicular Technology Conference*. Los Angeles, USA, September 2004.

Workshops sans comité de lecture

[F] Ignacy Gawędzki and Khaldoun Al Agha. Detecting Intentional Packet Loss in OLSR. *Third OLSR Interop Workshop*. Tokyo, Japan, October 2006.

[G] Ignacy Gawędzki, Hakim Badis and Khaldoun Al Agha. Link Capacity Estimation in QOLSR. *Second OLSR Interop Workshop*. Palaiseau, France, July 2005.

[H] Ignacy Gawędzki and Khaldoun Al Agha. Qolyester: a Modular OLSR Implementation in C++. *First OLSR Interop Workshop*. San Diego, USA, August 2004.

Bibliographie

[1] S. Corson and J. Macker. Mobile ad hoc networking (MANET) : Routing protocol performance issues and evaluation considerations. RFC 2501, IETF, January 1999.

[2] Internet Engineering Task Force (IETF). `http://www.ietf.org`.

[3] T. Clausen and P. Jacquet. Optimized link state routing (OLSR) protocol. RFC 3626, IETF, October 2003.

[4] David B. Johnson, David A. Maltz, and Josh Broch. DSR : The dynamic source routing protocol for multi-hop wireless ad hoc networks, December 24 2001.

[5] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4. RFC 4728, IETF, February 2007.

[6] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *WMCSA*, pages 90–100. IEEE Computer Society, 1999.

[7] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF, July 2003.

[8] Ian Chakeres and Charles Perkins. Dynamic MANET on-demand (DYMO) routing. Internet-Draft 14, IETF, June 2008.

[9] Thomas Clausen, Christopher Dearlove, Justin Dean, and Cédric Adjih. Generalized MANET packet/message format. Internet-Draft 13, IETF, June 2008.

[10] Bhargav R. Bellur and Richard G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *INFOCOM*, pages 178–186, 1999.

[11] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF). RFC 3684, IETF, February 2004.

[12] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68, 2001.

[13] John McQuillan, Ira Richer, and Eric C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transaction on Communications*, 28(5) :711–719, May 1980.

[14] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. *HICSS*, 09 :298, 2002.

[15] Philippe Jacquet, Anis Laouiti, Pascale Minet, and Laurent Viennot. Performance of multipoint relaying in ad hoc mobile routing protocols. In *NETWORKING '02 : Proceedings of the Second International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols ; Performance of Computer and Communication Networks ; and Mobile and Wireless Communications*, pages 387–398, London, UK, 2002. Springer-Verlag.

[16] Thomas Clausen, Christopher Dearlove, and Justin Dean. MANET neighborhood discovery protocol (NHDP). Internet-Draft 6, IETF, March 2008.

[17] Thomas Clausen, Christopher Dearlove, and Philippe Jacquet. The optimized link state routing protocol version 2. Internet-Draft 6, IETF, June 2008.

[18] C. Wullems, K. Tham, J. Smith, and M. Looi. A trivial denial of service attack on ieee 802.11 direct sequence spread spectrum wireless lans. *Wireless Telecommunications Symposium, 2004*, pages 129–136, May 2004.

[19] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denial of service attacks at the mac layer in wireless ad hoc networks. *MILCOM 2002. Proceedings*, 2 :1118–1123 vol.2, Oct. 2002.

[20] John Bellardo and Stefan Savage. 802.11 denial-of-service attacks : real vulnerabilities and practical solutions. In *SSYM'03 : Proceedings of the 12th conference on USENIX Security Symposium*, pages 2–2, Berkeley, CA, USA, 2003. USENIX Association.

[21] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3) :382–401, 1982.

[22] Radia J. Perlman. *Network Layer Protocols With Byzantine Robustness*. PhD thesis, Massachusetts Institute of Technology, 1988.

[23] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *WiSE '02 : Proceedings of the 3rd ACM workshop on Wireless security*, pages 21–30, New York, NY, USA, 2002. ACM Press.

[24] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. Odsbr : An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks. *ACM Trans. Inf. Syst. Secur.*, 10(4) :1–35, 2008.

[25] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, 1994.

[26] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99 : Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[27] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00 : Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, New York, NY, USA, 2000. ACM Press.

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

[29] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Communication Networks and Distributed Systems M&S*, 2002.

[30] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1) :193–209, 2003.

[31] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne : a secure on-demand routing protocol for ad hoc networks. In *MobiCom '02 : Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 12–23, New York, NY, USA, 2002. ACM Press.

[32] Adrian Perrig, J.D. Tygar, Dawn Song, and Ran Canetti. Efficient authentication and signing of multicast streams over lossy channels. *sp*, 00 :0056, 2000.

[33] Adrian Perrig, Ran Canetti, Dawn Xiaodong Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *NDSS*. The Internet Society, 2001.

[34] Zapata and Asokan. Securing ad hoc routing protocols. In *WiSe : Proceedings of the ACM Workshop on Wireless Security*, 2002.

[35] Leslie Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11) :770–772, 1981.

[36] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD : secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1) :175–192, 2003.

[37] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO '87 : A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, pages 369–378, London, UK, 1988. Springer-Verlag.

[38] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure link state routing for mobile ad hoc networks. In *SAINT Workshops*, pages 379–383. IEEE Computer Society, 2003.

[39] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. An advanced signature system for olsr. In *SASN '04 : Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 10–16, New York, NY, USA, 2004. ACM Press.

[40] Alia Fourati and Khaldoun Al Agha. A shared secret-based algorithm for securing the OLSR routing protocol. *Telecommunication Systems*, 31(2–3) :213–226, March 2006.

[41] Neel Sundaresan. Online trust and reputation systems. In *EC '07 : Proceedings of the 8th ACM conference on Electronic commerce*, pages 366–367, New York, NY, USA, 2007. ACM.

[42] Audun Jösang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2) :618–644, 2007.

[43] Sini Ruohomaa, Lea Kutvonen, and Eleni Koutrouli. Reputation management survey. In *ARES '07 : Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 103–111, Washington, DC, USA, 2007. IEEE Computer Society.

[44] Levente Buttyán and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *MobiHoc*, pages 87–96. ACM, 2000.

[45] Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *MONET*, 8(5) :579–592, 2003.

[46] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite : A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *INFOCOM*, 2003.

[47] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol. In *MobiHoc '02 : Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 226–236, New York, NY, USA, 2002. ACM Press.

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

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

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

[51] Pietro Michiardi and Refik Molva. CORE : a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In Borka Jerman-Blazic and Tomaz Klobucar, editors, *Communications and Multimedia Security*, volume 228 of *IFIP Conference Proceedings*, pages 107–121. Kluwer, 2002.

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

[53] Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting disruptive routers : a distributed network monitoring approach. *Network, IEEE*, 12(5) :50–60, Sep/Oct 1998.

[54] John R. Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *IEEE Symposium on Security and Privacy*, pages 132–141, 2000.

[55] Gerard J. Holzmann. *The Spin Model Checker*. Addison-Wesley Professional, 2004.

[56] A. MCDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks, 1999.

[57] C. Bettstetter. Mobility modeling in wireless networks : categorization, smooth movement, and border effects. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(3) :55–66, 2001.

[58] H.T. Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 34(5) :254–256, May 1946.

[59] ISO/IEC. *Programming languages — C++*, second edition, October 2003.

[60] SGI. *Standard Template Library Programmer's Guide*, 2006. `http://www.sgi.com/tech/stl/`.

[61] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[62] GNU general public license, version 2. `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt`.

[63] QOLSR, QoS with OLSR. `http://qolsr.lri.fr`.

[64] Gary V. Vaughan, Ben Elliston, Tom Tromey, and Ian Lance Taylor. *GNU Autoconf, Automake, and Libtool*. Sams Publishing, 2000.

[65] Doxygen. `http://www.doxygen.org`.

[66] Boost C++ libraries. `http://www.boost.org`.

[67] Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. *Boost Graph Library*, December 2001.

[68] Hakim Badis and Khaldoun Al Agha. QOLSR, QoS routing for ad hoc wireless networks using OLSR. *European Transactions On Telecommunications*, 16(5) :427–442, 2005.

[69] H. Badis, I. Gawedzki, and K. Al Agha. QoS routing in ad hoc networks using QOLSR with no need of explicit reservation. *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, 4, 2004.

[70] Alia Fourati, Khaldoun Al Agha, and Thomas Claveirole. Securing OLSR routes. In Kenjiro Cho and Philippe Jacquet, editors, *AINTEC*, volume 3837 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2005.

[71] *Crypto++ Library Reference Manual*. `http://www.cryptopp.com/docs/ref/`.