Université des Sciences et Technologies de Lille

Numéro d'Ordre : 3506 Année : 2005

Thèse
présentée
le 14/01/2005
par

Michaël Hauspie

pour obtenir le grade de
Docteur en Informatique

Contributions à l'étude des gestionnaires de services distribués dans les réseaux ad hoc

Jury :

Président	:	Vincent Cordonnier	Professeur, U.S.T.L.
Rapporteurs	:	Guy Bernard	Professeur, I.N.T. – Evry
		Éric Fleury	Professeur, I.N.S.A. – Lyon
Examinateurs	:	Christophe Gransart	Chargé de Recherche, I.N.R.E.T.S.
	:	Pierre Paradinas	Professeur titulaire de chaire, C.N.A.M. – Paris
	:	Jean-Jacques Vandewalle	Chercheur, Gemplus Systems Research Labs.
Directeurs	:	Jean-Marc Geib	Professeur, U.S.T.L.
		David Simplot-Ryl	Professeur, U.S.T.L.

Résumé

Les réseaux ad hoc sont des réseaux distribués, auto-organisés ne nécessitant pas d'infrastructure. Les entités formant un tel réseau doivent collaborer afin d'assurer le bon fonctionnement des services réseaux, tel que le routage. Dans un tel environement, de nombreux algorithmes développés pour le monde filaire ne peuvent être adaptés de façon naïve sans entraîner une congestion importante du réseau qui va réduire son efficacité. Notre travail de thèse se penche sur l'étude de la gestion de services. En effet, sans application, le développement d'une architecture comme les réseaux ad hoc est inutile. La gestion de services consiste à fournir tout les moyens possibles pour faciliter et rendre fiable l'utilisation d'applications distribuées.

Nos travaux contribuent à l'étude de deux points précis de la gestion de services. Premièrement, nous fournissons un algorithme permettant de répartir efficacement une information dans le réseau en sélectionnant certains objets du réseau pour être des réplicats de l'information. Cet algorithme peut alors être utilisé pour publier les informations relatives à un service afin de permettre sa recherche. Deuxièmement, nous avons étudié la prédiction de déconnexion entrainée par la mobilité des noeuds. Nous proposons trois solutions basées sur la recherche d'ensemble de chemins disjoints, la recherche de liens critiques et la recherche de noeuds critiques. Les recherches que nous proposons sont entièrement réalisées à partir d'informations locales.

Les résultats obtenus fournissent une base au développement d'un gestionnaire de services distribués. De plus, certains de nos algorithmes (comme la recherche d'ensembles de chemins disjoints) peuvent être réutilisés dans d'autres applications, comme le routage QoS multi-chemins.

Abstract

Ad hoc networks are distributed, self-organized networks which do not need fixed infrastructure. Entities in such networks must collaborate to make network services – such as routing – functional. In such environment, many algorithms from wired networks can not be naively adapted without jamming the network. We focus our work on the study of services management protocols. Indeed, without application, ad hoc networks technologies are useless. Managing services consists in providing a reliable and easy way to develop distributed applications.

Our work contributes to this study in two specific points. First, we provide an algorithm which achieves an efficient dissemination of an information among the nodes of the network. This algorithm can then be used to publish the description of a service among the network to make its research easy. Second, we study the disconnection prediction induced by the mobility of the nodes. We propose three methods, based on disjoint paths, critical links and critical nodes. The algorithms depicted in our work are only based on localized data and are working in a distributed manner.

Our results provide a settlement for the design of a distributed services manager. Moreover, ours results can be used in other application fields such as QoS multi-path routing.

À Agnès

Remerciements

« On parvient rarement à ses fins par ses propres moyens ; il faut toujours compter sur quelqu'un d'autre »

Du diable au coeur, Marie-Claude Bussières-Tremblay.

Je remercie tout particulièrement Vincent Cordonnier pour m'avoir accepté dans son équipe pendant trois ans et avoir accepté de présider le jury de cette thèse.

Je suis très reconnaissant envers Éric Fleury et Guy Bernard pour avoir accepté de rapporter mes travaux. Je les remercie d'avoir apporté tant de soins à la relecture de ce mémoire et d'avoir accepté de figurer dans mon jury.

Je tiens à exprimer ma gratitude à Pierre Paradinas et à Jean-Jacques Vandewalle; Je suis très honoré de leur présence dans mon jury.

Je remercie Jean-Marc Geib, Christophe Gransart et David Simplot-Ryl pour avoir encadré mes travaux durant ces trois années.

Mes remerciements les plus sincères à David Simplot-Ryl qui, non content d'être un excellent encadrant, a compris et a soutenu mes efforts pour mener à bien ce travail de thèse ainsi qu'un projet « personnel » qui m'est cher.

Merci également à Ivan Stojmenović pour notre collaboration fructueuse. Ses conseils et ses avis critiques sur mon travail ont été une aide précieuse.

J'exprime ma plus profonde reconnaissance à tous les membres de l'équipe RD2P passée et présente pour leur support et pour les moments de joies que nous avons partagés lors des quelques années que j'ai passé dans les bureaux 111 et 116. Je remercie en premier lieu mon éternel binôme, Damien Deville, qui supporte, encore au moment où j'écris ces lignes, la dure épreuve de la rédaction de thèse. Je lui souhaite une bonne finition et lui promet de relire ses chapitres et de se

retrouver autour d'un bon verre après nos soutenances respectives. Je ne pourrais jamais oublier les excellentes soirées que l'on a passées ensemble, tantôt remplies de joies, de délires, tantôt de peines, mais qui resteront toujours gravées comme certains de mes meilleurs moments.

Puis, dans un ordre qui n'a aucune importance, je remercie Gilles Grimaud pour les excellentes soirées que nous avons passé autour de produits de son terroir natal ; Julien Cartigny qui, par son excentricité légendaire, m'a apporté d'innombrables crises de rires ; Sébastien Jean pour sa concurrence farouche lors de nos concours à 19XX ; Alexandre Courbot pour nos nombreux « trolls » sur le sujet épineux du choix du meilleur système d'exploitation ; Laurence Caubrière pour son rire doux, discret et délicat ; Jean-Jaques Vandewalle pour son VPN ; Jean Carle pour m'avoir remplacé à Genève quand je croyais avoir perdu mon portefeuille ; Amandine Panier pour notre collaboration fructueuse. Sans avoir de souvenir précis, j'ai obligatoirement au moins une raison de remercier également : Caroline Fontaine, Farid Naït, Jean-Marie Place, Christophe Rippert, Nadia Bel Hadj Aissa, Vincent Benony, François Ingelrest et Mamhoud Taïfour.

Merci à Krupsy, Krupsy Junior et Krupsy Junior II, les cafetières de l'équipe RD2P, qui ont sauvé de nombreuses matinées des effets néfastes d'une nuit trop courte.

Je tiens également remercier l'ensemble des membres du LIFL et en particulier Isabelle Simplot-Ryl, Yann Hodique, Bruno Beaufils, Bernard Carré, Nathalie Devesa, Xavier Redon, Philippe Marquet, Nouredine Melab, Violeta Felea, Alain Fargue, Bruno Boursier, Claude Dehier, Samuel Degrande, Gilles Carin, Bérangère Doby-Dassonville, Annie Dancoisne et tous les autres...

Je voudrais également remercier mes parents pour m'avoir apporté support et soutient, tant psychologique que financier, pendant toute la durée de mes longues études.

Je ne me permettrais surtout pas d'oublier mes amis sans qui rien de ce que je n'ai accomplit n'aurait été possible. En particulier, merci à Frédéric Dhieux et Rémy Obein pour être les meilleurs amis du monde, sur qui on peut compter quelle que soit la situation. Nos années de collocation font, et feront toujours, partie des meilleures que j'ai passées. Merci également à Yann Le Maner, râleur, têtu mais qui n'en est pas moins également un ami hors pair dont je ne voudrais surtout pas me passer. Merci à toi pour nos soirées passées à détruire des centaines de lairs de huntress sur Dantooine... Rendez-vous en Azeroth...

Je remercie également tous les membres de Melting-Pot, bande d'informaticiens qui est passée, depuis le règne de l'Amiga, du code à la pétanque, en passant par le jeu en réseau. Je n'ai toujours pas l'intention de quitter ma position de

veilleur de nuit à chaque club ! Merci donc à Nicolas Guillois, Sebastien Wachter, Jean-Paul De Lemos, Vianney Lecroart, Nicolas Lacour, Benjamin Legros, Guillaume Denry, Cécile Lesgoirres, Matthieu Cardon, Marie Weerts, Marie-Pierre Etienne, Christophe Carron, Samuel Ledjmi, Mélanie Morrin, Mathieu Durand, Yann Le Maner, Damien Deville, Alban Lecocq, Gabriel Bizzotto, François Bonami, Frédéric Dhieux, Rémy Obein, Grégory Bouillez, Fabrice Lété, Grégory Guche, Hervé Meunier et les autres...

Merci également à Fabien et Fanny Lemattre pour les excellents moments passés ensemble, désolé encore pour avoir quitté votre mariage si tôt, de nombreuses lignes ont été ajoutées à ce mémoire grâce à ce sacrifice. Merci également à Rémi Coeugniet, Damien Renaud, Caroline Van Meyel et tous mes amis Calaisiens et Boulonais pour nos fêtes mémorables.

Je tiens à remercier Sébastien Chevalier et Bruno Vanhove qui m'ont fait confiance en me permettant de participer à la réalisation de leur projet. Je souhaite à la société VB2S (Virtual Business Solutions & Systems) une vie longue et prospère et je suis heureux de venir grossir ses rangs dès la fin de ma thèse. J'ai d'ailleurs une petite pensée pour mes nouveaux collègues Nicolas Hériveaux, Nicolas Cocu, Vincent Rousseaux et Yann Le Maner qui vont bien devoir me supporter...

Ces remerciements ne pourraient être complets sans quelques lignes pour la personne qui compte le plus pour moi. Je dédie amoureusement ce mémoire à Agnès Carpentier, devenue ma femme le 16 Octobre 2004, qui partage et embellit ma vie depuis de nombreuses années. Merci de supporter mon manque d'initiative manifeste ainsi que ma tête de linotte. Merci de m'avoir aidé à supporter les moments de doutes, passage quasi-obligé de tout thésard à un moment ou un autre des ses trois années de thèse. Merci d'être là, à mes côtés, tout simplement.

Table des matières

Table des figures

Chapitre 1

Introduction

« La recherche procède par des moments distincts et durables, intuition, aveuglement, exaltation et fièvre. Elle aboutit un jour à cette joie, et connaît cette joie celui qui a vécu des moments singuliers. »

Comment je vois le monde, Albert Einstein.

Les technologies actuelles en matière de réalisation de composants électroniques, et en particulier de microprocesseurs, permettent de développer des équipements de taille et de poids de plus en plus réduits. Cela a permis l'apparition d'objets informatiques portables de plus en plus puissants, tels que les ordinateurs portables et les assistants personnels (PDA [1]). Actuellement, les ordinateurs portables rivalisent presque avec des stations de bureau et les PDA atteignent voire dépassent la puissance des stations d'il y a seulement cinq ans. On peut citer à titre d'exemple les PDA de type «*Pocket PC*» qui sont équipés d'un processeur cadencé à 400 MHz et de 64 Mo de RAM ou plus.

Le caractère portable de ces objets nous incite à considérer une nouvelle approche de l'informatique. Les équipements et les informations qu'ils contiennent étaient jusqu'alors attachés à un lieu (comme un bureau par exemple...) et se retrouvent désormais attachés à un individu.

De plus, les technologies réseaux ayant pris beaucoup d'ampleur ces dernières années, ces objets portables commencent à être munis de moyens de communication. Tout d'abord, et ce depuis déjà quelques années, des possibilités de connexion aux réseaux filaires (grâce aux cartes PCMCIAs notamment) ont été apportées. Le besoin grandissant pour les utilisateurs de pouvoir être connectés en permanence, de pouvoir recevoir leur courrier électronique, leur données depuis leur ordinateur portable ou leur PDA à tout moment et sans contrainte, a entraîné depuis le milieu des années 90 l'arrivée d'interfaces réseau sans fil.

1. Personnal Digital Assistant

Bien que les ordinateurs portables et les nouvelles générations de PDA soient les premiers à bénéficier de cette technologie, ce type d'équipement ne constituent pas le seul qui va nous intéresser dans le cadre de l'informatique mobile. En effet, l'informatique s'intègre de plus en plus dans des domaines grands publics comme l'automobile ou la téléphonie. Dans le cas de l'automobile, on a déjà l'équipement électronique nécessaire et on peut imaginer venir y greffer des possibilité de communications.

Même si des interfaces sans fil telles que les interfaces infrarouges (IrDA [2]) existent, il paraît évident – et ce notamment car les interfaces infrarouges sont directionnelles, c'est à dire que l'émetteur va devoir pointer en direction du récepteur et car elles nécessitent un contact visuel direct – que la technologie principale qui va s'imposer est la communication par voie hertzienne. Cependant, l'adaptation directe des technologies utilisées dans le monde des réseaux filaires (comme Internet par exemple) ne peut être considéré comme viable si l'on souhaite tirer partie du potentiel offert par les interfaces de communication sans fil.

Réseaux filaires et réseaux sans fil, deux approches distinctes

L'utilisation d'une interface sans fil introduit nombres de différences par rapport à la communication par câble.

Tout d'abord, le spectre radio, et donc la capacité disponible pour le transfert de données, est limité par la réglementation. Là où un ajout de câbles suffit pour augmenter le nombre d'utilisateurs pouvant être satisfaits sur un réseau fixe, la bande de fréquence occupée par un réseau mobile ne peut être étendue. Cette restriction limite également le débit disponible imposant la nécessité d'une utilisation efficace du canal.

Ensuite, la qualité des liens radio peut varier avec le temps au gré des diverses interférences et de la mobilité des usagers. Cette situation mène donc à un taux d'erreur de transmission plus important que sur un réseau filaire et surtout à un taux très fluctuant.

Une des différences majeure entre ces deux types de réseaux reste tout de même le caractère dynamique d'un réseau sans fil. Le point d'accès d'une entité sur un réseau filaire est fixe alors que dans le cas d'un réseau sans fil, l'usager peut se connecter depuis différents lieux et peut même changer de point d'accès au cours de sa connexion. Le problème est alors dans un premier temps de retrouver un utilisateur dans le réseau et donc un chemin permettant de l'atteindre. Dans un deuxième temps, il faut pouvoir maintenir sa connexion de façon transparente et ce, malgré sa mobilité et ses changements de zone de communication.

Un autre aspect important est que l'accès physique aux données d'un réseau sans fil est non sécurisé. En effet, si dans le cas d'un réseau filaire on peut pro-

2. Infrared Data Association

téger les points d'accès et les câbles reliant les différents postes, dans le cas d'un réseau sans fil, n'importe qui peut *écouter* le réseau et donc récupérer les données transitant par l'interface air. Ceci va donc impliquer l'utilisation de mécanismes de chiffrement et d'authentification afin d'assurer la confidentialité des données.

Actuellement, la principale utilisation des réseaux sans fil se fait dans le cadre d'une architecture centralisée. En effet, dans le cas du GSM [Sco96] ou de WiFi (IEEE 802.11 b/g [bP99]), les usagers se connectent à un point d'accès central qui leur fournit l'accès au réseau. La recherche actuelle tends néanmoins à proposer des solutions n'utilisant pas de point d'accès et se basant sur la collaboration des entités formant le réseau. Ainsi, chaque objet va servir de routeur afin de relayer les paquets pour supporter la communication d'autres objets n'étant pas à portée de communication les uns des autres. Ce type de réseau est appelé réseau ad hoc et est supporté par un groupe de travail de l'IETF (Internet Engineering Task Force), le groupe MANET (Mobile Ad hoc NETwork) [Man].

De nombreux défis s'offrent à nous quand il s'agit de tenter de fournir les mêmes fonctionnalités dans les réseaux ad hoc que celles qui sont disponibles dans les réseaux filaires. Le problème auquel nous avons décidé de nous attaquer ici est celui de la gestion de services. Comment faire savoir au réseau que l'on fournit un service donné ? Comment rechercher un service ? Comment tenter de maintenir au mieux l'utilisation du service malgré la mobilité du réseau et l'instabilité de la connexité de ce dernier ? Nous allons tenter de fournir des réponses à ces questions par le biais de solutions originales, dédiées à un environnement ad hoc.

Orientation de notre approche

Durant notre travail, nous avons été désireux de nous restreindre à un réseau ad hoc dans sa forme la plus simple. On considère dans la suite du document que l'on ne dispose d'aucune infrastructure ni de système de positionnement. C'est pourquoi les protocoles que nous proposons ne se basent que sur des informations sur la topologie locale du réseau.

Organisation du document

Le **Chapitre 2** permet au lecteur d'acquérir les bases nécessaires pour la compréhension de la suite du document. Nous y ferons un bref historique des télécommunications, et en particulier de la communication sans fil. Nous introduirons ensuite les réseaux ad hoc et nous fournirons les définitions et outils qui ont été à la base de nos contributions. Enfin, nous définirons la notion de service et présenterons les principaux problèmes, liés à leur gestion, qui nous ont amené à proposer les résultats de ce mémoire.

Dans le **Chapitre 3**, nous présenterons notre contribution à la découverte

de services. Nous nous sommes concentrés, dans ce chapitre, à la dissémination d'information dans le réseau de manière efficace. En effet, si des informations sur un service sont réparties efficacement dans le réseau, il est facile de les retrouver en effectuant une recherche très localisée ce qui permet de ne pas surcharger l'ensemble du réseau.

Le **Chapitre 4** propose une solution originale permettant de prédire les partitionnement du réseau. Ceci permet à une application de savoir à l'avance que le service qu'elle utilise va devenir indisponible. Cette dernière pourra alors réagir, par exemple, en recherchant un autre objet fournissant un même service.

Enfin, le **Chapitre 5** conclut nos travaux et propose différentes perspectives de recherche.

Bien que chaque chapitre puisse être lu de manière indépendante, ce mémoire à été conçu comme un tout et il est donc conseillé au lecteur de l'aborder comme tel.

Chapitre 2

État de l'art

« l'admirable féerie à laquelle quelques instants suffisent pour qu'apparaisse près de nous, invisible mais présent, l'être à qui nous voulions parler et qui, restant à sa table, dans la ville qu'il habite [...], sous un ciel différent du nôtre, par un temps qui n'est pas forcément le même, au milieu de circonstances et de préoccupations que nous ignorons et que cet être va nous dire, se trouve tout à coup transporté à des centaines de lieues (lui et toute l'ambiance où il reste plongé) près de notre oreille, au moment où notre caprice l'a ordonné. »

À la Recherche du temps Perdu, Marcel Proust.

Ce chapitre, permettra au lecteur de découvrir – si ce n'est déjà fait – le monde des télécommunications et surtout des réseaux ad hoc. Après un bref rappel sur les différents pionniers qui ont permis aux formidables technologies dont nous ne pourrions plus nous passer de voir le jour, nous présenterons ce dont, au vue de l'engouement actuel de la recherche en réseaux, nous ne pourrons sûrement plus nous passer d'ici quelques années. La description que nous ferons des réseaux ad hoc fournira au lecteur les bases nécessaires à la compréhension de la suite du document et le sensibilisera aux problématiques liées à nos travaux.

2.1 Bref historique des télécommunications

2.1.1 La première transmission aérienne « moderne »

L'histoire de la télécommunication débute bien avant l'ère de l'informatique. De nombreuses civilisations ont tenté de proposer des solutions permettant de communiquer au-delà de la portée de la voix. On peut penser aux signaux de fumée des amérindiens, aux pigeons voyageurs utilisés par les Arabes ou les Chinois avant

les Européens. Cependant, ce n'est que vers la fin du XVIII^{e} siècle que le premier dispositif que l'on peut qualifier de « technologique » fit son apparition.

Dans les années 1790, après la révolution française, la guerre gronde aux portes du pays et la nécessité de pouvoir communiquer sur de longues distances plus rapidement qu'en envoyant un coursier à cheval se fait sentir. Un ingénieur français, Claude Chappe (1763-1805), propose une solution qui sera baptisé « tachygraphe » puis « télégraphe » du grec TELE (loin) et GRAPHEIN (écrire). La machine proposée par Chappe est composée de deux bras (ou voyants) articulés autour d'un troisième (Figure 2.1). En fonction de la position de ces trois voyants, on déduit différents signaux qui forment un code permettant de communiquer à 10 kilomètres grâce à une lunette longue portée. Ces voyants peuvent décrire 196 signaux différents qui, utilisés dans un code connu de l'émetteur et du récepteur du message, permettent de communiquer des ordres, des événements et toutes sortes d'informations. En 1794, plusieurs de ces dispositifs sont répartis entre Paris et Lille afin de permettre à ces deux villes de communiquer. Chaque signal est donc répété de proche en proche de Paris à Lille par les opérateurs de ce dispositif (appelés les stationnaires). La première ligne de télégraphie aérienne était née. Le 1er septembre 1794, la première dépêche télégraphique est transmise avec succès entre ces deux villes[1].

FIG. 2.1 – *Le télégraphe de Chappe*

2.1.2 Le fil comme support de communication

Dans les années 1840, l'américain Samuel Morse invente un télégraphe électrique simple: des piles, un interrupteur, un électro-aimant et des fils suffisent.

1. Ce premier message était destiné à l'assemblé nationale qui siégeait à Paris. *« Condé restituée à la république. La reddition à eu lieu ce matin à 6 heures »*

L'appareil de Morse, qui transmit le premier télégramme public en 1844, ressemblait à un simple commutateur électrique. Il permettait le passage d'un courant pendant une durée prédéfinie puis l'interrompait, le tout étant commandé avec la pression d'un doigt. Le premier récepteur Morse était équipé d'un crayon contrôlé électromagnétiquement. Ce crayon imprimait des marques sur une bande de papier, fixée sur un cylindre animé par un mouvement d'horlogerie. Les marques variaient en longueur suivant la durée des impulsions du courant électrique passant à travers les fils d'un électro-aimant et prenaient la forme visuelle de points et de traits. Par la suite, les opérateurs apprirent à reconnaître directement à l'oreille les traits et les points qui leur étaient transmis. Son appareil fut adopté par la plupart des pays européens et des réseaux nationaux basés sur le télégraphe de Morse virent le jour aux États Unis, en France, en Angleterre... En 1866, après plusieurs essais infructueux, le premier câble transatlantique fut installé et avec lui, le premier véritable réseau mondial de télécommunication se développa.

Aux environs de 1940, la première ère de l'informatique moderne fit son apparition. Rapidement, l'adaptation des technologies de télécommunications à l'informatique fut rapidement incontournable. En 1957, le ministère de la défense américain crée l'agence pour les projets de recherche avancée (ARPA). Dans ce cadre, le besoin de faire communiquer les différentes équipes de recherche aux quatre coins des États Unis se fait ressentir. Ce besoin a mené les chercheurs de l'ARPA à créer l'ARPANET, réseau destiné à relier entre elles les différentes universités du pays, qui grâce à la standardisation du modèle TCP/IP [Tan03, Pos81], évoluera vers l'Internet que nous connaissons actuellement.

2.1.3 La communication sans fil

Depuis peu, les systèmes de communication sans fil offrent aux utilisateurs la possibilité de profiter des joies des télécommunications quelle que soit leur localisation géographique. Pourtant, la communication sans fil est presque aussi vielle que la communication filaire...

En 1887, Heinrich Hertz vérifie par l'expérience les théories de Maxwell. Ces dernières, établies de façon mathématique par James Maxwell, nous disent que toute perturbation électrique donne naissance à des oscillations électromagnétiques. Ces oscillations seront amenées à êtres connues sous le nom d'ondes hertziennes. En 1890, Édouard Branly découvre le premier récepteur sensible aux ondes hertziennes. À partir des travaux de Branly, l'italien Gugliemo Marconi invente le premier appareil de télégraphie sans fil en 1895. Puis, Marconi va de succès en succès en augmentant les distances de transmission pour atteindre, en 1903, la transmission complète d'un message sur une distance de 3400 km !

Jusqu'à la fin des années 1980, la technologie sans fil a surtout été utilisée dans le cadre de la radio, de la télévision ou des communications réservés à d'impor-

tants organismes comme l'armée. L'arrivée des téléphones cellulaires GSM [Sco96] (*Global System for Mobile communication*) a offert à tous la possibilité de communiquer de n'importe où, avec n'importe qui. Cependant, un tel dispositif nécessite le déploiement d'une infrastructure coûteuse devant assurer le relais entre les téléphones portables et le réseau téléphonique filaire.

2.2 Les réseaux ad hoc

2.2.1 Définition

Si le déploiement d'une infrastructure est soit trop coûteuse, soit impossible pour des raisons diverses et variées (environnement hostile, besoin d'un déploiement rapide, ...), on peut envisager une solution distribuée pour permettre aux utilisateurs d'étendre leur communication au-delà de la portée de leur interface radio. Chaque utilisateur du réseau peut relayer les messages afin que tous les utilisateurs puissent joindre tous les autres, quelle que soit la distance qui les sépare, pourvu qu'il y ait assez d'utilisateurs répartis sur le chemin. Le réseau fourni est donc autonome et supporté par la collaboration de l'ensemble des participants. La Figure 2.2 montre un exemple de fonctionnement d'un tel principe. Ici, l'ordinateur A veut communiquer avec l'ordinateur C. Comme ils ne sont pas à portée directe de communication, A va envoyer son message à B, un téléphone, qui va à son tour transmettre le même message à C.

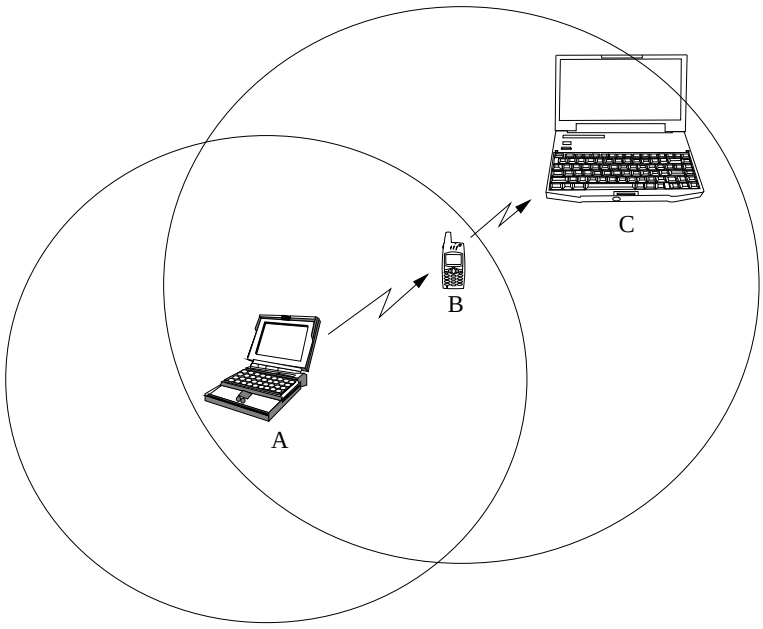


FIG. 2.2 – *Exemple de routage dans un réseau ad hoc.*

Comme souvent, l'idée n'est pas neuve et on peut voir que Chappe et son télégraphe visuel utilisait déjà ce principe (tout au moins pour la base) ! Cependant, s'il est facile de voir comment effectuer un tel relais quand il y a peu d'objets, comment faire quand ce nombre augmente ? Comment faire, par exemple, dans le cas présenté Figure 2.3, pour savoir quels relais utiliser pour transmettre un message d'un point à un autre du réseau ?

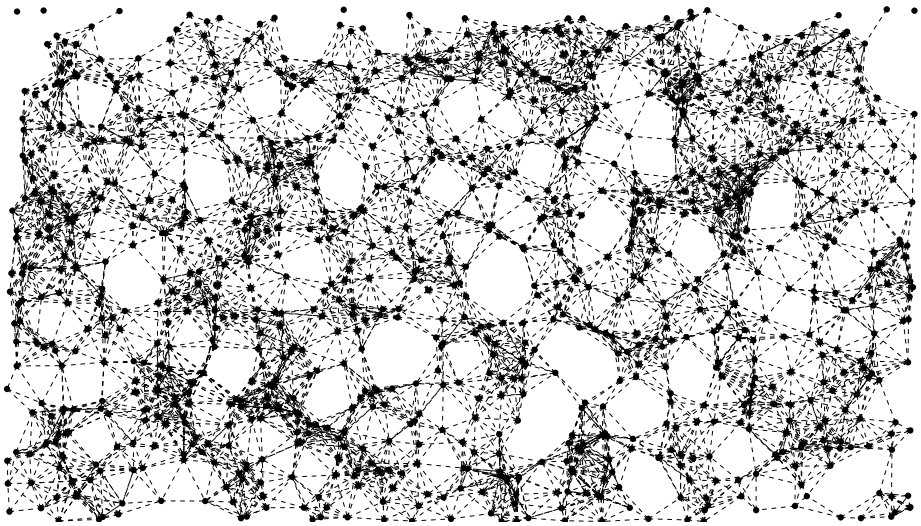


FIG. 2.3 – *Un réseau ad hoc à large échelle.*

Ce type de réseau est appelé réseau ad hoc ou auto-organisé. Ce domaine fait l'objet d'un fort intérêt dans la communauté réseau ce qui s'est traduit par la création d'un groupe spécifique de l'IETF (*Internet Engineering Task Force*) baptisé MANET [Man] (*Mobile Ad hoc NETworks*). Le lecteur pourra trouver des informations précises sur les différents domaines liées au réseaux ad hoc dans des ouvrages tels que [Sto02, BCGSar, Per01, Toh02]

La façon la plus simple de définir un réseau ad hoc est de le considérer comme un réseau à contrôle totalement décentralisé, où les objets communiquent par l'intermédiaire d'une interface sans fil. Par contrôle décentralisé, on entend la non nécessité d'une infrastructure fixe ainsi que le désir de n'utiliser que des informations locales dans la prise de décision des divers protocoles et algorithmes.

2.2.2 Domaines d'application

Historiquement, le premier domaine d'application des réseaux ad hoc fut le domaine militaire. Dans les années 1970, le DARPA (*Defence Advanced Research Projects Agency*) proposent *Packet Radio Network* [Kah77]. Ce protocole permet de déployer un mécanisme de communication entre les différents groupes d'unités par l'intermédiaire de véhicules qui communiquent ensemble par liaison radio.

Ces recherches ont mis en avant les différents fondamentaux liés à ce type de réseaux ainsi que leurs limitations. Nombres de points discutés dans ces travaux, ainsi que certaines des solutions proposées, restent d'actualité plus de vingt ans après. Cependant, ce protocole présente de nombreux défauts, notamment en ce qui concerne la mobilité et la taille de l'équipement. En effet, ce protocole se base sur l'hypothèse d'un mouvement assez lent et de faibles changements de topologie. De plus, les possibilités de l'électronique des années 1970 rendent le matériel encombrant et difficile à utiliser.

L'intérêt des militaires pour une telle technologie s'explique par le caractère particulièrement adapté des réseaux ad hoc aux situations hostiles. En effet, comme le réseau est auto organisé et qu'il ne nécessite pas d'infrastructure, il peut être déployé rapidement, sans difficulté et offrir une bonne tolérances aux pannes. D'où l'utilisation possible également dans le cadre de sinistres [ZS03], comme les tremblements de terre ou les incendies, pour permettre aux équipes de sauvetage de communiquer alors que les infrastructures de communication classiques sont détruites ou pour permettre aux survivants d'établir un réseau aidant à leur localisation. De plus, la possibilité pour les communications d'emprunter plusieurs routes renforce la fiabilité d'un tel réseau ce qui est indispensable dans un tel contexte.

Il est évident que dans un contexte plus commercial les réseaux ad hoc peuvent également servir pour former des réseaux locaux. En effet, la mise en place du réseau est plus simple, pas de câble à tirer dans le bâtiment d'où une économie intéressante pour une entreprise. Dans le cas de réunions temporaires comme des conférences, des *LANs party*[2] ou encore des *coding party*[3], l'organisation est simplifiée, toujours par la non nécessité de câblage. Cependant, des progrès sont encore à faire, notamment en ce qui concerne les couches physiques, car ce genre d'application peut nécessiter une bande passante importante, ce qui n'est pas encore fournit par WiFi à l'heure actuelle par exemple.

Dans le cadre de l'informatique omniprésente (telle que la décrit Mark Weiser [Wei91]), les réseaux ad hoc peuvent servir à relier entre eux tous les équipement de la maison ou établir les liens entre les différents composants informatiques des vêtements (dans le cadre de l'informatique vestimentaire ou *Wearable Computing* [Man96]). Dans ce cas, on parle non plus de LAN (*Local Area Network*) mais de PAN (*Personnal Area Network*) et le routage ad hoc peut être utilisé pour faire communiquer tous les éléments grâce à des équipements de faible puissance, et donc de faible portée (ce qui est un avantage, tant au point de vue de la consommation énergétique qu'au point de vue de la santé de l'utilisateur).

Enfin, les réseaux ad hoc s'avèrent très utiles dans le cas de mesures en milieu hostile. On parle alors de réseaux de senseurs [EGHK99, CSR04]. Les senseurs,

2. Ici, dans le sens de regroupement de joueurs
3. Concours de programmeurs, infographistes et musiciens

chargés de mesurer les propriétés physiques des environnements (comme la température, la pression...), sont dispersés (le plus souvent lâchés d'un avion ou d'un hélicoptère) par centaines, voire par milliers sur le site, effectuent leurs mesures et envoient les résultats à une station par l'intermédiaire d'un routage ad hoc à travers le réseau, souvent très dense, ainsi formé. La caractéristique principale d'une telle application est la forte contrainte d'énergie, de mémoire et la faible capacité de traitement de ces dispositifs.

Comme on peut le voir, les réseaux ad hoc ont un très large potentiel dans un futur proche et l'intérêt que porte la recherche pour ce domaine s'en explique donc très largement. De nombreux défis se posent avant de pouvoir utiliser ce type de réseaux dans toutes les applications citées plus haut. La suite de ce chapitre expose point par point les principales limitations ainsi que les solutions existantes pour y palier. Tout d'abord, nous verrons quelles différences existent en ce qui concerne la gestion de l'accès au médium radio par rapport à l'accès au médium filaire. Puis nous évoquerons le problème du transport des paquets d'une station à une (routage point à point) ou plusieurs autres (diffusion) pour finir par l'aspect qui va nous intéresser pour la suite du document, la gestion des services dans les réseaux ad hoc.

Cependant, avant d'entamer ces problèmes, il est indispensable de terminer cette section par différentes définitions dont nous allons nous servir tout au long de ce document en ce qui concerne la modélisation du réseau.

2.2.3 Modélisation et notations

2.2.3.1 Modélisation par les graphes

Dans la majorité des cas d'étude (et en particulier dans le nôtre), un réseau ad hoc peut être modélisé comme un graphe unitaire [CCJ90] $G = (V,E)$ avec V l'ensemble des nœuds du graphe (à chaque station du réseau correspond un nœud dans le graphe) et E l'ensemble des arcs donnant les possibilités de communication directes entre les stations (nous considérons dans ce document que la communication est symétrique, c'est à dire que si une station peut en entendre une autre, elle peut également se faire entendre d'elle. Le graphe correspondant est alors non orienté). Par la suite, nous utiliserons indifféremment les termes station, nœud ou encore mobile pour désigner une entité formant le réseau. Si on pose uv comme étant la distance physique séparant les nœuds u et v, et R le rayon de communication des nœuds, l'ensemble E se définit par :

$$E = \left\{(u,v) \in V^2 \mid uv \leq R\right\}. \tag{2.1}$$

À partir de cette modélisation de base, nous pouvons fournir différentes définitions qui vont nous servir par la suite.

Définition 1 *Soit u un nœud. On appelle voisinage de u l'ensemble $N(u)$ contenant les nœuds que u peut joindre directement. La relation binaire représentant le lien de communication est supposée symétrique,* i.e. $u' \in N(u) \Leftrightarrow u \in N(u')$ *(*i.e. *tous les nœuds du réseau ont la même portée de communication). On note $\dot{N}(u)$ l'ensemble $N(u) \cup \{u\}$.*

Définition 2 *Soient v et w deux nœuds du réseau. Une route p entre v et w est une suite de nœuds o_1,o_2,...,o_n telle que $o_1 = v$, $o_n = w$ et quel que soit i entier, $1 \leq i < n$, $o_{i+1} \in N(o_i)$. On note $\tilde{p}$ l'ensemble des nœuds appartenant au chemin, soit $\tilde{p} = \bigcup_{i=1}^{n} \{o_i\}$. On note $|p|$ le nombre de* sauts *de p ($|p| = n - 1$). L'ensemble des routes reliant v à w est notée $P(v,w)$.*

Il est évident que toutes les routes disponibles ne sont pas forcément intéressantes pour la communication. Par exemple, les routes très longues ou les routes contenant des boucles ne sont pas intéressantes. Les routes optimales en nombre de sauts sont quant à elles peu nombreuses et sensibles aux changements de topologie [Toh97]. Nous définissons donc ici les types de routes potentiellement intéressantes.

Définition 3 *Soient v et w deux nœuds du réseau et $p = o_1$,o_2,...,$o_n \in P(v,w)$ une route entre v et w.*

1. *p est appelée* ***route optimale*** *si et seulement si $\forall p' \in P(v,w) \quad |p'| \geq |p|$. Si p est optimale, $|p|$ est appelée* ***distance*** *entre v et w et est notée $d(v,w)$.*
2. *p est dite* ***sans boucle*** *si et seulement si pour tout couple i,j d'entiers, $1 \leq i,j \leq n$, $o_i = o_j \Rightarrow i = j$.*
3. *p est dite k-**sous-optimale** (avec $k \geq 1$) si et seulement si p est sans boucle et $|p| < d(v,w) + k$. On note $SOP_k(v,w)$ l'ensemble des routes k-sous-optimales entre v et w.*

2.2.3.2 Réduction du degré d'un graphe

Pour de nombreuses applications, et en particulier celles que nous présentons Chapitre 3, il est intéressant de proposer des algorithmes permettant de diminuer le degré moyen d'un graphe. Le degré moyen d'un graphe est le nombre moyen d'arêtes pour chaque nœud. On peut citer, par exemple, les arbres de recouvrement minimaux (*Minimum Spanning Tree*) [Kru56], les graphes RNG (*Relative Neighborhood Graph*) [Tou80], les graphes de Gabriel [GS69], le graphe des voisins MPR [LQV01] ou les ensembles dominants. Comme nous avons retenu ce dernier ensemble pour notre proposition, nous allons analyser les différents algorithmes possibles pour construire un ensemble dominant de façon distribuée.

Soit $G(V,E)$ un graphe, un sous-ensemble D de V sera dominant si et seulement si tout nœud de V est dans D ou est un voisin d'un nœud de D. Plus formellement, D est dominant si et seulement si $\dot{D} = V$.

Les ensembles dominants peuvent être utilisés dans les domaines du routage [DW03] pour déterminer les nœuds appartenant à l'espace de recherche des routes et de la diffusion [WD03] pour déterminer les nœuds qui vont retransmettre les messages. Dans ce cas, on utilise des ensembles dominants connectés. Un ensemble dominant est un ensemble dominant connecté (CDS, Connected Dominating Set) si le sous-graphe formé par cet ensemble est connexe. Quand un nœud décide de diffuser un message, si tous les nœuds appartenant au CDS réémettent, tout le réseau est couvert. L'intérêt est de minimiser la taille du CDS de manière à limiter le nombre de nœuds qui retransmettent le message. De même pour le routage, diminuer la taille du CDS réduit la complexité de la recherche de route.

Wu et Li proposent un algorithme de marquage fournissant un CDS dans [WL99b]. Les nœuds sont supposés disposer d'une priorité qui peut être fonction de l'identifiant du nœud, du niveau de batterie, du nombre de voisins ou encore aléatoire. La première étape de l'algorithme consiste à marquer les nœuds qui possèdent au moins deux voisins qui ne sont pas directement connectés. Ensuite, deux règles sont successivement appliquées pour réduire la taille du CDS précédemment obtenu. La première règle (Règle 1) consiste à retirer la marque d'un nœud si tous ses voisins sont aussi voisins d'un nœud marqué de plus haute priorité que lui. La deuxième règle (Règle 2) consiste à retirer la marque d'un nœud si tous ses voisins sont aussi voisins de deux nœuds marqués directement connectés et de priorités supérieures à lui.

Dans [DW03], Dai *et al.* présentent une nouvelle règle, appelée Règle k, permettant de diminuer la taille de l'ensemble dominant obtenu. Cette règle consiste à retirer la marque d'un nœud si son voisinage est « couvert » par un ensemble de nœuds connectés, de priorités supérieures à lui. Cette règle diminue de manière plus importante que les règles 1 et 2 le nombre de nœuds dominants.

Dans [CSR04], les auteurs proposent une règle pour déterminer si un nœud n'est pas dominant. Cette proposition est une variante de la Règle k. Un nœud u n'est pas dominant si l'ensemble de ses voisins de priorités supérieures à lui est connecté et « couvre » tous ses voisins.

Ces définitions posées, nous pouvons maintenant nous intéresser aux problèmes que l'on peut rencontrer dans les réseaux ad hoc.

2.3 Un environnement difficile

Répondre aux exigences des applications décrites précédemment pose de nombreux défis, induits par l'aspect distribué et l'utilisation d'interfaces de communication sans fil dans les réseaux ad hoc. Obtenir un environnement de travail d'une qualité équivalente à celle fournie par le monde filaire s'avère très difficile. En effet, de nombreux obstacles vont venir mettre en échec des tentatives d'adaptation naïves des protocoles existants.

Premièrement, la qualité du lien fourni par une connexion sans fil est sans commune mesure avec celui d'une connexion filaire de type Ethernet 802.3 [bP85] par exemple. En plus de la différence évidente de débit (54 Mbits/s pour 802.11g [bP99], 1 Gbits/s voire 10 Gbits/s [bP02] pour le filaire), le plus flagrant est sans nul doute l'écart de stabilité du lien. Là où, en filaire, on atteint sans problème le débit théorique, avec une connexion radio, le débit, mais aussi le délai, la gigue, vont être fortement dépendant de l'environnement. Les matériaux constituant les murs, le temps qu'il fait, les différents appareils parasites vont grandement affecter la qualité du lien radio, sans que cela soit prévisible. De plus, comme nous allons le voir un peu plus loin, l'accès au médium est bien plus compliqué dans le cas d'un lien radio.

Deuxièmement, le caractère distribué du réseau rends inadapté la plupart des protocoles classiques. Si l'on prends l'exemple du routage (indispensable dans un réseau), un simple routage IP ne peut être effectué. Quels objets endosseraient le rôle de routeurs ? De plus, comme les objets utilisent une connexion sans fil, les utilisateurs de ces derniers peuvent se déplacer tout en souhaitant continuer à bénéficier des avantages de la connexion (on peut très bien imaginer une radio qui diffuserait son contenu pour un baladeur par exemple). Il est alors impensable d'utiliser un routage basé sur des tables statiques comme dans le cas d'IP.

Nous allons maintenant aborder, point par point, les techniques qui ont été élaborées afin de répondre aux problèmes de l'accès au médium radio, à la diffusion d'un message dans tout le réseau et au routage.

2.3.1 L'accès au médium

2.3.1.1 Un médium limité

Une interface de communication radio permet aux utilisateurs d'accéder au réseau facilement tout en pouvant être mobile. Néanmoins, cet immense avantage souffre d'inconvénients importants qui limitent le débit disponible pour l'utilisateur. L'utilisation de l'air comme support de communication est soumit à des réglementations strictes qui empêchent d'utiliser n'importe quelle fréquence. Il est donc impossible d'augmenter à volonté le débit afin de répondre à toutes les demandes. Dans le monde filaire, il suffit de rajouter un fil ou une fibre optique pour gagner en débit.

Il existe pourtant un problème commun aux deux mondes. Celui du partage de l'accès au médium. En effet, si deux utilisateurs veulent parler en même temps sur le même médium (que ce dernier soit l'air ou un fil), il se produit une collision. Dans le meilleur des cas, une seule des deux communications va être utilisable. En général, les deux communications seront brouillées et la communication aura été faite pour rien, ce qui diminue le débit (car on a perdu du temps) et dans le cas des périphériques portables, consomme inutilement de l'énergie.

2.3.1.2 Protocoles sans collision

De nombreux protocoles ont été développé afin de fournir un partage équitable du canal et d'éviter les collisions de façon absolue. On peut citer par exemple les réseaux basés les anneaux à jeton (ou *token ring*) [bP98a]. Un jeton circule dans le réseau de poste en poste. Le poste qui possède le jeton peut émettre ses données. Comme il est le seul à posséder le jeton, personne n'émettra en même temps que lui. Une fois la transmission effectuée, le jeton est passé à la station suivante. La Figure 2.4 montre comment est architecturé un tel réseau.

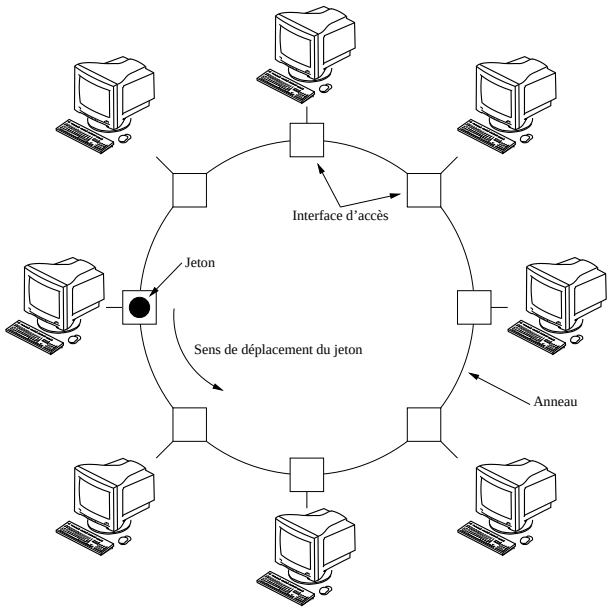


FIG. 2.4 – *Un réseau basé sur un anneau à jeton*

Ce type d'architecture est dit sans collision. C'est à dire que, par le biais d'une synchronisation (ici, le jeton), les postes se partageant le canal collaborent afin de ne produire aucune collision. D'autres exemples d'algorithmes sans collision sont les algorithmes TDMA (*Time Division Multiple Access*) [Meh84] ou FDMA [Tan03, Chapitre 2] (*Frequency Division Multiple Access*). Dans le premier cas, le temps est divisé en intervalles finis et à chaque station est attribué un slot. Une station qui désire émettre le fera uniquement pendant le slot qui lui est alloué. Dans le deuxième cas, à chaque station correspond une fréquence d'émission. Ainsi, les stations peuvent transmettre en même temps sans brouiller leurs émissions. Dans le cas du GSM [Sco96], par exemple, les deux techniques sont mixées. Le canal est divisé en sous fréquences (FDMA), qui sont elles même divisée en intervalles de temps (TDMA).

L'avantage de telles méthodes d'accès est que les collisions sont inexistantes et que le partage du canal est parfaitement équitable. Par contre, il est nécessaire de

synchroniser les stations, ce qui très difficile à faire de façon distribuée. De plus, si certaines stations n'ont rien à transmettre, il y a un gâchis de la bande passante disponible car la partie du canal qui leur est réservée ne peut être utilisée par une autre station.

Afin d'éviter la nécessité d'une telle synchronisation ainsi que la perte d'efficacité existante quand des stations n'ont rien à transmettre, il existe des méthodes de gestion du canal à accès multiple. Ici, une station qui a besoin d'émettre va tenter de le faire immédiatement sans avoir besoin « d'attendre son tour ». Il se peut donc qu'il y ait des collisions et le but des protocoles de gestion d'un canal à accès multiple va être de les minimiser.

2.3.1.3 Protocoles à accès multiple

Dans les années 1970, N. Abramson *et al.* ont imaginé le protocole ALOHA [Abr70]. L'idée de base du protocole est simple : laisser les utilisateurs accéder librement au canal lorsqu'ils ont des données à transmettre. Dans ces conditions, il est clair que des collisions se produisent. En écoutant le canal pendant la transmission, l'utilisateur peut savoir si une collision à eu lieu et, dans ce cas, retransmettre la trame. Afin d'améliorer la capacité d'ALOHA, Roberts propose en 1972 [Rob72] de diviser le temps en slots, ou intervalles finis, chacun d'eux correspondant à une trame. Cette technique nécessite la synchronisation des stations mais réduit la période de vulnérabilité des trames. Néanmoins, le principal problème de la méthode reste qu'une station émet sa trame quand elle le désire, sans se préoccuper de savoir si le canal est libre à ce moment.

Les protocoles reposant sur le fait que le comportement des stations est déterminé par le résultat de l'écoute du canal sont des protocoles à détection de porteuse (*carrier sense protocols*). Kleinrock et Tobagi ont analysé en 1975 [KT75] plusieurs protocoles appartenant à cette famille : CSMA 1-persistant(*Carrier Sens Multiple Access*), CSMA non persistant et CSMA p-persistant. La différence de ces protocoles résidant dans leur réaction face à un canal occupé. Le premier (1-persistant) continue à écouter le canal et transmet dès que celui ci se libère. Le second (non persistant) arrête d'écouter le canal pendant un temps aléatoire avant de tester si celui ci est à nouveau libre. Dans le dernier protocole (p-persistant), la station ne transmet sa trame qu'avec une probabilité p quand le canal est libre. Si le média est occupé, la station procède comme dans le cas non persistant.

Ces protocoles constituent une amélioration d'ALOHA car ils assurent que les stations n'émettrons pas si elles constatent que quelqu'un d'autre le fait déjà. Une autre amélioration consiste à faire en sorte qu'une station qui détecte une collision cesse immédiatement son émission, afin de libérer le canal au plus vite et de gagner du temps. De plus, ceci permet de réémettre la trame perdue. Cet algorithme est connu sous le nom de CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) et est utilisé par Ethernet 802.3 [bP85].

Si deux stations détectent la collision en même temps, elles risquent de retransmettre le paquet en même temps, et donc de générer une nouvelle collision, et ainsi de suite... Pour pallier à ce problème, Ethernet utilise un algorithme d'attente aléatoire (ou stochastique). Les stations qui détectent une collision tirent un temps d'attente aléatoire avant de ré-émettre le message. Ce temps est pris entre 0 et T_{MAX}. Si une collision survient à nouveau, le temps T_{MAX} est doublé. Au plus il y a de collisions, au plus les stations vont attendre avant de réémettre, ce qui permet de diminuer la contention du canal et donc de pouvoir retransmettre avec moins de risques de collision.

2.3.1.4 Accès au lien radio

À l'instar d'Ethernet, il existe des méthodes d'accès aléatoire à une interface radio. Cependant, la technique utilisée (CSMA/CD) n'est pas applicable directement dans le cas d'une interface sans fil. En effet, différents problèmes viennent rendre impossible la mise en place du CSMA/CD. Premièrement, la grande majorité des interfaces ne permettent pas d'écouter le canal pendant l'émission. Les collisions ne sont donc plus détectables. Deuxièmement, certaines situations liées à la portée de communication de l'interface mettent en faute le principe de l'écoute du canal avant émission. Ces deux situations particulières sont appelées problème du terminal caché et problème du terminal exposé.

Problème du terminal caché La situation correspondant à ce problème est illustrée Figure 2.5. Ici, la station A transmet une trame à la station B. La station C veut également transmettre un message à B. Si elle utilise un algorithme de type CSMA, elle va écouter le canal pour savoir si celui ci est libre. Comme la portée de communication de A ne lui permet pas d'atteindre C, ce dernier va penser que le canal est libre, émettre sa trame, et venir brouiller la trame que A est en train d'envoyer à B. On voit donc que C n'a aucun moyen de savoir si A est en train d'émettre car il est « caché » par le manque de portée de communication. S'il émet, on a donc une collision au niveau du récepteur du message.

Problème du terminal exposé Ce problème, illustré Figure 2.6 est l'inverse du précédent. Ici, la station A transmet une trame à D. Pendant ce temps, B désire communiquer avec C et écoute le canal. Il entend A (car il est a porté de communication) et ne transmet donc pas sa trame à C. Hors, cette transmission ne brouillerais pas le mobile D car il n'est pas a portée de B. Le débit utile est donc diminué car B n'émet pas alors qu'il aurait pu le faire.

Protocoles MACA et MACAW Un des premiers protocoles conçus pour les réseaux locaux sans fil est MACA [Kar90] (*Multiple Access with Collision*

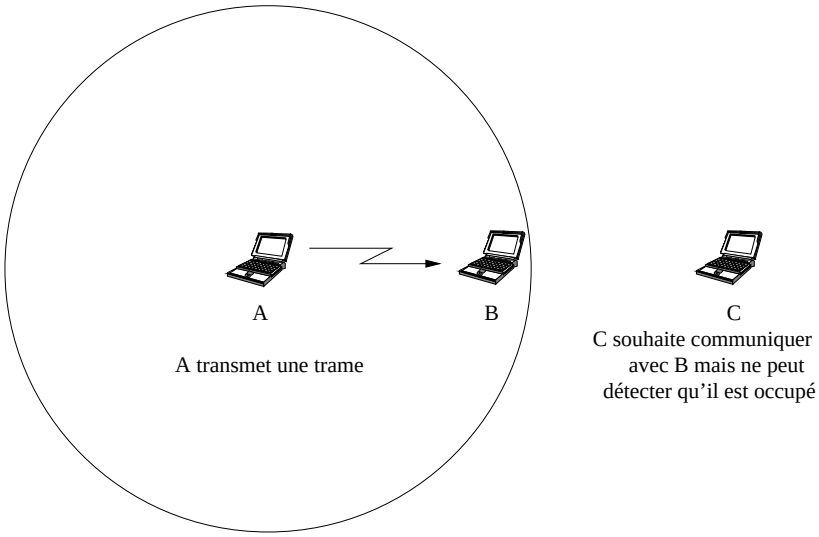


FIG. 2.5 – *Problème du terminal caché*

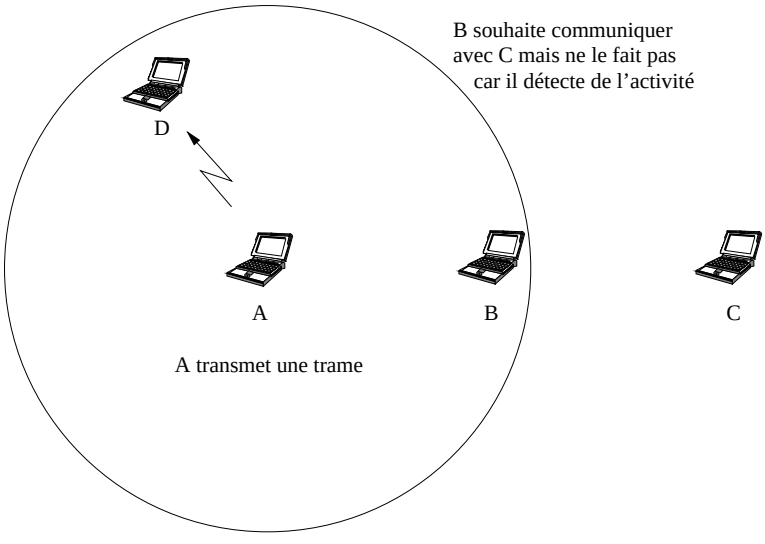
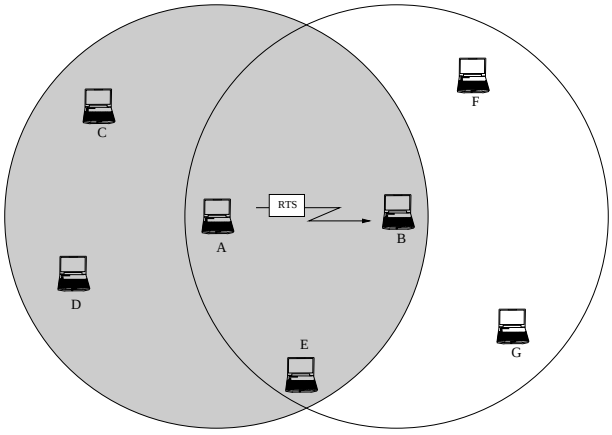
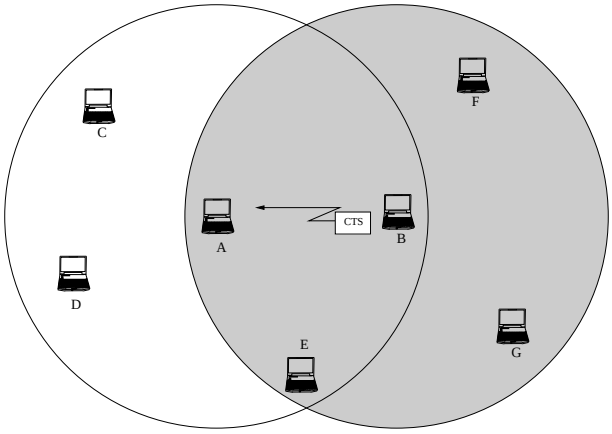


FIG. 2.6 – *Problème du terminal exposé*

Avoidance). L'idée de base de l'algorithme et de tenir au courant les stations voisines de l'émetteur et du récepteur du désir d'effectuer une transmission. Pour cela, l'émetteur envoie une petite trame (qui a donc peu de risque de générer une collision) afin de prévenir ses voisins ainsi que le récepteur. Le récepteur répond à cette trame, nommée le RTS (*Ready To Send*) par une deuxième, le CTS (*Clear To Send*). Toutes les stations ayant reçu le RTS ou le CTS s'interdisent alors de transmettre des données pendant un temps précisé dans le RTS et le CTS (qui dépends de la taille de la trame à transmettre). Cette procédure est illustrée Figure 2.7 où A envoie une trame à B. Grâce au RTS, C, D et E ne vont pas brouiller la transmission et grâce au CTS, F, G et E ne vont pas brouiller non plus. Cet échange de trames permet donc de gérer le problème du terminal caché illustré Figure 2.5.



(a) A envoie d'abord un RTS à B



(b) B répond par un CTS

FIG. 2.7 – *Le protocole MACA*

Après analyse des défauts du protocole, Bharghavan *et al.* l'ont optimisé pour

donner naissance au protocole MACAW (MACA *for Wireless*) [BDSZ94]. En plus de reprendre le principe des trames RTS et CTS, les auteurs ont d'abord observé que sans acquittement dans la couche liaison, les stations ne détectent qu'une trame n'a pas été transmise que très tard (à l'aide d'un acquittement dans la couche transport par exemple) menant ainsi à de mauvaises performances. Ils ont donc ajouté l'envoi d'une trame d'acquittement (ACK) après la trame de donnée. Ils décidèrent également d'utiliser CSMA afin d'empêcher qu'une trame RTS ou CTS entre en conflit avec une autre communication et de gérer le principe d'attente stochastique en cas d'échec de transmission par couple de stations et non pas par station afin de rendre le protocole plus équitable.

Couche MAC de 802.11b Le principal produit commercial disponible est le WiFi qui implémente la norme IEEE 802.11b [bP99]. En ce qui concerne la couche MAC, la norme présente deux modes de fonctionnement. Le premier est appelé **fonction de coordination distribué** ou **DCF** (*Distributed Coordinated Function*) qui ne fait pas appel à une entité de contrôle centralisée (et peut donc être comparé avec Ethernet). Le second, appelé **fonction de coordination par point d'accès**, ou **PCF** (*Point Coordination Function*) est utilisé dans le cas ou un point d'accès est disponible. Ce dernier mode est facultatif mais toutes les implémentations se doivent d'accepter le mode DCF. Dans le mode PCF, le point d'accès demande aux stations si elles ont quelque chose à transmettre et ces dernières ne peuvent le faire qu'à l'invite du point d'accès. Il n'y a donc pas de collisions dans ce mode. Il est bien évident que ce mode est inintéressant pour les réseaux ad hoc.

Le mode DCF utilise le protocole CSMA avec évitement de collisions, ou CSMA/CA (*CSMA with Collision Avoidance*). Ce protocole se divise également en deux utilisations distinctes, la diffusion ou le point à point. Dans le premier cas, la station veut émettre une trame à destination de tous ses voisins. Elle commence par attendre un temps minimum, appelé DIFS (*DCF Inter-Frame Spacing*) afin de laisser aux autres stations une chance d'acquérir le canal. Puis, elle tire un nombre aléatoire représentant le temps pendant lequel le canal doit rester libre avant qu'elle puisse émettre son message. Si, après ce temps, le canal est toujours libre, la station peut émettre sa trame. Dans le second cas (la transmission point à point), le protocole CSMA/CA se base sur MACAW et utilise le même principe d'écoute du canal que dans le cas de la diffusion plus le mécanisme de RTS, CTS et d'ACK.

Bluetooth Ce système à été développé à l'origine par Ericsson, IBM, Intel, Nokia et Toshiba dans le but de connecter les téléphones portables à d'autres équipements (des assistants personnels ou des oreillettes par exemple). Bien que le but premier était simplement de supprimer les câbles entres les équipements, le

succès de la norme l'amena à être utilisée dans le cadre de LAN sans fil. Le principe général de l'accès au médium est un système de type TDMA où un maître donne les consignes aux esclaves en leur attribuant des slots de temps. Ceci explique le débit très faible (1 Mbits/s) des équipements Bluetooth. Ce faible débit ainsi que le mode de fonctionnement maître/esclave rend Bluetooth peu adapté au contexte qui nous intéresse ici. C'est pourquoi nous n'en ferons plus allusion à partir de maintenant.

2.3.2 Le routage

Le routage est en quelque sorte le mécanisme clé des réseaux ad hoc. C'est grâce au mécanisme de routage que les stations formant le réseau vont pouvoir communiquer, même si elles ne sont pas à portée directe de communication. Il est donc très important d'avoir un protocole de routage efficace si on veut pouvoir tirer parti du potentiel des réseaux ad hoc. Néanmoins, si dans le cas des réseaux filaires comme Internet, le routage est « facile », il en va tout autrement des réseaux ad hoc. En effet, alors que dans le monde filaire les liaisons inter-routeurs sont connues, statiques et peuvent donc être stockées dans des tables de routage qui sont très rarement modifiées, dans les réseaux ad hoc, la mobilité va entraîner la nécessité d'une constante modification de ces tables. De plus, dans les réseaux ad hoc, il faut que toutes les stations soient des routeurs car, sinon, une station pourrait ne pas avoir de routeur dans son voisinage et donc ne pas être en mesure de communiquer avec le reste du réseau.

Si une station A veut joindre une station B, sur Internet, la route qu'emprunteront les paquets est connue au début de la communication et restera la même pendant toute la durée de cette dernière. Dans un réseau ad hoc, la station A ne sait pas, a priori, où se trouve B et même si elle le trouve, rien ne prouve que B sera toujours au même endroit quelques secondes plus tard. Les stations ne peuvent donc s'appuyer sur des informations statiques et vont devoir en permanence tenter d'obtenir des informations récentes sur la ou les routes à emprunter pour joindre une destination donnée. Les deux opérations vitales du routage dans un réseau ad hoc seront donc la recherche de routes mais également la maintenance de ces routes car durant la communication, la mobilité va faire qu'il faudra sûrement changer de relais entre la source et la destination.

Enfin, toujours à cause de la mobilité, il se peut qu'aucune route ne soit disponible entre deux stations et donc que ces stations se retrouvent physiquement déconnectées. Contrairement au cas d'un réseau filaire où une déconnexion constitue une réelle panne, ceci doit être considéré comme un événement normal dans un réseau ad hoc.

Il existent trois grandes familles de protocoles de routage ad hoc. Les protocoles réactifs, proactifs et hybrides. Nous allons maintenant en décrire quelques

uns parmi les plus connus. Le lecteur avide de précisions est invité à se référer à [Fee99, BMJ+98, RS96, Jai00, RT99, Joh94] pour de plus amples informations sur les problèmes liés au routage ainsi que sur des comparaisons et analyses des différents protocoles existants. Le panorama que nous faisons omet volontairement les algorithmes de routages géographiques que nous allons aborder pour les besoins du Chapitre 3.

2.3.2.1 Algorithmes proactifs

Ce type d'algorithme consiste à maintenir en permanence des informations qui serviront à obtenir une route vers une station donnée. Le principe peut s'apparenter au protocole de routage Bellman-Ford [FF93] utilisé dans la mise à jour des routeurs sur Internet. Chaque station du réseau va maintenir une ou plusieurs tables qui lui permettront de savoir à quels voisins transmettre un paquet en fonction du destinataire recherché. Quand la topologie du réseau est modifiée, les stations diffusent les modifications qu'elles perçoivent afin de mettre à jour les informations contenues dans les tables. Il existent plusieurs algorithmes de ce type, qui se différencient par les informations utilisées dans les tables ainsi que les méthodes de mises à jour de ces tables.

Le protocole DSDV [PB94] (*Destination Sequenced Distance Vector*) par Perkins *et al.* maintient une table contenant, pour chaque destination, le voisin à joindre pour y parvenir, ainsi que le nombre de sauts et un numéro de séquence. L'algorithme considère une route R_1 comme meilleure qu'une route R_2 si elle a un numéro de séquence plus important (*i.e.*, la route est plus récente donc les informations ont plus de chances d'être correctes) ou si le numéro de séquence est le même mais que la distance estimée vers la destination en empruntant la route R_1 est plus faible. Chaque station émet sa table de routage régulièrement en incrémentant le numéro de séquence correspondant à la route menant à lui. Quand une station estime que sa route vers une autre est coupée, elle met à jour sa table de routage avec une distance infinie. Afin de diffuser les mises à jour de la table, la station peut émettre deux types de paquets: la table complète (*full dump*) ou uniquement les parties de la table qui ont été modifiées (*incremental*) afin de réduire l'utilisation du médium.

Dans [MGLA95], Murthy *et al.* proposent le protocole WRP (*Wireless Routing Protocol*). Le principe est sensiblement identique à DSDV mais ils utilisent, en plus des informations de distance et d'âge des informations, la notion de coût de route (en terme de latence entre une station et les différents destinataires). Chaque station émet périodiquement les changements de sa table de routage ainsi que des demandes de confirmation de présence à ses voisins afin de les informer des changements topologiques et de vérifier la validité de son voisinage.

Le protocole CSGR (*Cluster Switch Gateway Routing*) présenté par Chiang *et*

al. s'articule autour d'une architecture basée sur un regroupement des stations en *clusters*. Chacun de ces *cluster* possède un chef qui se charge des communications à l'intérieur de son propre *cluster* et maintient les informations de routage lui permettant de joindre les chefs des autres *clusters*. Cette approche trouve son intérêt dans le cas où les stations d'un même *cluster* se déplacent peu les unes par rapport aux autres, car il y a peu de changement dans la topologie du *cluster*. Par exemple, un tel protocole de routage peut être parfaitement adapté à une situation de PAN (*Personnal Area Network*). Cependant, si les *clusters* sont amenés à être modifiés trop fréquemment à cause de la mobilité, cette approche s'avère être trop coûteuse pour la mise à jour des informations.

Un des protocoles proactifs les plus intéressants à l'heure actuelle est sans nul doute OLSR [CJL+01], protocole issu de l'INRIA qui est maintenant standardisé par l'IETF [CJ03]. Cet algorithme est conçu pour minimiser le coût associé aux messages de contrôle. Chaque station calcule régulièrement le sous-ensemble MPR (*MultiPoint Relaying*) [LQV01] de ses voisins. L'ensemble MPR est l'ensemble des voisins nécessaires pour joindre les voisins à deux sauts. Ainsi, lors de la diffusion des messages de contrôle qui construisent les tables de routages, seuls les stations appartenant à l'ensemble MPR vont être utilisés pour relayer les messages (ainsi que pour construire les routes).

L'intérêt principal des algorithmes proactifs et que l'on peut trouver facilement et rapidement le destinataire sans avoir à lancer une recherche dans le réseau. De plus, les informations collectées pour aider au routage peuvent s'avérer très utiles pour d'autres applications et, comme les informations sont mises à jour régulièrement, les pertes de routes sont peu fréquentes.

Cependant, ces algorithmes sont tributaires d'une mise à jour des informations régulière et fiable, ce qui induit une charge constante du réseau, due aux messages de contrôle. Dans le cas de réseaux à mobilité forte, cette charge se révèle être une catastrophe car la quasi totalité de la bande passante est consacrée aux messages de contrôle et les applications n'ont plus assez de ressources. De plus, si le réseau est très grand, la quantité d'information à diffuser et à mémoriser devient également un problème.

2.3.2.2 Algorithmes réactifs

Les algorithmes réactifs suivent une politique radicalement opposée à celle des algorithmes proactifs. Ici, aucune maintenance régulière n'est effectuée et la recherche de routes est effectuée « à la demande ». Lorsqu'une station cherche à en joindre une autre, elle utilise un protocole de diffusion (envoi d'un message à tout le réseau) afin de découvrir une route permettant d'y mener. Quand le correspondant reçoit le message de diffusion, dans lequel est stocké la liste des stations qui ont été utilisées pour acheminer ce message, il peut répondre par un

message qui suivra cette route afin d'informer la source. La Figure 2.8 donne un exemple simple d'une telle procédure.

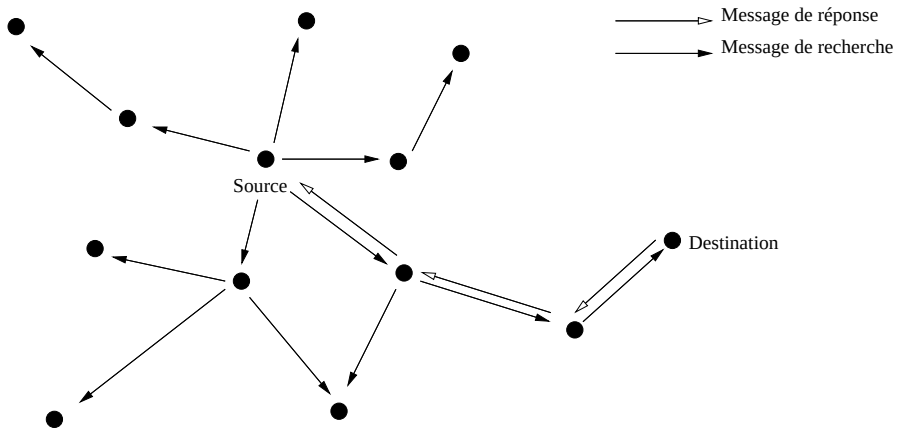


FIG. 2.8 – *Recherche de route par un protocole réactif*

L'algorithme réactif le plus connu (dont diverses implémentations ont été effectuées tant pour Linux que pour Windows) est AODV (*Ad Hoc On-demand Distance Vector*) [PR99, PBRD03]. Dans cet algorithme, le chemin parcouru par le paquet de recherche n'est pas conservé dans le paquet afin de minimiser la taille de ce dernier. Chaque recherche est identifiée par un numéro de séquence et les relais utilisés lors de la recherche de la route mémorisent la station précédente. Lors de la réponse, chaque station valide le chemin dont elle est relais.

Le protocole DSR (*Dynamic Source Routing*) [JM96] proposé par Johnson et Maltz est très similaire à AODV. La différence principale est située au niveau du paquet de recherche et de réponse dans lesquels, contrairement à AODV, le chemin à suivre est mémorisé. De plus, ce protocole gère les liens asymétriques (situation qui correspond à une différence de puissance d'émission entre deux stations A et B qui implique que B peut entendre A mais pas réciproquement).

D'autres protocoles, tels que SSR (*Signal Stability Routing*) [DRWT97] ou ABR (*Associativity-Based long-lived Routing*) [Toh97] introduisent la notion de fiabilité de routes (respectivement en tenant compte de la qualité du signal en réception ou des changements topologiques du voisinage). Lors de la construction de la route, la diffusion est biaisée de façon à suivre les liens qui ont un haut degré de fiabilité.

Cette classe d'algorithmes s'avère efficace dans les réseaux de taille importante et/ou à forte mobilité. En effet, comme les routes sont construites à la demande, on évite une charge constante et importante du réseau due aux changements de topologie. Par contre, si l'algorithme utilisé lors de l'étape de diffusion est inefficace, les performances de ces algorithmes peuvent devenir extrêmement mauvaises.

2.3.2.3 Algorithmes hybrides

Le principe des algorithmes hybrides est de combiner les deux approches afin de tenter de tirer parti des avantages des deux principes, tout en compensant leurs inconvénients.

L'algorithme TORA (*Temporally-Ordered Routing Algorithm*) [PC97] consiste à établir un graphe acyclique orienté dont la racine est la destination (*i.e.*, le nœud destination est le seul sans arc sortant). Ainsi, depuis chaque station, on peut retrouver la destination en suivant l'orientation du graphe. Cet algorithme est un algorithme distribué dont une version tourne sur chaque station et pour chaque destination. Le routage peut être vu comme un écoulement de fluide dans des tubes. Chaque tube représente une liaison entre deux nœuds, la jonction entre deux tubes est un nœud et le fluide qui s'écoule représente les paquets. Chaque nœud à pour attribut une hauteur et le fluide va s'écouler du point le plus haut au point le plus bas. S'il est bloqué à un nœud, la hauteur de ce dernier est augmentée telle qu'elle soit supérieure à celle de ses voisins et que le liquide puisse continuer sa course. Quand un nœud a besoin d'une route vers une destination, il émet un paquet QUERY contenant l'adresse de la destination. Ce paquet se propage dans le réseau jusqu'à arriver à un nœud voisin du nœud de destination. Celui-ci émet alors un paquet UPDATE contenant sa hauteur (la hauteur de la destination étant 0, la sienne sera 1). Quand un nœud reçoit le paquet UPDATE, il s'attribue une hauteur supérieure à celle contenue dans le paquet ce qui a pour effet d'orienter les arcs du graphe de façon à avoir une route vers le destinataire depuis tous les points du réseau. Quand un paquet de données doit être émis, il va suivre le graphe grâce à la différence entre sa hauteur et celle de ses voisins. Quand un nœud découvre que la route vers une destination n'est plus valide, il réajuste sa hauteur et transmet un paquet UPDATE pour mettre à jour le graphe. La connaissance des arcs descendants (*i.e.*, de la hauteur des voisins) est obtenue par diffusion régulière de l'information de hauteur par les nœuds. Cet algorithme est donc principalement réactif (car il construit la route à la demande) mais possède une partie proactive (la mise à jour des arcs) qui permet d'optimiser cette recherche. La Figure 2.9 montre les différentes étapes de la recherche d'une route. Le couple $(H_i; I_i)$ de chaque nœud représente la hauteur (H_i) et l'identifiant (I_i) de celui-ci.

Le protocole ZRP (*Zone Routing Protocol*) [HP98] utilise un protocole proactif pour joindre les stations situées à une distance inférieure à k sauts et un protocole réactif pour le routage entre les groupes (ou *routing zones*). Chaque station connaît (grâce à la partie proactive de l'algorithme) comment joindre les autres membres de son groupe. Si elle doit joindre quelqu'un n'appartenant pas à ce groupe, elle utilise le protocole IERP (*Interzone Routing Protocole*) [HPS01] qui envoie une demande aux stations situées en bordure du groupe. Ces dernières transmettent le message aux bordures des zones voisines et si la destination ap-

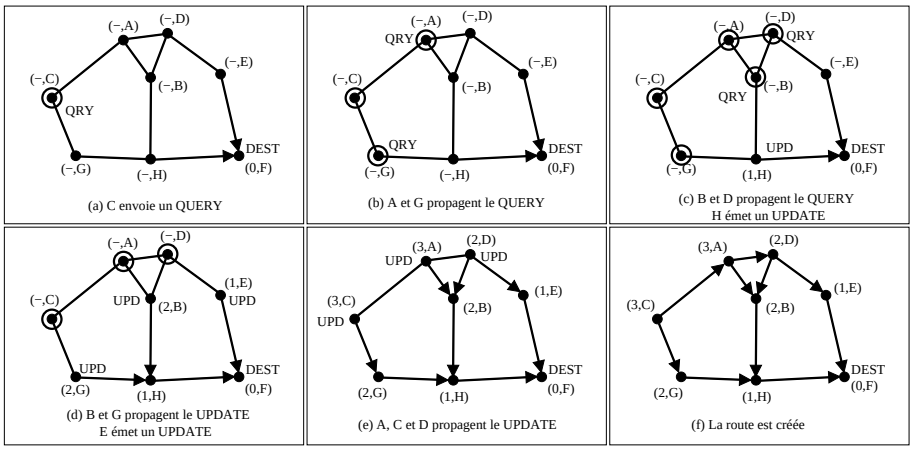


FIG. 2.9 – *Création d'une route par l'algorithme TORA*

partient à une de ces zones, la route utilise à nouveau le protocole proactif à l'intérieur de cette dernière pour joindre la destination.

Comme nous avons pu le constater lors de la description des protocoles (tant réactifs que proactifs ou hybrides), il est très souvent question d'algorithme de diffusion. La qualité de ces algorithmes peut grandement modifier les performances du routage et ils forment donc un domaine d'étude primordial pour l'efficacité des algorithmes de routages. De plus (et nous le verrons par la suite quand nous exposerons nos contributions Chapitres 3 et 4) de tels algorithmes peuvent également être très utiles dans le domaine applicatif et plus particulièrement, en ce qui nous concerne, pour la gestion de services.

2.3.3 Diffusion

La diffusion est l'opération d'envoyer un message à tous les membres du réseau. De nombreux algorithmes ont été mis au point afin d'effectuer cette opération en essayant de limiter au maximum son coût. La qualité d'un algorithme de diffusion va dépendre de différents critères :

- sa fiabilité ou sa capacité à joindre effectivement toutes les stations du réseau. Un protocole fiable doit permettre de joindre la totalité du réseau si celui-ci est connexe ;
- la consommation énergétique. Si on se place dans un contexte où l'énergie des stations est limitée (ce qui sera à priori le cas car elles utilisent des batteries), il est important de consommer le moins possible. Cette consommation sera fréquemment fonction du nombre d'émissions du message de diffusion.
- la latence moyenne. Ceci représente le délai moyen entre le début de la

diffusion et la dernière réception du message de diffusion ;

L'algorithme le plus simple est l'inondation aveugle (*blind flooding*). Son fonctionnement est trivial : chaque station retransmet le message de diffusion si elle le reçoit pour la première fois. L'implémentation est également très simple, la source insère dans le message de diffusion un numéro de séquence (*sequence number*) qui doit être unique pour chaque diffusion [4]. Chaque mobile qui reçoit le message de diffusion recherche le numéro de séquence de ce dernier dans une table. Si il n'est pas présent, il retransmet le message et insère le numéro de séquence dans la table. Sinon, le mobile ignore tout simplement le message car il l'a déjà reçu auparavant. Bien que la simplicité de l'algorithme peut le rendre attractif au premier abord, sa naïveté le rends extrêmement coûteux et donc difficilement utilisable si on veut obtenir des performances raisonnables. En effet, toutes les stations vont émettre au moins une fois le message ce qui, le plus souvent, s'avère inutile. Même avec une densité de réseau faible, il est évident que de nombreux mobiles vont avoir des voisins communs qui, si chaque mobile émet le message diffusion, vont recevoir plusieurs fois la même information. De plus, comme tous les voisins d'un mobile vont tenter de retransmettre en même temps, la congestion engendré au niveau MAC risque fort de devenir catastrophique ! Ce problème appelé *Broadcast Storm Problem* à été analysé par Tseng *et al.* [TNC02]. Ils proposent différentes modifications pouvant être apportées à l'algorithme de base permettant de diminuer les impacts négatifs. Sans rentrer dans les détails – le lecteur pourra se référer à [TNC02] pour de plus amples informations – on peut citer les cinq modifications proposées :

- diffusion probabiliste. Chaque mobile à une probabilité P, fixée au départ de retransmettre le message ;
- schéma fondé sur le comptage. Un mobile ne retransmet pas le message s'il l'a déjà reçu plus de C fois (également fixé au départ) ;
- schéma fondé sur la distance. Un mobile de retransmet pas le message s'il le reçoit d'un autre situé à une distance inférieure à D (toujours fixée au départ) ;
- schéma fondé sur la position. Grâce à des informations de positionnement fournies par un système de type GPS, un mobile ne retransmet le message que s'il peut joindre un nombre suffisant de voisins situés plus loin que lui de la source ;
- schéma fondé sur les groupes. Le réseau est organisé de façon hiérarchique en plusieurs groupes et seuls certains mobiles appartenant à un groupe retransmettent le message.

Ces travaux ont fourni une excellente base pour les travaux sur la diffusion et de nombreuses approches ont ensuite été proposées. Le lecteur pourra trou-

4. Un tel résultat est très facile à obtenir en mixant l'identifiant de la source et un numéro qu'elle incrémente à chaque diffusion différente.

ver dans [LG97, WL99a, LQV01, CIS03, CSS03] nombres de protocoles efficaces. Cette liste, non exhaustive, présentes des méthodes permettant d'améliorer l'efficacité, de diminuer les coûts (tant en diffusion qu'en énergie), basées sur différents schémas inspirés par ceux donnés précédemment.

Comme nous venons de le voir, de nombreux travaux se focalisent sur la proposition de nouveaux protocoles pour remplir les fonctions des couches « basses » du modèle OSI [Tan03, Chapitre 1, Section 4]. Mais, si le routage, l'accès MAC ou la diffusion constituent des protocoles vitaux, il est évident qu'il faut également se pencher sur le problème applicatif. En effet, ce sont les applications qui créent le besoin d'avoir un réseau fonctionnel. La fin de ce chapitre va se consacrer à la présentation de la notion de service ainsi qu'aux différents moyens qui ont été mis en œuvre pour faciliter la conception et l'utilisation de ces services dans un réseau. Nous présenterons enfin les besoins de la gestion de service qui nous ont poussé à mener les travaux présentés dans ce document.

2.4 Vers une gestion de service mobile et distribuée

Les réseaux ad hoc semblent très prometteurs. Afin de profiter de leur potentiel, il faut proposer des solutions logicielles adaptées, prenant en compte l'aspect dynamique et distribué, intrinsèque à ce type de réseau. Nous allons tout d'abord présenter la notion de service et donner un éventail des différentes catégories possibles. Puis nous expliquerons pourquoi, dans le cadre de la gestion de services, les solutions existantes dans le monde filaire ne sont pas adaptées.

2.4.1 Qu'est ce qu'un service ?

Le terme « service » peut être défini comme étant « *une fonction d'utilité commune, publique et l'activité organisée qui la remplit* » [Mer97]. De nombreux exemples de services peuvent être trouvés dans le monde réel. On peut citer, par exemple, le téléphone, le courrier, la presse, la télévision...

Dans le monde informatique, un service offre un ensemble de fonctionnalités qui peuvent être de fournir l'accès à des informations, des traitements sur des informations, des possibilités de dialogue avec d'autres utilisateurs ou d'autres programmes.

D'une manière générale, il est possible de noter les différents acteurs du fonctionnement d'un service :

– le ou les utilisateurs : ce sont eux qui justifient la création et le déploiement d'un service. En fonction du service, ces utilisateurs peuvent être nombreux, interagir entre eux ou uniquement avec le service ;

– l'infrastructure de communication : elle permet au service d'être utilisé ;
– les ressources utilisées et fournies par le service : c'est à partir de ces ressources que le service va offrir ses fonctionnalités.

De même, selon la nature de leur fonction, on peut distinguer plusieurs catégories de services :

– les services permettant le bon fonctionnement du réseau;
– les services fournissant des moyens de communications entre les utilisateurs ;
– les services qui donnent accès à des informations ou des données ;
– les services qui donnent accès à des applications.

2.4.1.1 Les services du réseau

Comme nous l'avons vu, une infrastructure réseau est nécessaire au bon fonctionnement d'un service. Cependant, pour que ce réseau fonctionne, ce dernier doit aussi se reposer sur des services.

L'exemple le plus parlant et à coup sûr le plus important quel que soit le type du réseau sera la transformation d'adresse ou le service de nommage. Par exemple, si on considère un réseau de type Ethernet [bP85] relié à Internet, cette transformation va intervenir à deux niveaux.

Tout d'abord, pour accéder à une machine, il faut connaître son adresse IP. Dans le cas d'IPv4 [oSC81], connaître l'adresse d'une machine est à la rigueur assez facile mais si on considère IPv6 [DH98] cela devient beaucoup plus difficile. Afin de faciliter le processus, Internet utilise un système de transformation de nom symbolique qui permet de transformer un nom facile à retenir (par exemple *kwak.lifl.fr*) à une adresse IPv4 ou IPv6. Ce service est appelé *Domain Naming Service* (DNS) [Moc87] et maintient des associations entre noms symboliques et adresses IP de façon distribuée. L'espace de nommage est divisé en *domaine* et *sous-domaine* pour décentraliser l'information. Quand une application veut connaître l'adresse IP de la machine *kwak.lifl.fr*, elle interroge tout d'abord un des serveurs racines qui lui donne l'adresse du serveur gérant le domaine *.fr*. Elle interroge de la même façon ce serveur qui lui donne alors le serveur gérant le sous-domaine *lifl.fr*. Enfin, elle peut interroger ce dernier pour obtenir l'adresse de la machine *kwak.lifl.fr*.

Après le routage du paquet effectué (c'est à dire une fois que celui-ci est parvenu au réseau local de la machine recherchée), il faut encore transformer l'adresse IP en l'adresse correspondant au réseau physique utilisé. Dans le cas d'Ethernet, ce rôle est joué par un service nommé *Address Resolution Protocol* (ARP) [Plu82].

Le but de ces services est donc de permettre de communiquer sur le réseau : localiser les sites, transporter les données de manière fiable et efficace. Cependant,

même si ces services sont primordiaux, ils sont, pour la plupart, complètement masqués aux utilisateurs. Les autres catégories de services seront plus des services dont les utilisateurs ont conscience.

2.4.1.2 La communications entre utilisateurs

Le but de ces services sera de permettre aux utilisateurs de s'affranchir des distances et de communiquer soit en temps réel soit en différé. On parlera alors respectivement de communication *synchrone* ou *asynchrone*.

Les services de communication synchrone les plus utilisés sont sans conteste le chat (IRC) et les messagerie instantanée de type MSN ou ICQ. On peut également citer les audio- et visioconférences.

Pour ce qui est des communications asynchrones, la présence simultanée des différents intervenants n'est pas obligatoire. Le principal représentant de ce type de service de communication est la messagerie électronique [Ros93] qui permet aux utilisateurs de recevoir et d'envoyer des messages par un réseau local ou dans le monde par Internet.

Dans le cas de la messagerie Internet, l'envoi des messages se fait par coopération de plusieurs serveurs grâce au protocole SMTP (*Simple Mail Transfert Protocol*) [Pos82]. L'espace des utilisateurs (à l'instar de l'espace de nommage DNS) est divisé en domaines. Chaque domaine possède un serveur SMTP qui gère une liste d'utilisateurs. Grâce aux protocoles POP [MR96] ou IMAP [Cri03], par exemple, les utilisateurs peuvent consulter leur boîte aux lettres et récupérer leurs messages quelle que soit leur position géographique.

Un autre exemple très répandu de service de communication asynchrone est celui fourni par les serveurs respectant le protocole NNTP [KL86]. Ce service est orienté communication de groupes et permet par le biais de forums de discussions (ou *newsgroup*) de dialoguer, d'échanger des idées ou de fournir un support de travail.

Grâce aux nombreuses normes régissant le fonctionnement des services que nous venons de citer, de nombreux logiciels clients ont pu être développé afin de répondre aux exigences des utilisateurs qui sont aussi différentes que nombreuses. On peut citer les développeurs de client de messagerie qui se livrent une bataille farouche afin de fournir le meilleur accès au service à l'utilisateur (avec des logiciels comme Outlook, Eudora, Thunderbird...).

2.4.1.3 L'accès à l'information ou aux données

Un autre type de service important est celui concernant les services offrant de l'information ou des données. Avec l'essor des réseaux, les utilisateurs ont rapidement ressenti le besoin de transférer leurs fichiers, par exemple pour les partager avec d'autres utilisateurs. Un des premiers services qui est venu répondre

à cette demande fut FTP (*File Transfert Protocol*) [PR85] qui, par une série de commande simple, permet de transférer des fichiers textes ou binaires entre un serveur et la station de l'utilisateur. Pour plus de transparence, et pour répondre aux besoins d'entreprises désireuses de faciliter la centralisation et la sauvegarde des fichiers des employés, NFS (*Network File System*) [Mic89] à été développé dans le but d'uniformiser l'accès à un système de fichier distant avec l'accès à un système de fichier local. Ce service est encore couramment utilisé aujourd'hui.

Avec l'augmentation de la taille des données à transférer, les services de transfert de fichiers distribués ont commencé à voir le jour. Aujourd'hui, de nombreux systèmes, appelés réseaux pair-à-pair (*peer-to-peer*), permettent d'échanger un très grand nombre de fichiers de taille importante sans avoir besoin de posséder un serveur muni d'une connexion à très haut débit qui est très coûteuse. Un des premiers service de ce type fut *Napster* qui a ensuite été suivi par *Kazaa*, *eDonkey* ou encore plus récemment *Bittorrent*. Le principe de ces services est de ne pas télécharger son fichier sur un serveur unique mais d'en prendre plusieurs parties chez des utilisateurs différents. Ainsi, chaque utilisateur télécharge les fichiers qu'il désire tout en permettant aux autres d'accéder à ceux qu'il partage. De nombreux projets de recherche se sont d'ailleurs concentré sur l'amélioration des algorithmes de recherche dans ce type de réseau [SMK+01, RD01, LNBK02].

Le partage d'information a été initiée par le développement de Gopher [AML+93]. Ce système présente l'information sous forme d'une arborescence de menu dont chaque entrée pointe vers un autre menu, un document ou un autre site Gopher. L'utilisateur peut naviguer grâce à une interface graphique qui transmet les requêtes de l'utilisateur au serveur Gopher.

Le successeur de Gopher est bien entendu le World Wide Web [BL90] que nous utilisons tous les jours. Dans ce système, le navigateur communique avec les serveurs WWW par l'intermédiaire du protocole HTTP (*Hyper Text Transfert Protocol*) [FGM+99]. Les documents sont codés avec le format MIME [FB96] et désigné par des URL [BLMM94].

Pour pouvoir retrouver une information, des services d'annuaires et de recherches par mots clés sont disponibles. Les services comme Veronica (utilisé dans Gopher) ou plus récemment Google permettent de trouver toute sorte d'information parmi l'immense quantité disponible sur le réseau. Ces services peuvent être très complexes et mettre en œuvre de nombreux résultats de recherche [GGL03, BP98b].

Encore une fois, les services appartenant à cette catégorie sont pour la plupart définis par des normes strictes, facilitant le développement de plusieurs programmes fournissant le même service et donc, permettant un déploiement large (par exemple, pour les serveur web, Apache et IIS).

2.4.1.4 Les services applicatifs

Ce type de service constitue le type le plus évolué. Ils offrent aux utilisateurs des possibilités d'effectuer des achats, de réserver des billets d'avion de contrôler des machines à distance, etc...

Le premier exemple français de ce type de service fut le Minitel qui, grâce à un terminal spécialisé, permettait d'accéder (et permet toujours d'ailleurs, même si le Web tend à remplacer ce terminal préhistorique) à un grand nombre d'applications comme des services bancaires par exemple.

Grâce à l'exécution d'applications dont l'entrée et la sortie sont « redirigées » vers une page HTML au moyen de scripts CGI (*Common Gateway Interface*) [CR99], les services applicatifs basés sur le Web se développent de plus en plus.

Cependant, contrairement aux types de services présentés dans les sections précédentes, aucune norme n'est disponible quant au développement de ces applications. Les programmeurs doivent en général réaliser leur propre gestion de l'interaction entre leur service et les utilisateurs ou les autres services.

Pour pallier à ces problèmes des couches logicielles intermédiaires permettant de développer facilement des applications distribuées ont été développées et pour certaines, standardisées. Les standards les plus connus de ces « middleware » [Ber96] sont les bus à objets répartis CORBA (*Common Object Request Broker*) [Obj04] de l'OMG (*Object Management Group*), DCOM [Mic95] de Microsoft et Java/RMI [Mic98] de Sun.

Le principe sous-jacent de ces architectures et d'uniformiser l'appel à des méthodes sur des objets, que ces derniers soit hébergés par la machine locale ou un hôte distant. De plus, l'architecture de ces middleware permet également de masquer l'hétérogénéité du matériel, du système d'exploitation et même (sauf pour Java/RMI) du langage de programmation. Une application développée en C++, qui tourne sur une plate-forme Windows pourra appeler des méthodes sur un objet programmé en Java, hébergé par une machine sous Unix de façon complètement transparente. Grâce à ces abstractions, l'écriture de service ainsi que l'interopérabilité de ces derniers est simple à mettre en œuvre. Grâce à des serveurs de types pages jaunes, une application pourra même, si elle en a besoin, demander à obtenir une référence sur un objet fournissant le service qu'elle désire (par exemple, un objet qui peut effectuer un chiffrement) de façon dynamique.

2.4.2 Notre problématique

Quel que soit le type du service que l'on veut utiliser, son utilisation va obligatoirement passer par un point clé. En effet, le minimum nécessaire pour utiliser un service est de pouvoir découvrir quel objet du réseau le fournit. Dans le monde filaire, la solution est simple à trouver. Il suffit de disposer de serveurs qui re-

censent les services disponibles. Comme ses serveurs sont toujours accessibles, il suffit de les interroger pour obtenir l'adresse de l'hôte qui fournit le service que l'on cherche. Pour fournir un service, il est également facile de le publier auprès d'un de ces serveurs afin de le rendre disponible au reste du réseau.

Dans les réseaux ad hoc, une telle centralisation des données est inadaptée. En effet, les serveurs qui recensent les services peuvent être inaccessibles à cause de la mobilité. Si on considère que le réseau est uniquement constitué d'entités mobiles, donc de petite taille, il est peu probable qu'il existe dans le réseau une machine possédant les capacités physiques en terme de mémoire, d'énergie et de bande passante pour se permettre de stocker tous les services disponibles et répondre aux requêtes de tous les autres membres du réseau. Il convient donc de proposer des méthodes permettant de distribuer l'information dans le réseau. C'est ce point que nous adressons dans le Chapitre 3.

Une fois que le service à été trouvé, il s'agit de s'en servir. Si l'on met de côté l'aspect standardisation des services (point qui ne fait pas l'objet de notre travail), un problème apparaît quand on passe du monde filaire au monde ad hoc. En filaire, une fois la connexion établie, cette dernière reste valide tout au long de la phase d'utilisation du service. Si ce n'est pas le cas, le réseau est considéré comme en panne et aucun mécanisme n'est mis en place car cela n'est pas censé arriver. Il est en effet rare d'être privé de son serveur mail favori et, quand c'est le cas, il suffit souvent d'attendre quelques heures pour de nouveau profiter de plusieurs mois d'utilisation sans problème. Par contre, dans les réseau ad hoc, cette conjecture de fiabilité n'est plus vérifiée. Il n'est en effet pas rare que la mobilité des nœuds entraîne la séparation du réseau en plusieurs composantes disjointes. Si la plupart des protocoles de routage proposent de corriger les routes quand elles deviennent invalides, une fois que deux nœuds ne peuvent plus **physiquement** se joindre, il est trop tard pour tenter quoique ce soit ! Il devient alors intéressant d'analyser l'état du réseau afin d'essayer de prédire les moments où les nœuds vont être physiquement déconnectés car, si on prévoit l'événement suffisamment à l'avance, il sera encore temps de réagir. Cette réaction peut être de différente nature, chercher à dupliquer le service, chercher un autre nœud fournissant un service équivalent, renforcer la connexion en déplaçant d'autre nœuds... Le Chapitre 4 traite des méthodes possibles pour effectuer une telle prédiction.

Chapitre 3

Dissémination d'informations

« Découvrez ce que vous aimeriez faire et faites tout votre possible pour y parvenir »

Jonathan Livingston le goéland, Richard Bach.

Dans ce chapitre, nous présentons une solution originale au problème de diffusion d'information dans les réseaux ad hoc. L'application que nous visons en particulier est la recherche de services mais nous nous attacherons au problème de diffusion dans une optique générale. La solution proposée pourra alors se décliner pour diverses applications comme, évidemment, la recherche de services [BR00], mais aussi le routage géographique [Sto02, BCSW98] ou toute autre application nécessitant la répartition de données dans le réseau en vue de la rendre accessible le plus facilement possible du plus grand nombre de nœuds.

3.1 Objectifs

Après sa création, le cycle de vie d'un service va passer par son utilisation. Dans cette optique, nous devons fournir aux applications un moyen de retrouver un tel service dans le réseau. Dans les réseaux filaires, dans le cas de CORBA [GGM99] ou Jini [Mic99b] par exemple, la recherche de services est gérée par un serveur central. Chaque nouveau service s'enregistre auprès du serveur qui pourra donc donner l'identifiant du fournisseur du service si une application vient à lui demander.

Dans le cadre des réseaux ad hoc, cette solution, comme toute solution centralisée, n'est pas adaptée. En effet, on rencontre plusieurs inconvénients lors de l'utilisation de tels protocoles dans un environnement ad hoc :

– les objets éloignés du serveur central sont pénalisés, le temps d'acheminement de la requête croissant avec la distance ;

- les requêtes sont toujours faites auprès du même serveur qui devient un point critique du réseau. En plus de connaître un phénomène de goulot d'étranglement, si ce serveur est un objet mobile également, il devient difficile de le retrouver ;
- enfin, si le serveur tombe en panne, tout le réseau est paralysé.

Pour palier à ces problèmes, la communauté ad hoc tend à proposer des versions distribuées des protocoles classiques. Par exemple, dans le cadre de la recherche de services, il est intéressant de distribuer la connaissance des services sur l'ensemble du réseau. L'objectif est alors d'obtenir une distribution efficace.

Dans le cadre de l'informatique mobile, disposant de peu ou pas d'accès fixe, il est difficile de pouvoir prévoir la façon dont on va répartir l'information afin qu'elle soit située proche des applications qui vont s'en servir.

De plus, les objets considérés sont soumis à certaines contraintes (que l'on ne retrouve d'ailleurs pas dans les réseaux filaires) comme une faible capacité mémoire et une limitation des réserves d'énergie. Si on ne considère pas ces contraintes, on peut envisager une répartition idéale où chaque objet aurait connaissance de toute l'information disponible dans le réseau. Cependant, une solution réaliste et efficace doit limiter le nombre d'objets qui ont connaissance de l'information et répartir celle-ci de manière à en diminuer le temps de recherche.

Dans ce chapitre, nous proposons un algorithme probabiliste qui permet de distribuer de manière satisfaisante une information parmi les objets formant le réseau. Cet algorithme est basé sur un critère de dissémination efficace de l'information qui prend en considération les distances parcourues par l'information que l'on cherche à répartir et la requête qui vise à la retrouver. Nous proposons ensuite une amélioration de cet algorithme utilisant les ensembles dominants.

Pour la recherche de services, nous nous intéressons à la diffusion dans le réseau de l'identifiant du fournisseur ainsi que de sa description. Néanmoins, dans la suite, nous laisserons de côté l'aspect spécialisé d'un algorithme de diffusion de service. En effet, un tel algorithme peut également être utilisé pour diffuser n'importe quel type d'information et pas seulement des informations relatives à un service. Par exemple, notre algorithme peut être utilisé pour diffuser la position de l'objet en vue d'un routage géographique comme DREAM [BCSW98] par exemple. C'est pourquoi, dans la suite, nous parlerons de diffuser une information au sens général.

3.2 Travaux existants

Bien que nous voulons nous intéresser à la dissémination d'information dans un sens général, la littérature envisage souvent celle-ci en vue d'une utilisation spécifique. En particulier, diffuser de l'information dans le réseau est principale-

ment utilisée pour la découverte de services et la recherche de la position d'un objet en vue d'un routage géographique. Un autre aspect intéressant de la dissémination d'information est également les différentes politiques de cache que l'on peut utiliser afin de gérer la mémoire de manière efficace.

3.2.1 Découverte de services

La mobilité impose aux objets une mise à jour régulière de la connaissance qu'ils ont de leur environnement. La découverte de l'environnement et des services qu'il met à disposition des objets est difficile à réaliser à cause de l'absence d'infrastructure fixe caractérisant les réseaux mobiles.

L'étape préliminaire à la découverte de services est l'étape qui consiste à répartir ces services ou plutôt la connaissance de ces services [1] à travers le réseau. Une bonne dissémination de l'information peut permettre de réduire la quantité de messages émis pour découvrir cette information ainsi que le temps de requête.

Certains travaux traitent de découverte de services sans même aborder cette répartition. Dans [KKR], par exemple, les auteurs proposent d'organiser les services disponibles du réseau en clusters suivant différentes couches. Les clusters sont formés à partir de la proximité physique des nœuds possédant l'information et de la proximité sémantique des informations possédées par les différents nœuds du réseau. Le but de cette approche est de faciliter la recherche de documents à travers le réseau.

Plusieurs propositions de standardisation de protocoles de découverte de services ont été faites par différents consortiums et organismes industriels. Nous pouvons citer Jini [Mic99b] de Sun, SLP (Service Location Protocol) [GCJD99] de l'IETF (Internet Engineering Task Force), UPnP (Universal Plug and Play) [mic99a] de Microsoft, Salutation [sal99] de IBM, ou encore SDP (Service Discovery Protocol) [blu99], protocole de découverte de services de Bluetooth. Le principe de Jini par exemple, repose sur un mécanisme de *lookup* pour la recherche d'un service. Les serveurs enregistrent les services qu'ils fournissent auprès d'un annuaire. SLP fonctionne selon un principe similaire : les serveurs enregistrent leurs services auprès d'un ou plusieurs DA (Directory Agents). Ce type d'architecture, centralisée, ne convient pas aux réseaux ad hoc, la connexion avec l'objet central étant difficile à maintenir [KF02].

Dans [KT04], les services sont enregistrés auprès de DA présents sur des nœuds formant un ensemble dominant (*cf* Section 2.2.3.2). Chaque nœud du réseau, s'il n'est pas dominant, est attaché à un Virtual Access Point (VAP), point d'accès à l'ensemble dominant. Un nœud i enregistre les services qu'il fournit auprès du DA présent sur son VAP, Vap_i. Les nœuds dominants sont sélectionnés en

1. La connaissance des services peut par exemple être concentrée sur un objet auprès duquel les serveurs enregistrent les services qu'ils fournissent.

fonction de la fréquence de rupture de lien qui leur est associée, l'objectif étant d'obtenir un ensemble stable. Cet ensemble est adapté à la topologie du réseau par ajout/suppression de nœuds. L'inconvénient est que cette solution tend à pénaliser toujours les mêmes nœuds, l'ensemble dominant, étant, par construction, le plus stable possible. Pour rechercher l'information, les auteurs proposent un mécanisme de *multicast* (opération consistant à envoyer un paquet à un ensemble de nœuds, par opposition à l'envoie à un seul nœud). Un nœud recherchant un service s'adressera en premier lieu à son VAP puis aux VAP voisins (deux VAP sont au plus séparés par deux sauts) jusqu'à trouver le service souhaité (un TTL permet de stopper une recherche infructueuse).

Une structure de multicast peut être difficile à maintenir. Les auteurs montrent cependant que leur proposition comparée à un mécanisme sans infrastructure centralisée, où les nœuds inondent le réseau jusqu'à obtenir une réponse (*anycast* ou *broadcast*), n'est pas forcément mauvaise. Les points confrontés sont le temps moyen de latence, le nombre de messages de contrôle émis et le taux de requêtes réussies. La proposition est, entre autres, comparée aux protocoles de routage AODV [PR99] et DSR [BCS99] version *anycast*. Ces protocoles sont des protocoles réactifs (les routes sont construites à la demande) et sont à l'origine *unicast*, c'est à dire qu'un paquet émis est destiné à un nœud particulier. Ces deux protocoles ont été choisis par les auteurs car ils sont efficaces en terme de nombre de messages de contrôle envoyés et peuvent ainsi mettre en évidence l'efficacité de la solution.

Dans [LB04], les auteurs proposent une évaluation des performances de plusieurs stratégies de publication et de requête. Un couple de stratégies de publication et de requête exécute en ronde les protocoles de publication et de requête associés.

Les serveurs publient les services qu'ils fournissent et les clients demandent les services dont ils ont besoin respectivement selon les protocoles de publication et de requête utilisés. Les stratégies confrontées sont :

- une stratégie gloutonne ;
- une stratégie incrémentale ;
- une stratégie sans mémoire ;
- une stratégie avec mémoire ;
- une stratégie modeste.

L'évaluation des différentes stratégies se fait par rondes successives. Dans chaque ronde, un serveur publie un service et un client en fait la recherche. À chaque ronde, les stratégies peuvent modifier leur comportement pour tenir compte des rondes passées.

Dans la stratégie gloutonne, le serveur publie le service sur tous les autres nœuds et le client le recherche également sur tous les autres (grâce à un algorithme de *broadcast*).

Dans le cas de la stratégie incrémentale, à chaque ronde, le serveur étend la

zone dans laquelle il publie son service. De même, le client qui fait sa requête étend la zone de cette dernière à chaque ronde.

Dans la stratégie sans mémoire, le serveur choisi aléatoirement un ensemble de nœud et publie son service sur cette ensemble. Le client fait de même et recherche le service sur un ensemble de nœud choisi aléatoirement (mais évidement différent de celui du serveur).

Enfin, dans la stratégie avec mémoire, le client sélectionne aléatoirement un ensemble de nœuds à chaque ronde parmi les nœuds non joint à la ronde précédente.

Quand la stratégie modeste est utilisée, tous les nœuds du réseau publient des services et émettent des requêtes auprès de leurs voisins situés à un saut.

La stratégie gloutonne n'est pas appropriée aux réseaux ad hoc car les objets étant limités en capacité mémoire, il n'est pas réaliste que tous les objets connaissent tous les services disponibles dans le réseau. De plus, si tous les nœuds connaissent tous les services disponibles dans le réseau, il est inutile que les nœuds émettent leurs requêtes à tous les nœuds du réseau. En effet, dans cette configuration de répartition de l'information, une requête à un nœud voisin suffit.

La stratégie sans mémoire peut convenir aux réseaux ad hoc si l'ensemble des nœuds est petit et si ces nœuds sont bien répartis à travers le réseau. L'inconvénient est que cette répartition est aléatoire et ne suit donc aucun schéma reproductible. La stratégie avec mémoire tend vers une connaissance de l'ensemble des services disponibles dans le réseau par chaque nœud et n'est donc pas adaptée aux réseaux ad hoc. La stratégie modeste ne fournit pas une répartition intéressante de l'information puisque, dans le meilleur des cas, les nœuds désirant obtenir l'information publiée par la source réduiront le coût de la requête d'un saut.

Bien que n'étant pas destinée à la découverte de services, la solution proposée dans [KK04] peut trouver application dans ce domaine. Les auteurs évaluent des heuristiques de placement de réplicats dans des systèmes distribués à grande étendue et proposent une méthodologie pour la sélection d'heuristique. Des contraintes sont introduites comme par exemple un seuil de latence moyen. Un système de contraintes est ainsi obtenu. Le concepteur doit calculer la limite inférieure générale et les limites inférieures correspondant à toutes les classes d'heuristiques susceptibles de convenir au système. L'heuristique la plus convenable est celle ayant la plus petite limite. Si cette limite est proche de la limite inférieure générale, il n'existe pas, parmi les classes d'heuristiques non testées, d'heuristique qui serait réellement plus convenable pour le système que l'heuristique ayant cette limite. Cette approche ne conviendrait pas à un réseau de type ad hoc puisque c'est une approche « off-line », où le concepteur doit configurer les paramètres et choisir une heuristique appropriée au système à chaque fois que la topologie est modifiée.

3.2.2 Routage géographique

Lorsque le routage utilisé est géographique [BCSW98, Sto02], un nœud désirant envoyer un message à un autre doit en premier lieu obtenir la localisation de celui-ci. Pour accéder à une telle information, le nœud considéré peut émettre par broadcast une requête de localisation et attendre la réponse d'un nœud possédant l'information souhaitée. Chaque nœud connaît sa position géographique grâce à un système de positionnement comme GPS [2] [gps97]. Dans ce type de routage, un nœud devant transmettre un paquet choisit, par exemple, dans sa liste de voisins, le nœud le plus proche de la destination comme prochain saut.

Plusieurs algorithmes utilisent des tables de hachage distribuées (DHT [3]) pour déterminer la localisation des nœuds formant le réseau. C'est le cas notamment de Grid [Li01] et de Tribe [VdAFdR03, VdAFdRar].

Le service de localisation Grid [Li01] (GLS [4]) est un service de localisation distribué. Les nœuds mettent à jour leur localisation auprès de serveurs de localisation. Chaque nœud peut tenir le rôle de serveur de localisation vis à vis d'autres nœuds. Le principe de Grid est de partitionner le réseau de manière récursive en suivant le principe des *quadtree*. L'espace est découpé en quatre zone carrées, puis on partitionne à nouveau ces zones de manière récursive jusqu'à arriver à la taille minimale souhaitée. Le niveau 0 correspond au carré de taille minimale. Un carré de niveau n est composé de quatre carrés de niveau $n-1$. Chaque nœud doit choisir un nœud comme serveur de localisation dans chacune des trois zones prenant part avec lui à la formation d'une zone de niveau supérieur. La Figure 3.1 montre dans quelles zones le nœud B devra choisir ses serveurs de localisation. Cette répartition assure que plus on s'éloigne d'un nœud, plus ses serveurs de localisation sont éparpillés. Comme le nombre de serveurs de localisation associés à un nœud diminue avec la distance, le nombre total de nœuds mémorisant l'information tend à être minimisé. Un nœud met à jour sa localisation auprès de ses serveurs de localisation lorsqu'il a bougé d'une certaine distance d depuis sa dernière localisation mise à jour. Un nœud va choisir comme serveurs de localisation des nœuds dont les identifiants sont « proches » du sien. Le nœud le plus proche d'un nœud B est le nœud dont l'identifiant est le plus petit identifiant supérieur à celui de B. L'espace d'adressage considéré est cyclique, c'est à dire par exemple, si on considère l'espace d'adressage compris entre 1 et 20, que 18 est plus proche de 1 que de 14. Un nœud A désirant contacter le nœud B, suivra la même démarche. Il enverra sa requête au nœud qu'il connaît le plus proche de B. Ce nœud transmettra la requête de la même manière et ainsi de suite jusqu'à atteindre un serveur de localisation de B.

Le principal inconvénient de cette solution est que ce système nécessite l'uti-

2. Global Positioning System.
3. Distributed Hash Tables
4. Grid Location Service

lisation de GPS. Ceci n'est évidement pas gênant dans le cas de l'application pour laquelle Grid à été conçu, mais empêche donc son utilisation dans un cas plus général ou l'on ne disposerait pas d'un tel équipement. De plus, il peut se produire qu'un nœud aille chercher une information loin de lui alors qu'il peut la trouver à proximité. Ceci est le cas pour les nœuds situés en bordure de zone puisque la distance entre un nœud en bordure désirant atteindre un nœud près de lui en bordure d'une zone différente et le serveur de localisation de ce nœud, est forcément plus grande que celle séparant les deux nœuds.

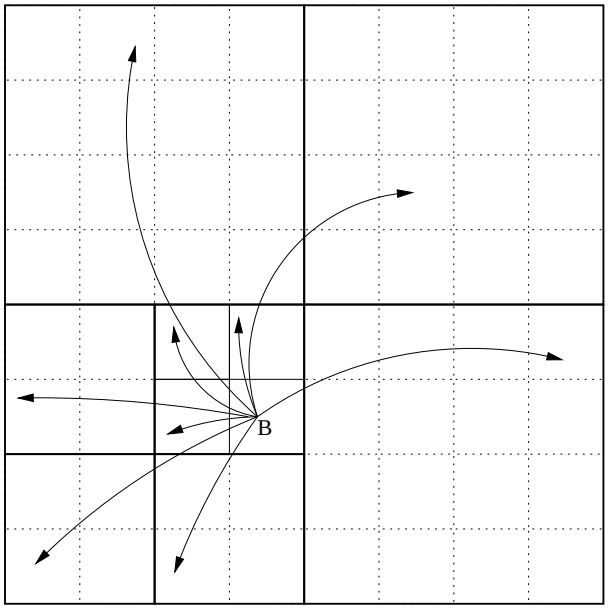


FIG. 3.1 – *Exemple de répartition Grid pour le nœud B.*

Dans Tribe [VdAFdR03, VdAFdRar], on retrouve la notion de point de rendez-vous. Un nœud désirant envoyer un message à un autre nœud peut retrouver l'endroit où la localisation de celui-ci est stockée, à partir de son identifiant. Un nœud met à jour sa localisation auprès d'un seul nœud : son ancre. Ce nœud est découvert à la demande. Chaque nœud formant le réseau possède un identifiant universel, un identifiant virtuel, une adresse relative et une zone de contrôle. Un nœud calcule son identifiant virtuel dans l'espace d'adressage à partir de son identifiant universel. Son ancre est alors le nœud dont la zone de contrôle contient cet identifiant virtuel. L'ancre connaît l'adresse relative du nœud et sa localisation courante. Lorsqu'un nœud désire communiquer avec un autre, il doit préalablement demander sa localisation courante. Pour cela, il calcule l'identifiant virtuel du nœud destination en utilisant le même calcul que celui-ci. Il peut ainsi trouver le nœud ancre de la destination. Ceci est possible car les identifiants universels et la fonction de conversion en identifiants virtuels sont connus de tous les nœuds

formant le réseau. Lorsqu'un nœud transmet un paquet, il choisit dans son voisinage un nœud dont la zone de contrôle est proche de la zone de contrôle contenant l'identifiant virtuel de la destination. Contrairement à Grid, Tribe n'utilise pas d'information géographique pour déterminer les nœuds qui vont mémoriser la localisation des autres nœuds et pour router les paquets. L'inconvénient de cette solution est que le nœud ancre d'un nœud peut se trouver loin de lui. Ceci augmente le nombre de messages émis ainsi que le temps de requête.

3.2.3 Techniques de cache coopératif

Quand un nœud transmet à un autre nœud une donnée qui ne lui est pas destinée, il peut décider de conserver cette donnée ou une référence sur cette donnée ainsi que l'adresse du nœud à qui elle destinée (cela peut permettre par la suite d'accéder à la donnée de manière plus rapide). Lorsqu'une requête concernant cette donnée parvient jusqu'à ce nœud, il peut, s'il l'a mémorisée, retourner directement cette donnée au nœud qui la demande. L'ensemble des nœuds constituant le réseau forment un système de cache coopératif dans lequel les nœuds tirent bénéfice des informations présentes dans le cache des autres nœuds. Une technique de cache coopératif est caractérisée notamment par sa politique de mémorisation et de remplacement d'information dans le cache.

Dans [NSC03], les auteurs proposent des stratégies de cache efficaces au niveau de l'économie d'énergie. Ils formulent le problème de placement de cache comme un problème de programmation linéaire. Ils introduisent donc des contraintes comme l'énergie nécessaire à la transmission et à la réception d'une donnée. Les auteurs se sont principalement attachés à minimiser l'énergie consommée mais pas la taille de cache utilisée. De plus, les coûts de stockage sont considérés comme étant insignifiants.

Dans [SI03], Sailhan *et al.* proposent une technique de cache de données web minimisant les coûts énergétiques de communication entre les nœuds formant le réseau, en particulier quand un point d'accès fixe n'est pas à portée de communication d'un nœud désirant récupérer une donnée sur le web. La solution proposée fournit une stratégie de remplacement de l'information dans le cache des objets en plus d'un protocole de cache coopératif. Si un nœud est à N sauts d'un point d'accès fixe au web, il ne contactera pas un nœud à distance supérieure ou égale à N pour récupérer un document. Si un nœud désirant un document dispose d'un point d'accès au web à portée de communication, il s'adresse directement à celui-ci. Si ce n'est pas le cas, il va d'abord interroger ses voisins (nœuds à portée de communication). En dernier lieu, il envoie une requête au point fixe le plus proche. Si un nœud acheminant la requête vers le point d'accès au web possède le document, il l'envoie directement au nœud demandeur. Chaque nœud conserve en mémoire un profil des nœuds avec lesquels il correspond pour pouvoir déter-

miner ceux étant susceptibles de mémoriser une donnée l'intéressant. De plus, si deux nœuds sont supposés posséder une copie de la donnée de manière équitable, le nœud de plus grande capacité mémoire par exemple sera privilégié, la donnée ayant moins de risques d'avoir été remplacée. La stratégie locale de cache associée à ce protocole consiste à mesurer la probabilité d'accès à une donnée et le coût énergétique associé à la récupération de cette donnée. La probabilité d'accès est approximée par le nombre de fois que le document a été demandé depuis qu'il a été mis en cache. Le coût énergétique associé à la récupération d'une donnée dépend entre autres de la présence de celle-ci dans l'entourage proche du nœud. La considération de la validité du document intervient après la considération du coût énergétique quand l'énergie restante du nœud est faible. C'est la donnée ayant le moins d'importance selon ces critères qui est remplacée.

Cette solution est efficace au niveau de la consommation d'énergie et du temps de recherche mais elle perd en intérêt si elle est adaptée à un environnement sans point d'accès fixe, la recherche d'une donnée devant être limitée par un TTL plutôt que par le fait d'atteindre un tel point.

Yin *et al.* proposent dans [LG04] trois approches se distinguant par la mise en cache :

- de la donnée (*CacheData*);
- du chemin menant à la donnée (*CachePath*) ;
- selon le contexte, de la donnée ou du chemin menant à la donnée (*HybridCache*).

Dans ces trois approches, un nœud ayant besoin d'une donnée s'adresse au nœud possédant la copie originale de celle-ci. La cohérence des caches est maintenue grâce à un TTL.

Dans *CacheData*, les nœuds intermédiaires relayant un message, mémorisent la donnée si elle a été beaucoup demandée auparavant ou s'ils ont assez d'espace mémoire libre, pour pouvoir répondre aux futures requêtes. Pour éviter que plusieurs nœuds ne cachent la donnée inutilement, un nœud ne mets pas en cache une donnée si toutes les requêtes proviennent du même nœud.

Dans *CachePath*, les nœuds intermédiaires mémorisent le chemin de la donnée pour rediriger les requêtes futures vers le nœud le plus proche possédant la donnée.

Dans *HybridCache*, des seuils sont instaurés. Ces seuils concernent certains critères comme la taille de la donnée, la proximité entre le nœud intermédiaire et le nœud possédant une copie et le TTL associé à la donnée. Si la donnée est de petite taille, *CacheData* peut être adopté car seulement une petite partie du cache va être utilisée pour stocker la donnée. Si Le TTL de la donnée est faible, il n'est pas judicieux d'utiliser *CachePath* puisqu'elle risque d'être invalidée rapidement. Si le nœud intermédiaire et le nœud possédant une copie de la donnée sont très proches, *CachePath* peut faire économiser un grand nombre de sauts pour accéder

à la donnée. Ces approches sont efficaces en termes de temps de requête et de complexité de messages. Leur inconvénient est que si des nœuds proches du nœud demandeur ne se trouvent pas entre celui-ci et le nœud possédant la copie originale de la donnée, ils ne seront pas détectés.

Les solutions mises en œuvre précédemment ne tiennent pas compte des ressources des objets en terme de capacité mémoire. De plus, la maintenance des architectures proposées peuvent être difficile à maintenir à cause de la mobilité du réseau. C'est pourquoi nous avons voulu nous orienter vers une solution simple, qui distribue efficacement et rapidement les informations dans le réseau afin de répondre à ces contraintes.

3.3 Vers une dissémination efficace

Une dissémination efficace de l'information doit minimiser le nombre de nœuds pénalisés par la mémorisation de l'information tout en tenant compte des distances parcourues par l'information et par la requête. De plus, il faut pouvoir l'effectuer grâce à un protocole simple et rapide, qui permettra de supporter la mobilité. Pour pouvoir évaluer le caractère efficace d'une dissémination, nous proposons un critère qui, s'il est vérifié par l'ensemble du réseau, témoignera de la bonne distribution de l'information. Puis, nous proposerons un algorithme distribué simple permettant de faire une dissémination vérifiant ce critère.

3.3.1 Critère de dissémination efficace de l'information

Une répartition efficace de l'information doit considérer les distances parcourues par l'information et par la requête. Diminuer la distance entre un nœud ayant conservé une trace de l'information et un nœud qui en fait la recherche permet d'en diminuer le temps ainsi que le coût.

Pour formaliser cette idée, nous introduisons un facteur d'éloignement λ. Ce facteur permet de contrôler la distance maximale séparant un nœud possédant une trace d'une information d'un nœud n'en possédant pas. De plus, un nœud doit pouvoir trouver l'information recherchée ou une trace de celle-ci sur un nœud plus proche de la source que lui. Intuitivement, plus λ sera petit, plus la distance séparant un nœud possédant l'information d'un nœud ne la possédant pas sera faible et donc, plus le coût de la recherche sera faible également. Par contre, le nombre de nœuds nécessaires à la dissémination de l'information sera plus important. Cette formalisation est donnée par la définition 4.

Définition 4 *Soit v un nœud, $Cache(v)$ représente l'ensemble des informations qui sont stockées dans le cache de v. Soit u un nœud qui possède une information I_u à disséminer à travers le réseau. L'information I_u est efficacement disséminée si la propriété suivante est vérifiée :*

$$\forall v \in V, m < d(u,v) \leq M, \begin{cases} I_u \in Cache(v) \\ \text{ou} \\ \exists w \in V \text{ t.q.} I_u \in Cache(w), d(v,w) < \lambda d(v,u), \\ d(u,w) \leq d(u,v), \end{cases}$$

m et M représentent les distances minimale et maximale entre lesquelles nous considérons le critère, $d(x,y)$ représente la distance en nombre de sauts entre les nœuds x et y, λ est le facteur d'éloignement, ($\lambda \in]0,1[$).

Ce critère reflète le fait qu'un nœud peut tirer avantage à trouver l'information sur un nœud plus proche que lui de la source, à condition que ce nœud se trouve à moins d'une certaine distance de lui, distance représentant une fraction de sa distance à la source. La Figure 3.2 met en évidence ceci pour $\lambda = \frac{1}{2}$. Le nœud S possède une information I_S qu'il a disséminée. Le nœud V vérifie la propriété car il est couvert par le nœud U, plus proche de S que V, la distance séparant U et V étant inférieure à la moitié de la distance séparant V et S.

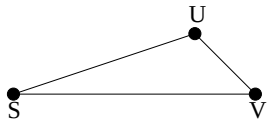


FIG. 3.2 – *Exemple de vérification de la propriété pour $\lambda = \frac{1}{2}$.*

Un équilibre juste est atteint pour $\lambda = \frac{1}{2}$ puisque les distances parcourues par l'information et par la requête sont alors identiques. Plus on s'éloigne de la source d'information, plus la distance maximale séparant un nœud possédant cette information dans son cache et un nœud ne la possédant pas est grande. Ceci est acceptable car un nœud éloigné de la source peut accepter de payer cette distance pour trouver l'information dont il a besoin. De plus, un nœud éloigné d'une source disséminant une information, est susceptible d'être plus proche d'une autre source disséminant la même information dans le réseau. Dans le cas de la recherche de services, par exemple, il est plus pertinent de privilégier l'utilisation d'un service proche de soit afin d'obtenir une meilleur qualité de service.

Afin de trouver une méthode permettant de disséminer une information selon ce critère, nous avons tout d'abord étudié le placement de l'information dans un cas parfait. Dans ce cas, on peut placer les nœuds géométriquement afin de vérifier la propriété de manière optimale.

3.3.2 Étude géométrique

Les formules présentées dans cette section sont déduites de propriétés géométriques simples. Le lecteur pourra aisément les retrouver.

Nous étudions ici le meilleur emplacement possible des nœuds mémorisant l'information dans un graphe de densité infinie. Nous pouvons donc disposer les nœuds où nous le souhaitons. L'étude réalisée est une étude géométrique, nous utilisons donc indifféremment les termes « nœud » et « point ». Un nœud u ayant mémorisé l'information « couvre » un nœud v s'il est plus proche de la source que v et si la distance le séparant de v est inférieure à la distance de v à la source (cf. Définition 4). Nous utilisons les expressions « couvre », « est couvert par » ou « couverture » lorsque ceci est vérifié.

Nous considérons la dissémination d'une information par un nœud S, source d'information. m représentant la distance minimale à partir de laquelle nous examinons la dissémination de l'information, les nœuds situés à l'intérieur du cercle de centre S et de rayon m sont couverts par S. Les points situés sur le cercle de centre S et de rayon m sont en limite de couverture. Ces points se trouvent au rang 0, à distance m de la source. Certains de ces points vont devoir mémoriser l'information et permettre ainsi d'étendre la couverture au delà du rang 0. Ces points vont couvrir jusqu'à une certaine distance de la source.

Soit un nœud u ayant mémorisé au rang 0. Notons m_0 la distance à la source correspondant au rang 0. On a donc $m_0 = m$. L'ensemble des nœuds v en limite de couverture du nœud u est un cercle dont le centre O est situé à distance c de la source et dont le rayon vaut R_0 (Figure 3.3). c et R_0 s'expriment en fonction de λ de la manière suivante :

$$c = \frac{1}{1-\lambda^2} \cdot m_0 \tag{3.1}$$

$$R_0 = \frac{\lambda}{\lambda^2-1} \cdot m_0 \tag{3.2}$$

Les points couverts par le nœud u sont les points situés à l'intérieur du cercle obtenu. Le rayon R_0 de ce cercle est plus grand que la distance Ou. On retrouve facilement ce résultat en résolvant l'inéquation $R_0 > Ou$. On obtient alors la condition $\lambda^2 < \lambda$, ce qui est toujours vrai.

Il s'agit maintenant de déterminer l'emplacement des nœuds qui vont mémoriser pour couvrir les nœuds qui ne le sont pas par les nœuds mémorisant au rang 0. Nous choisissons de placer un nœud mémorisant à l'intersection la plus éloignée de la source de deux zones consécutives de couverture (cercles associés à deux points ayant mémorisé). Placer un point mémorisant à cet endroit permet de couvrir une plus grande zone de points non encore couverts et donc de minimiser le nombre total de points mémorisant nécessaires pour couvrir toute la zone étudiée (délimitée par M). Cet méthode de construction nous impose une condition sur le nombre de nœuds que l'on peut placer au rang 0 (donc sur le nombre de nœuds qui vont mémoriser l'information). En effet, la distance séparant deux nœuds qui mémorisent doit être telle que les couvertures associés aux

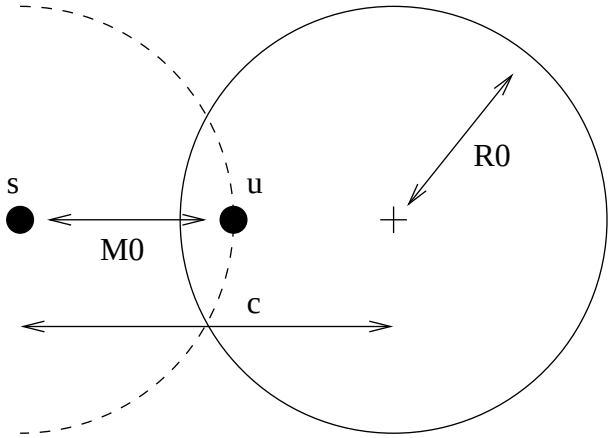


FIG. 3.3 – *Nœuds en limite de couverture de u.*

deux nœuds aient une intersection. Ceci est illustré figure 3.4.

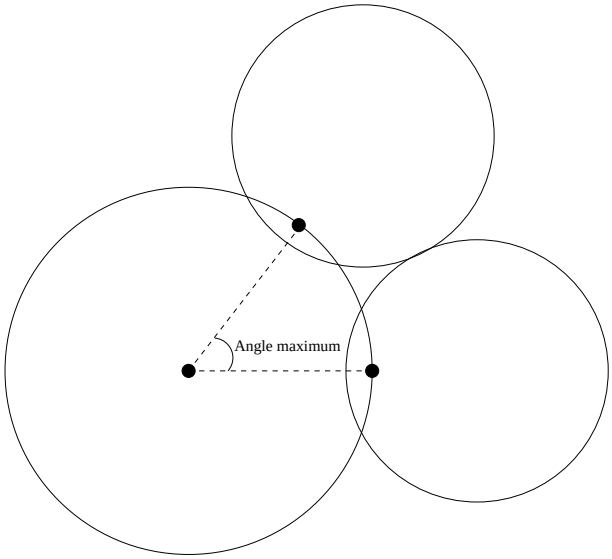


FIG. 3.4 – *Angle maximum pouvant exister entre deux nœuds afin que leurs cercles de couverture aient une intersection.*

Si on note θ_{max} l'angle formé par les deux nœuds, on a :

$$\theta_{max} = 2acos\left(1 - \frac{\lambda^2}{2}\right) \tag{3.3}$$

Nous en déduisons donc le nombre de nœuds mémorisant minimum nécessaires pour couvrir le rang 0 en totalité :

$$a_{min} = \left\lceil \frac{2\pi}{\theta_{max}} \right\rceil \tag{3.4}$$

Plus le nombre de nœuds mémorisant à un rang est faible, plus l'éloignement entre deux nœuds mémorisant à ce rang est important, plus l'intersection des cercles associés à deux points consécutifs mémorisant sera proche de la source et donc, plus le nombre de rangs nécessaires pour couvrir toute la zone sera important.

À l'inverse, plus le nombre de nœuds mémorisant à un rang est important, plus l'intersection des cercles associés à deux points consécutifs mémorisant sera éloignée de la source et donc, moins le nombre de rangs nécessaires pour couvrir toute la zone sera important.

Il s'agit donc de déterminer le nombre k de points à ajouter au nombre minimum a_{min} de points nécessaires pour couvrir le rang 0 de manière à minimiser le nombre total de points mémorisants. Le nombre de points mémorisants au rang 0 sera alors $a_0 = a_{min} + k, k \in \mathbb{N}$.

Ce nombre est facilement calculable expérimentalement. La Figure 3.5 montre que celui-ci est égal à 1 lorsque $\lambda = \frac{1}{2}$ et que la zone à couvrir est de 20 sauts. 6 rangs sont alors nécessaires pour atteindre cette couverture.

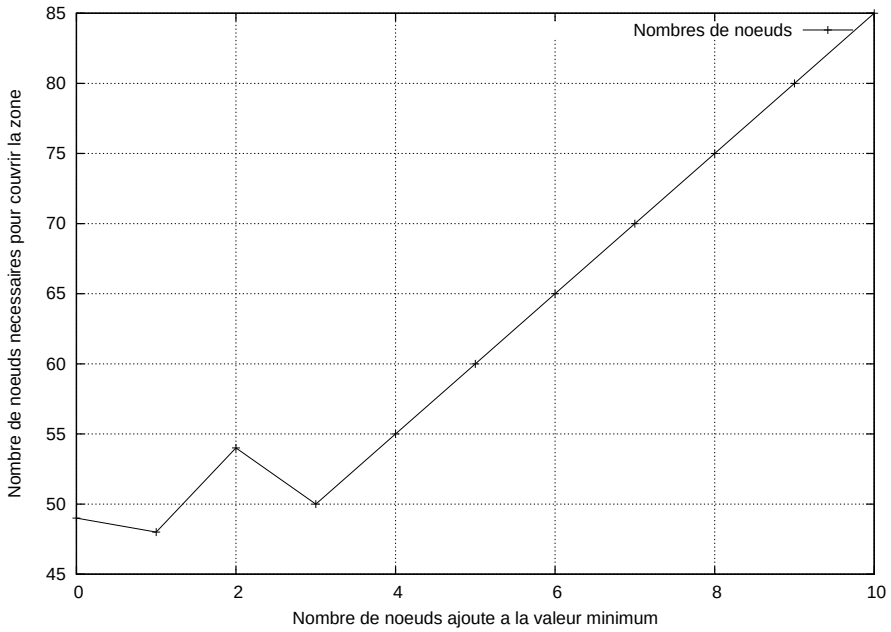


FIG. 3.5 – *Nombre de nœuds à ajouter à a_{min} pour couvrir toute la zone.*

Une fois la valeur de k déterminée, la valeur de l'angle θ (l'angle séparant deux points mémorisant sur le rang 0) peut être déduite :

$$\theta = \frac{2\pi}{a_{min} + k}. \tag{3.5}$$

Connaissant l'angle existant entre deux nœuds du rang 0, on peut s'intéresser à l'extension de la couverture des nœuds du rang 0. Nous devons choisir des nœuds situés en limite de couverture des nœuds de rang 0 à une distance m_1 de la source. Ces nœuds appartiendront alors au rang 1.

La distance à la source m_1 des nouveaux points mémorisant est alors exprimée comme suit:

$$m_1 = \frac{\lambda^2 \cos\frac{\theta}{2} + \sqrt{\lambda^2 - \left(\sin\frac{\theta}{2}\right)^2}}{1 - \lambda^2} \cdot m_0 + \cos\frac{\theta}{2} \cdot m_0 \tag{3.6}$$

θ représente comme précédemment l'angle séparant deux points ayant consécutivement mémorisé l'information.

Le rang $n+1$ est déterminé en considérant les intersections les plus éloignées de la source des cercles associés aux points consécutifs ayant mémorisé au rang n. m_n peut être déterminée de la manière suivante :

$$m_n = \tau^n \cdot m_0$$

avec

$$\tau = \frac{\lambda^2 \cos\frac{\theta}{2} + \sqrt{\lambda^2 - \left(\sin\frac{\theta}{2}\right)^2}}{1 - \lambda^2} + \cos\frac{\theta}{2}$$

La Figure 3.6 montre un exemple de construction pour $\lambda = 0.8$, ici, trois rangs sont représentés. Les points mis en évidence sont les points qui mémorisent.

La construction du rang $n+1$ à partir du rang n comme expliquée précédemment, implique que l'angle θ séparant deux nœuds mémorisant est identique d'un rang à l'autre. Le nombre de nœuds mémorisant est donc identique d'un rang à l'autre. Lorsqu'il n'y a pas d'intersection possible entre deux cercles, nous choisissons de placer un point mémorisant au milieu de la droite issue des deux points coupant le cercle de rang juste inférieur au rang courant. Ce choix permet de couvrir un plus grand nombre de points non encore couverts.

Par construction, le nombre de points mémorisant reste donc le même pour tous les rangs :

$$a_0 = a_1 = \ldots = a_n$$

a_n représente le nombre de nœuds mémorisant au rang n.

À partir de cette étude géométrique, nous proposons un protocole probabiliste qui permet de diffusion l'information en respectant notre critère. Ce protocole est distribué et exécuté sur chacun des nœuds, qui prennent leur décision de manière totalement indépendante en fonction de leur distance à la source de l'information.

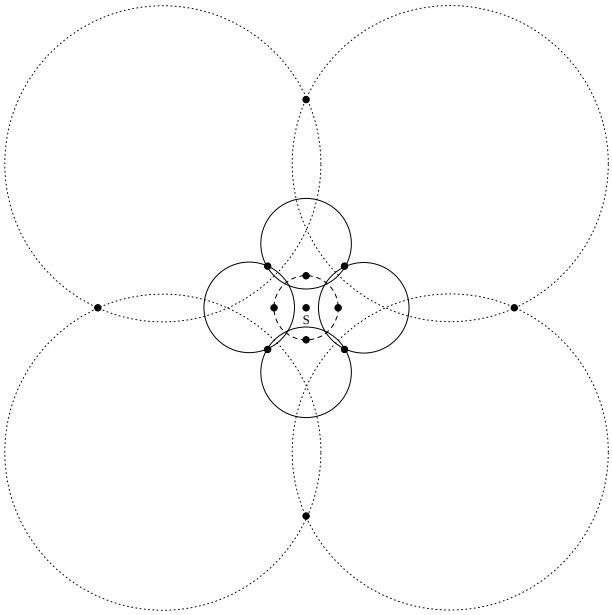


FIG. 3.6 – *Exemple de construction pour* $\lambda = 0.8$.

3.3.3 Algorithme purement probabiliste

Par la suite, nous qualifions de source un nœud qui possède une information à disséminer. Tout nœud du réseau peut être source d'information.

À partir du critère présenté dans la section précédente, nous déduisons une probabilité pour chaque nœud formant le réseau de mémoriser une information en fonction de sa distance à la source. Le rang r_i correspond à la distance m_i à la source. Un certain nombre de nœuds a_0 doivent théoriquement mémoriser l'information à cette distance. La valeur de m_i n'est pas forcément entière. La distance entre deux nœuds se calculant en nombre de sauts parcourus d'un nœud pour atteindre l'autre (valeur entière), le nombre de nœuds qui mémorisent nécessaires à la distance m_i est réparti sur les distances entières inférieure et supérieure à m_i. La probabilité qu'a un nœud u de mémoriser est alors définie de la manière suivante :

$$P(d) = \begin{cases} \frac{a_0}{NC_d} & i \text{ t.q. } m_i = d, \\ \frac{a_0}{NC_d + NC_{d+1}} & \text{t.q. } m_i \notin \mathbb{N}, \\ & \text{et } \lfloor m_i \rfloor = d(S,u), \\ 0 & \text{sinon,} \end{cases} \tag{3.7}$$

NC_d est le nombre de nœuds situés à distance d de la source. Nous approximons ce nombre en considérant la surface du disque de rayon extérieur Rd et de rayon intérieur $R\,(d-1)$ comme représenté par la Figure 3.7. Cette surface est exprimée

ainsi :

$$S = \pi R^2 d^2 - \pi R^2 (d-1)^2 = \pi R^2 (2d-1)$$

R est le rayon de transmission. NC_d peut donc être approximé par la formule suivante :

$$NC_d = D(2d-1) \tag{3.8}$$

D représente le nombre de nœuds par zone de communication (*i.e.* la densité du réseau). La précision de cette approximation est montrée par la Figure 3.8.

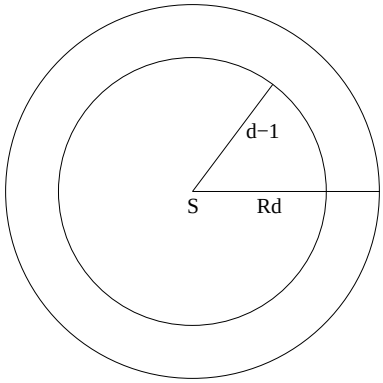


FIG. 3.7 – *Approximation de NC_d.*

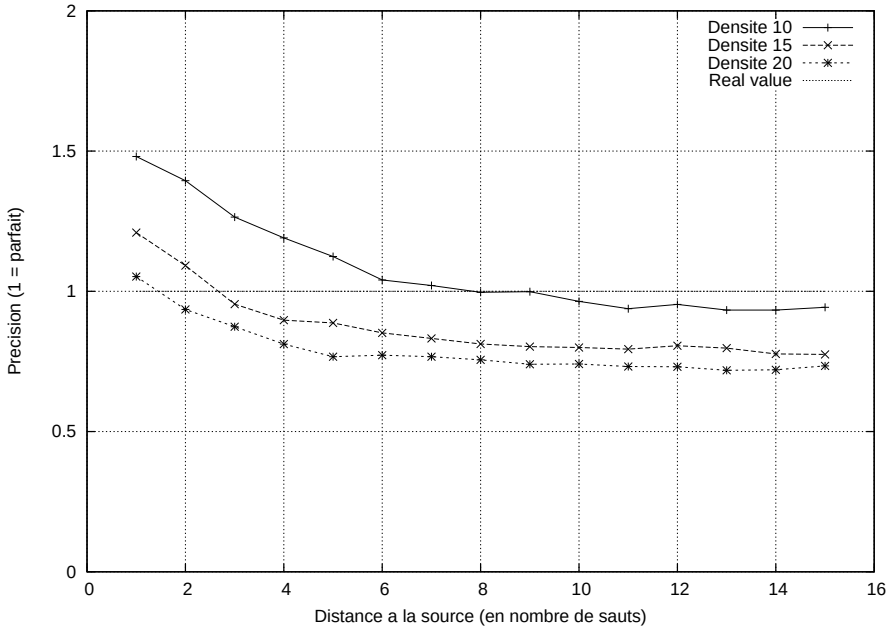


FIG. 3.8 – *Evaluation de la précision de NC_d.*

La publication est réalisée par diffusion d'un paquet contenant des renseignements sur l'information disséminée. Le nombre de sauts donne la distance à la source du nœud qui reçoit ce paquet. Seuls les nœuds appartenant à un rang auront une probabilité de mémorisation différente de 0. L'algorithme à exécuter sur chaque nœud à la réception d'un paquet de publication est donné Figure 3.9.

Pour tous les nœuds $u \in G(V,E)$
  Générer un nombre aléatoire p
  Si $p < P(u)$
    Cacher l'information dans le nœud u
  Fin si
Fin Pour

FIG. 3.9 – *Algorithme de dissémination.*

Pour éviter qu'un nœud déjà « couvert » par le dernier nœud qui a mémorisé ne mémorise à son tour, la distance à la source de celui-ci est conservée dans le paquet de dissémination.

3.3.4 Amélioration par utilisation d'ensemble dominants

Lorsque l'algorithme précédent est exécuté, deux nœuds voisins, ou très proche, peuvent mémoriser l'information. Ceci est inutile car une seule mémorisation suffirait à vérifier le critère, et est du au fait que les nœuds prennent leur décision de manière totalement indépendante.

Pour réduire ce phénomène, nous utilisons les ensembles dominants. Par leur nature, les ensembles dominants assurent que tout nœud du réseau se situe au plus à un saut d'un nœud dominant. Dans notre cas, si seuls les nœuds appartenant à un ensemble dominant ont une probabilité de mémoriser l'information, cela limitera la probabilité d'avoir deux nœuds voisins mémorisant la même information.

Dans les algorithmes visant à déterminer des ensembles dominants (cf. Section 2.2.3.2 page 24), c'est généralement l'identifiant du nœud qui est utilisé comme priorité. L'inconvénient est que les nœuds pénalisés (dans le cadre de la diffusion de message par exemple) sont toujours les mêmes.

La règle que nous utilisons pour construire un ensemble dominant est une adaptation de la règle énoncée dans [CSR04]. Cette règle permet de construire un ensemble dominant connexe. Dans notre cas, la connexité n'est pas nécessaire (et n'est même pas souhaitable pour minimiser la mémorisation par deux nœuds voisins), les nœuds mémorisant l'information n'ayant pas besoin de pouvoir communiquer entre eux. La règle utilisée est donc la suivante : un nœud u n'est pas

dominant si l'ensemble de ses voisins de priorités supérieures à lui « couvre » tous ses voisins.

Pour éviter que les nœuds pénalisés soient toujours les mêmes et ce indépendamment de la source, nous utilisons comme priorité pour construire un ensemble dominant un hash entre l'identifiant du nœud et l'identifiant de la source. Ainsi, un ensemble dominant différent est construit par source d'information différente. Chaque nœud sait s'il est dominant pour une source donnée d'information.

Dans l'algorithme précédent, deux choses changent :

- seuls les nœuds dominants ont une probabilité de mémoriser l'information ;
- la probabilité de mémoriser reste la même, à part que NC_d représente maintenant le nombre de nœuds dominants à distance d de la source.

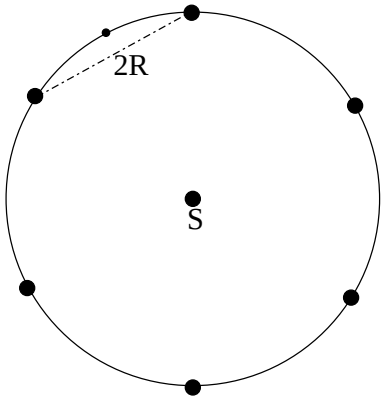


FIG. 3.10 – *Approximation de NC_d dans le cadre des ensembles dominants.*

Supposons que les nœuds dominants soient situés sur des cercles de même centre, à savoir la source, et de rayons croissants. La distance séparant deux nœuds dominants est alors $2R$, R étant le rayon de transmission des nœuds (de par la définition des ensembles dominants). Ceci est représenté par la Figure 3.10. Nous approximons donc le nombre de nœuds dominants à distance d de la source en considérant le périmètre du cercle de rayon Rd et la distance $2R$:

$$NC_d = \pi d, \tag{3.9}$$

La Figure 3.11 montre la précision de cette approximation pour une densité de 15 nœuds par zone de communication. Le nombre de nœuds dominants à distance d de la source est surestimé à une distance relativement éloignée de celle-ci, ce qui n'est pas gênant. En effet, nous verrons par la suite que nos objectifs sont toujours atteints.

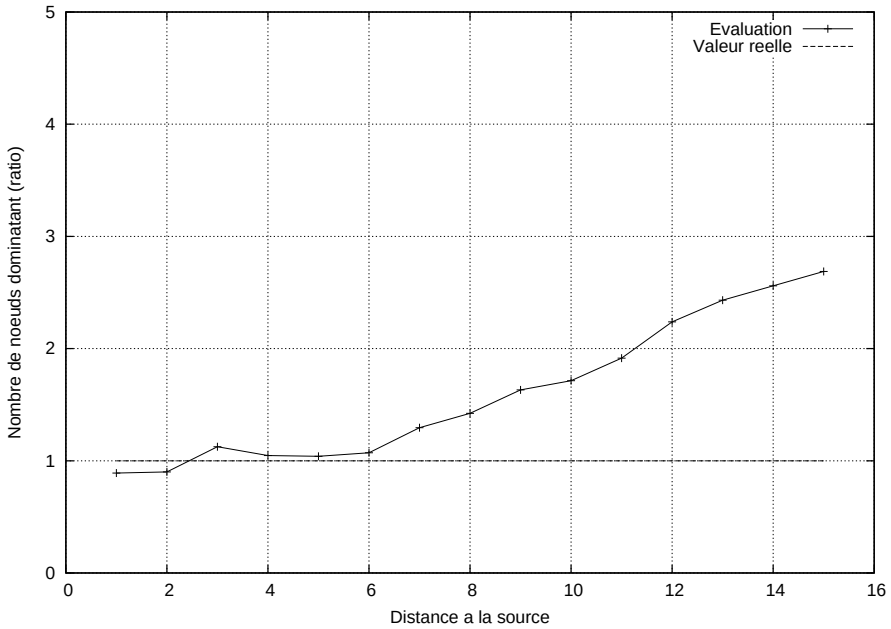


FIG. 3.11 – *Evaluation de la précision de NC_d dans le cas des ensembles dominants.*

3.3.5 Algorithme de recherche

Grâce à notre algorithme de dissémination, l'information est répartie de façon efficace dans le réseau. Concrètement, cela signifie qu'un nœud qui recherche cette information a de forte chance de la trouver proche de lui. Il est donc facile de proposer un algorithme de recherche qui tient compte de cette propriété. Comme le nœud s'attend à trouver l'information proche de lui, il commence à interroger ses voisins proches (2 ou 3 sauts). S'il n'obtient pas de réponse, il augmente la portée (en nombre de sauts) de sa requête. Il réitère le procédé, en augmentant toujours la portée, jusqu'à trouver l'information qu'il recherche ou avoir augmenté la distance maximale de la requête jusqu'à une distance D_{max} qu'il juge trop importante. Cette distance maximale doit être fonction de la nature de l'information à rechercher ainsi que de l'importance pour le nœud de la trouver. La Figure 3.12 propose l'algorithme correspondant. H est la portée (en nombre de sauts) de la requête, D_{max} est la distance maximale au delà de laquelle on arrête l'algorithme et on considère la requête comme un échec et $f(H)$ est une fonction qui défini la façon dont H sera augmenté entre chaque requête. Au plus cette fonction donne un résultat élevé, au plus on pourra trouver l'information rapidement (car on va tout de suite commencer à chercher loin) mais au plus on encombrera le réseau. On peut, par exemple, prendre $f(H) = \frac{H}{2}$.

$H \leftarrow 0$
Tant que $H < D_{max}$
  Effectuer une recherche de l'information à distance H
  Si l'information est trouvée
    Arrêt
  Sinon
    $H \leftarrow H + f(H)$
  Fin Si
Fin Tant que

FIG. 3.12 – *Algorithme de recherche*

3.4 Expérimentations

Les expérimentations ont été réalisées sur un graphe unitaire de 500 nœuds. Les nœuds sont répartis uniformément dans une surface carrée. Le rayon de communication est de 10 mètres.

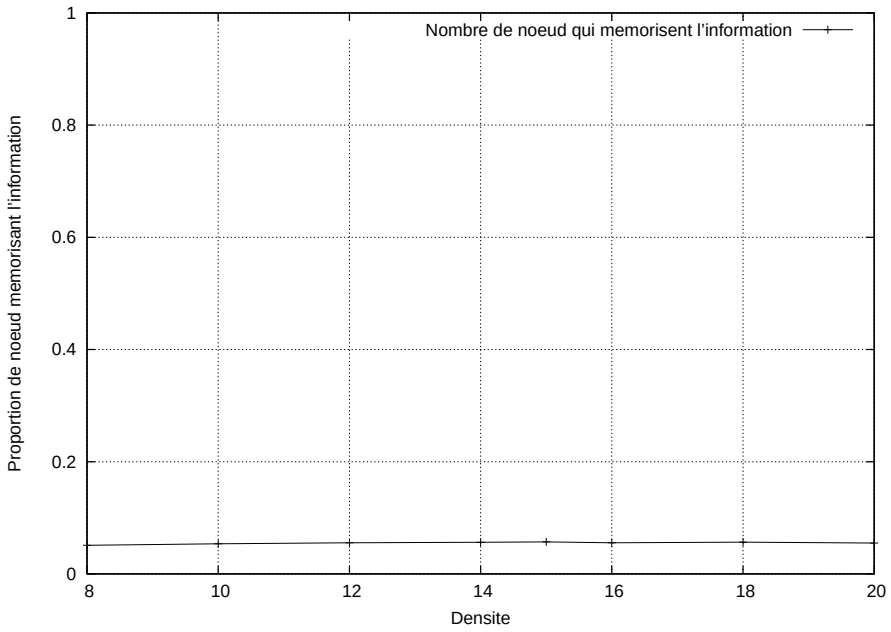
Un nœud au centre de la surface est choisi pour tenir le rôle de source de l'information. Ce nœud diffuse un paquet de dissémination contenant son identifiant, et des renseignements concernant l'information qu'il dissémine. Ces renseignements dépendent de l'information qui est publiée. Dans le cadre de la publication de services, ce paquet peut contenir la description du service et l'identifiant du nœud qui le publie. Les nœuds qui transmettent le paquet de dissémination exécutent l'algorithme donné Figure 3.9. Le paramètre M n'est pas fixé, la zone considérée à couvrir étant la totalité du réseau.

3.4.1 Algorithme probabiliste de base

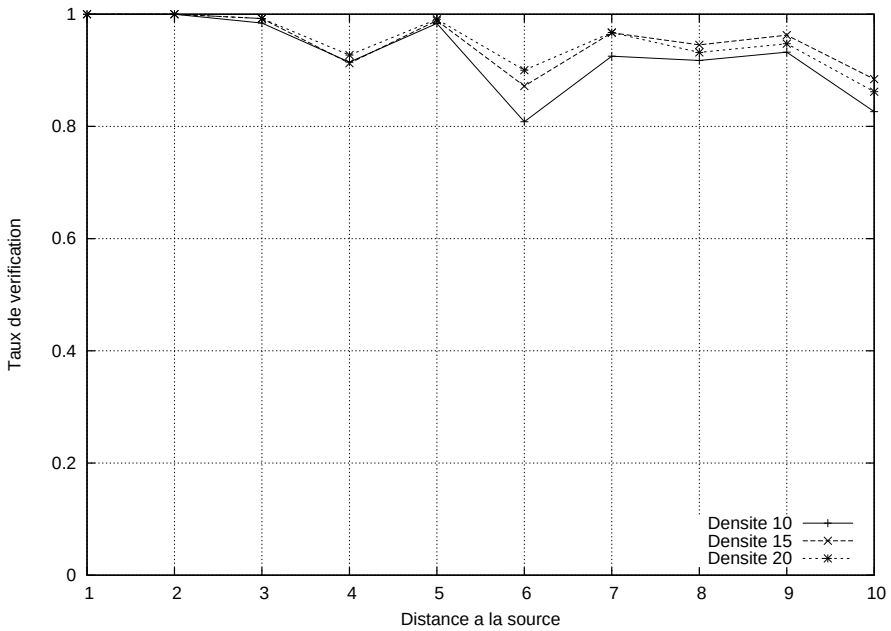
Les Figures 3.13.a et 3.13.b montrent la proportion de nœuds qui mémorisent l'information et le taux de vérification de la propriété donnée par la définition 4 à travers le réseau. Nous pouvons constater que la propriété est bien vérifiée autour de la source et, ce, en utilisant une petite quantité de nœuds mémorisant.

Avec ces résultats, nous pouvons espérer être capables de gérer un plus grand nombre de sources d'information tout en vérifiant la propriété sans utiliser énormément d'entrées dans le cache des nœuds qui mémorisent. La Figure 3.14 montre que ceci est vérifié. En effet, le nombre d'entrées dans le cache des nœuds mémorisant reste faible alors que l'on augmente le nombre de nœuds disséminant une information.

L'influence du facteur d'éloignement λ est mise en évidence par la Figure 3.15. Les résultats sont donnés pour une densité de valeur 15 et des valeurs de λ égales à 0.5, 0.7 et 0.9. La propriété est bien vérifiée pour chacune de ces trois valeurs.



(a) Nombre de nœuds mémorisant l'information



(b) Nombre de nœuds vérifiant la propriété après la dissémination

FIG. 3.13 – *Efficacité de l'algorithme de dissémination.*

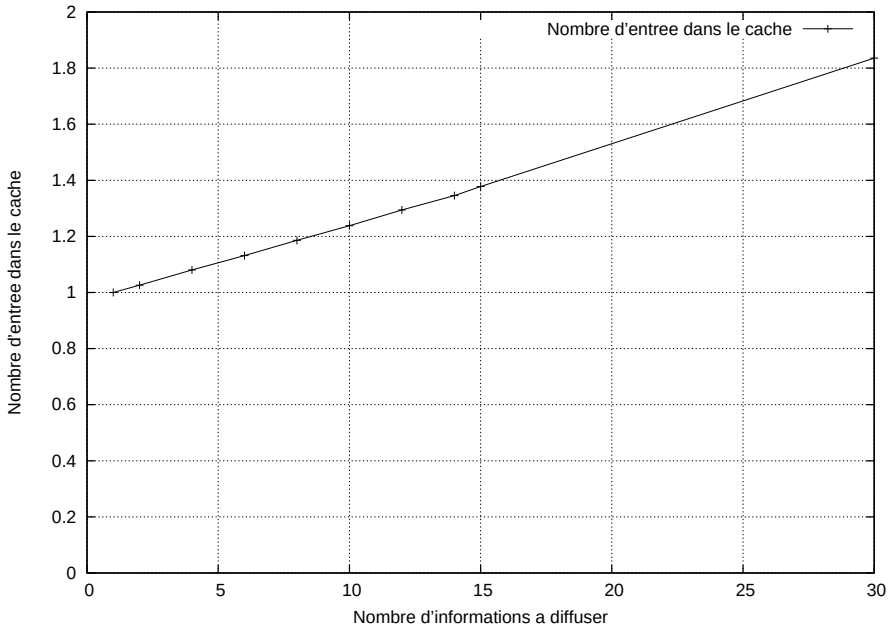


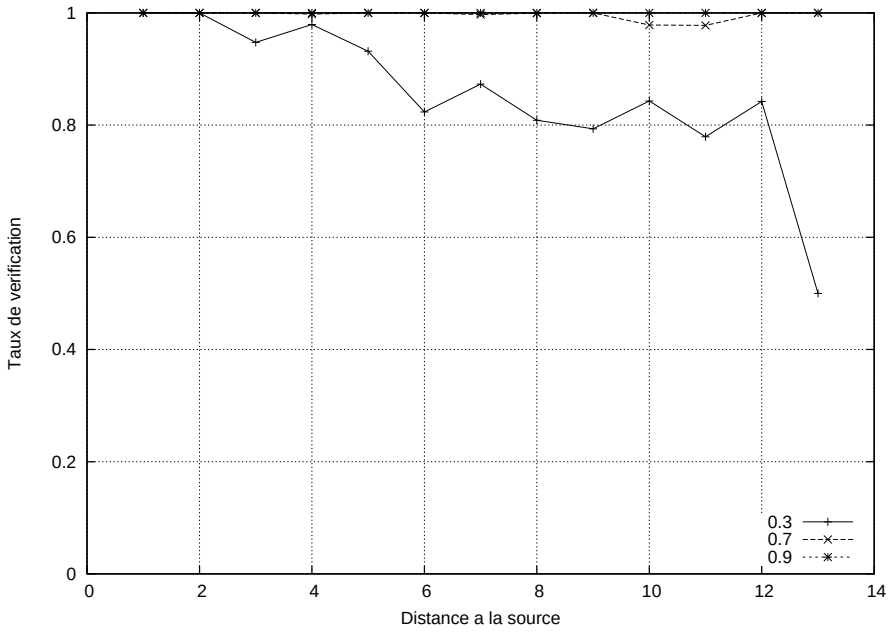
FIG. 3.14 – *Nombre d'entrées moyen dans le cache des nœuds qui mémorisent.*

De plus, le nombre moyen d'entrées dans le cache des nœuds qui mémorisent est plus faible pour des valeurs de λ plus élevées. Le facteur λ influe donc bien sur les besoins en cache des nœuds mémorisant.

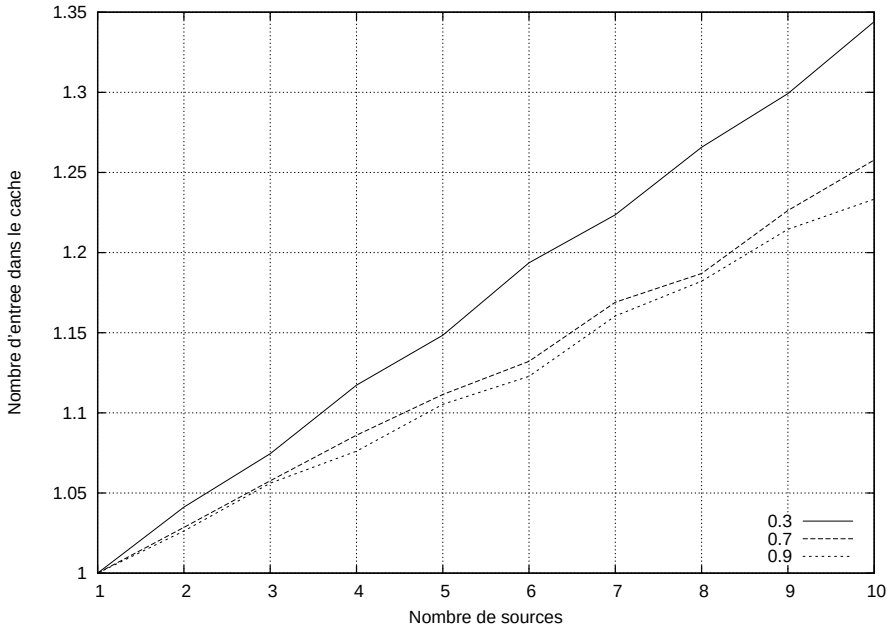
3.4.2 Amélioration avec les ensembles dominants

Pour limiter le fait que deux nœuds proches et à même distance de la source ne mémorisent tous les deux l'information, nous utilisons des ensembles dominants.

Les Figures 3.16.b et 3.16.a montrent que l'utilisation de tels ensembles permet de diminuer le nombre de nœuds nécessaires à la vérification de la propriété pour $\lambda = 0.5$. Les résultats obtenus avec d'autres valeurs de λ restent similaires mais ne sont pas représenté pour des raisons de clarté. La Figure 3.16.c montre que la taille de cache utilisée en moyenne par les nœuds qui mémorisent quand plusieurs sources ont une information à disséminer est moins élevée dans le cas de l'utilisation des ensembles dominants que dans le cas de l'algorithme de base. Ceci reflète que la priorité utilisée pour construire les ensembles dominants (un hash entre l'identifiant de la source et l'identifiant du nœud) répartit bien les nœuds sélectionnés.

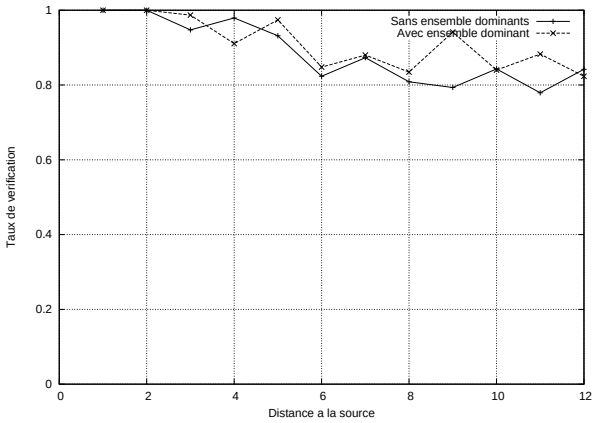


(a) Influence sur la vérification de la propriété

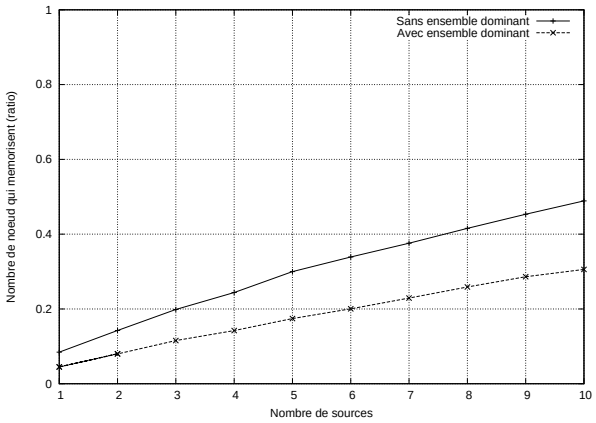


(b) Influence sur la taille de cache nécessaire

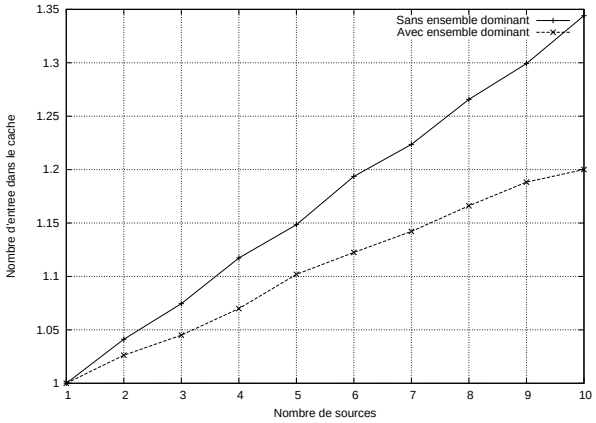
FIG. 3.15 – *Influence de* λ.



(a) Influence sur la vérification de la propriété



(b) Influence sur le nombre de nœuds utilisés



(c) Influence sur la taille de cache utilisée par les nœuds qui mémorisent

FIG. 3.16 – *Amélioration obtenue avec l'utilisation des ensembles dominants.*

3.5 Conclusion

Les expérimentations concernant l'algorithme de base montrent que le critère est bien respecté dans un graphe de densité non infinie. De plus, lorsque plusieurs sources disséminent une information, nous observons que la taille de cache moyenne utilisée par les nœuds qui mémorisent n'est pas très élevée, ce qui met en évidence que ce ne sont pas toujours les mêmes nœuds qui sont pénalisés.

Les expérimentations concernant les ensembles dominants montrent que le nombre de nœuds mémorisant est encore diminué. Le critère est toujours respecté et, à nouveau, quand plusieurs nœuds disséminent une information, la taille de cache moyenne utilisée par les nœuds qui mémorisent n'est pas très élevée, ce ne sont donc pas toujours les mêmes nœuds qui sont pénalisés.

En analysant les résultats obtenus lors des simulations, nous avons été confortés dans notre intuition première : le facteur d'éloignement λ influe sur l'utilisation du cache des objets. En effet, plus la valeur de λ est élevée, plus le nombre de nœuds mémorisant nécessaires à couvrir la totalité de la zone diminue. Il apparaît donc naturel que les nœuds formant le réseau adaptent leur facteur d'éloignement λ en fonction de la taille de cache dont ils disposent. Ne disposant plus d'un facteur d'éloignement global à partir duquel vérifier le respect de la propriété donnée par la Définition 4, pour un nœud, vérifier la propriété, signifie la respecter en considérant le facteur d'éloignement local.

L'idée serait alors de déterminer la valeur de λ adaptée à la taille de cache de l'objet. À chaque taille de cache va correspondre un λ et donc une table de probabilités en fonction de la distance à la source. Chaque nœud utilise alors la table associée à sa taille de cache. Cette adaptation, non évaluée, permet d'envisager un algorithme encore plus efficace.

Chapitre 4

Prédiction de partitionnement

« Oh ! Attention chérie ça va couper ! »

La cité de la peur, Projectionniste anonyme.

Dans ce chapitre, nous nous intéresserons aux méthodes de prédiction de partitionnement du réseau. Plus particulièrement, nous proposerons des méthodes ne nécessitant pas d'information de positionnement. Nous présenterons tout d'abord les objectifs de tels algorithmes ainsi que des exemples d'applications et les travaux existants dans ce domaine. Puis, nous présenterons nos contributions et en ferons l'évaluation.

4.1 Objectifs

Les algorithmes de prédiction de partitionnement visent à l'amélioration de la durée de vie d'un service par la détection de rupture de la connexité du réseau. En effet, si la rupture physique de la connexion est considérée comme une panne dans les réseaux filaires, dans le cadre des réseaux ad hoc, c'est un événement normal du système. Comme l'illustre la Figure 4.1, la mobilité des nœuds, ainsi que la portée limitée de leur communication implique la possibilité de voir disparaître toute possibilité de trouver une route entre l'utilisateur et le fournisseur d'un service donné.

Si on pouvait fournir aux applications la possibilité de détecter un tel événement, celles-ci pourraient adapter leur comportement afin de maintenir le service [CCB02]. Plusieurs réactions à un tel événement sont envisageables. En fonction de la nature du service, il est même tout à fait possible de fournir plusieurs types de comportement. Ces derniers vont de la duplication du service sur un nœud plus fiable, à la recherche d'un autre nœud fournissant le même service, en passant par le renforcement des routes par le déplacement forcés d'autres nœuds (ceci

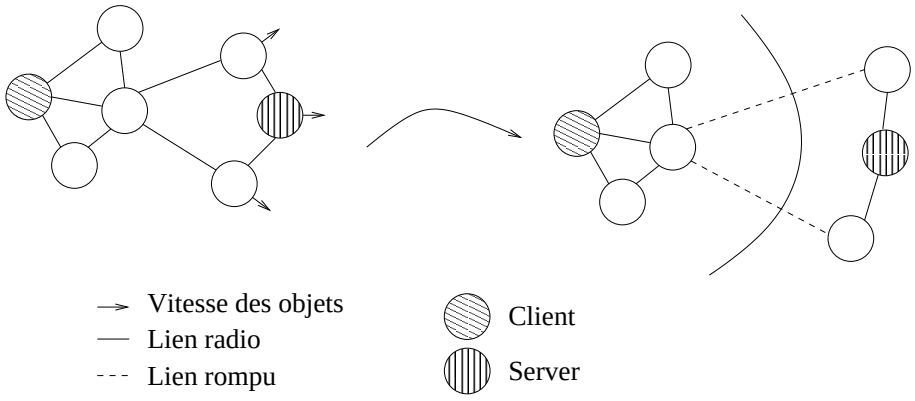


FIG. 4.1 – *Partitionnement du réseau dû à la mobilité.*

constituant tout de même une solution peu envisageable dans le cadre d'un réseau grand public) [BR04]. L'étude de politique de réaction face aux partitionnement du réseau peut faire l'objet de travaux futurs mais ne sera pas abordée dans ce mémoire qui se focalise sur la prédiction de ces partitionnements.

La section suivante présente les travaux existants en matière de prédiction de partitionnement.

4.2 Travaux existants

Dans [PC97], Park *et al.* présentent TORA, un protocole de routage adaptatif pour les réseaux ad hoc assurant une maintenance des routes. Dans ce protocole, les auteurs utilisent une méthode de détection de partition du réseau *après coup* qui permet de nettoyer les tables de routages des entrées concernant les nœuds qui ne sont plus atteignables. Ce protocole détecte donc la partition du réseau après que celle-ci soit arrivée. Cette approche ne permet donc pas de réagir en avance à une future déconnexion. Un service disponible sur un nœud qui devient non atteignable deviendra donc inutilisable par les nœuds qui sont en train de l'utiliser. Nos travaux se focalisent sur la prédiction des partitionnements du réseau afin de prévenir les applications de la *future* non disponibilité des services qu'elles sont en train d'utiliser.

Dans [SCN01], Shah *et al.* cherchent à améliorer l'accès aux données dans un réseau ad hoc en détectant les partitionnements du réseaux avant qu'ils n'arrivent. Dans leur scénario, un nœud n_1 nécessite des informations qui sont disponibles sur un autre nœud situé quelque part dans le réseau (notons le n_2). Afin de permettre à n_1 d'accéder aux données situées sur n_2 même si la connexion avec ce dernier casse physiquement, les auteurs proposent un mécanisme de réplication des données basé sur un système de prédiction de partitionnement. Chaque nœud embarque un système de positionnement de type GPS [gps97] et calcule sa vi-

tesse par mesures successives de sa position. Périodiquement, chaque nœud diffuse cette information dans le réseau. Donc, chaque nœud appartenant à une même composante connexe connaît le comportement de ses voisins concernant la mobilité. Ils peuvent donc prédire, par interpolation de trajectoire, à quel moment un nœud hébergeant une information donnée va quitter la composante avant qu'il le fasse vraiment. Le propriétaire de l'information procède alors à l'élection d'un nouvel hôte susceptible de recevoir le réplicat ainsi qu'à la réplication effective sur ce dernier. L'avantage principal de cette méthode est que chaque nœud connaît exactement *quand* le partitionnement du réseau va arriver si les mouvements des nœuds sont réguliers (pas de changement brutal de direction). Par contre, elle a au moins deux inconvénients. Tout d'abord, elle suppose que les objets soient équipés d'un système de positionnement qui peut être souvent encombrant, cher et fortement consommateur d'énergie. Plus important, la mise à jour des trajectoires auprès des nœuds du réseau entraîne une constante et importante charge du réseau si l'on veut maintenir des informations cohérentes. De plus, l'interpolation des trajectoires pourrait s'avérer être une opération lourde pour de très petits objets (téléphones, capteurs...).

Wang *et al.* proposent une approche similaire dans [WL02]. Néanmoins, leur proposition est plus centralisée et est basée sur une extension du *Reference Point Group Mobility* (RPGM) par Hong *et al.* [HGPC99]. Les nœuds embarquent un système de positionnement et envoient leur position et leur vitesse à un serveur central qui les rassemble en groupe en fonction de leur caractéristique de mobilité (position, vitesse). Le serveur connaît alors la façon dont se déplacent chaque groupe et peut prédire les partitionnement du réseau et avertir les nœuds concernés. Les avantages et inconvénients d'une telle méthode sont équivalents à ceux de la méthode citée précédemment avec pour inconvénient supplémentaire l'utilisation d'un serveur central qui devient alors un point critique du réseau, tant au niveau de la charge réseau qu'au niveau disponibilité.

Malgré les bons résultats que peuvent donner de telles méthodes, les contraintes qu'elles entraînent rendent impossible leur utilisation sur tout type d'objet. En effet, ces méthodes sont basées sur l'utilisation d'un système de positionnement et/ou sur la nécessité d'avoir un nœud joignable par tous les autres. Il serait préférable de fournir un algorithme de prédiction pouvant fonctionner sur tout type d'objets munis uniquement d'une interface de communication sans fil, ce qui est la contrainte minimum que nous voulons nous imposer. Nous allons donc nous intéresser à un système complètement distribué qui n'utilise pas d'outil de positionnement. Cela rendra notre travail utilisable dans toutes les configurations possibles de réseau ad hoc car nous n'imposons que l'utilisation d'une interface de communication sans fil ce qui est la définition même d'un réseau ad hoc tel que nous l'étudions dans ce mémoire.

4.3 Définition générale du système de prédiction

Afin de déterminer une architecture efficace pour la prédiction de partitionnement, nous devons tout d'abord nous pencher sur la définition précise du scénario pour lequel notre solution a été pensée. La prédiction trouve son utilité dans le cadre d'un service continu ou une application cliente reste en communication avec une application serveur pendant un certain laps de temps. Il est donc inutile de chercher à connaître de façon globale les déconnexions pouvant survenir dans tout le réseau. Seules les ruptures intervenants sur les chemins disponibles entre le client et le serveur vont nous intéresser. De plus, il est préférable de laisser l'évaluation au client car le serveur ne peut se permettre d'effectuer des mesures qui peuvent être coûteuses pour chacun de ses clients (qui peuvent être nombreux).

Nous proposons donc d'effectuer des mesures sur la ou les routes disponibles entre le serveur et le client à intervalles réguliers. Si une faiblesse est détectée, nous pouvons alors prévenir les deux applications qui pourront alors adapter leur comportement en fonction de leur politique de réaction face aux ruptures du réseau.

Nous pouvons alors, grâce à cette petite analyse, proposer un algorithme générique répondant à nos exigences (Figure 4.2) qui sera exécuté périodiquement par le nœud client.

```
Évaluer la robustesse du lien entre le client et le serveur
Si cette robustesse est inférieure à un seuil donné
  Envoyer une alerte à l'application
```

FIG. 4.2 – *Algorithme générique de prédiction de partitionnement.*

L'algorithme de base est très simple mais possède néanmoins un point sombre et peu trivial concernant l'évaluation de la robustesse du lien. C'est cette évaluation que nous allons étudier dans la suite de ce chapitre. Nous proposerons tout d'abord une évaluation basée sur des ensembles de chemins nœud-disjoints puis sur la notion de liens critiques et de nœuds critiques. Enfin, nous évaluerons les performances de ces trois méthodes d'évaluations.

4.4 Évaluation de la robustesse du lien par ensembles de chemins nœud-disjoints

4.4.1 Description de la méthode

La première méthode d'évaluation que nous proposons se base sur l'idée suivante. Plus il y aura de chemins existant entre le client et le serveur, moins les

chances de rupture de la connexion sont importantes. En effet, si un des chemins disparaît, la communication sera toujours possible en passant par un autre. La Figure 4.3 illustre cette idée grâce à deux topologies.

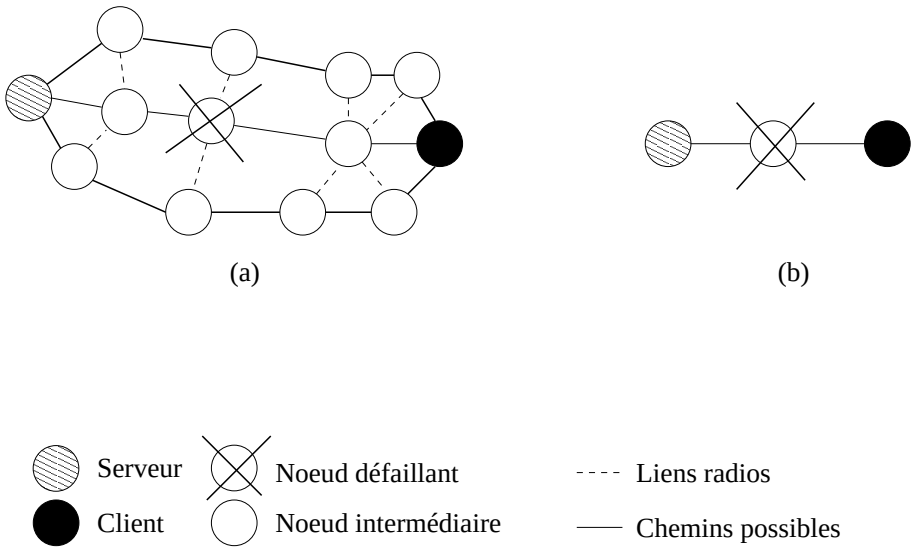


FIG. 4.3 – *Exemple de topologies.*

Dans le premier cas, Figure 4.3.a, le lien est fiable. En effet, si un nœud disparaît, il reste encore deux routes possibles pour faire transiter les paquets. Par contre, dans le deuxième cas, Figure 4.3.b, si il ne reste qu'un chemin et que celui-ci cède, la communication est définitivement rompue.

Néanmoins, il nous faut utiliser un sous ensemble de $SOP_k(v,w)$ (voir Définition 3 page 24). Cet ensemble est l'ensemble des ensembles de chemins nœud-disjoints de longueur inférieure ou égale à $d(v,w) + k$ entre v et w (où v et w sont le serveur et le client) noté $DSP_k(v,w)$ (*vertex-Disjoint Sub-optimal Paths*) et défini par (on conserve ici les notations définies dans la Section 2.2.3 page 23) :

$$DSP_k(v,w) = \left\{ S \in 2^{SOP_k(v,w)} \mid \quad \forall p,p' \in S \quad \tilde{p} \cap \tilde{p}' \neq \{v,w\} \Rightarrow p = p' \right\}. \quad (4.1)$$

En d'autres termes, un ensemble de chemins nœud-disjoints est un ensemble de chemins qui n'ont en commun que la source et la destination. L'intérêt de l'utilisation d'un tel ensemble est que si un nœud disparaît, un et un seul chemin sera rendu inutilisable.

Ainsi, cet ensemble permet de représenter avec une certaine fiabilité le fait que le lien entre le client et le serveur est fort ou non. En effet, s'il y a de nombreux chemins nœud-disjoints entre les deux nœuds, il faudra alors qu'un nombre important de nœuds cessent de participer au routage des paquets entre le serveur

et le client (parce qu'ils se sont déplacés ou qu'ils se sont éteints) pour que la communication soit rompue.

La valeur de l'évaluation de la robustesse du lien sera alors le nombre de chemins nœud-disjoints qu'il existe entre le client et le serveur. S'il n'en reste qu'un, le lien devient faible car la disparition d'un seul nœud peut entraîner la rupture de la connexion entre le serveur et le client. La Figure 4.4 montre l'évolution du nombre de chemins disjoints entre le client et le serveur durant une connexion. On voit bien que ce nombre chute progressivement jusqu'à tomber à 1 peu avant la rupture effective de la connexion.

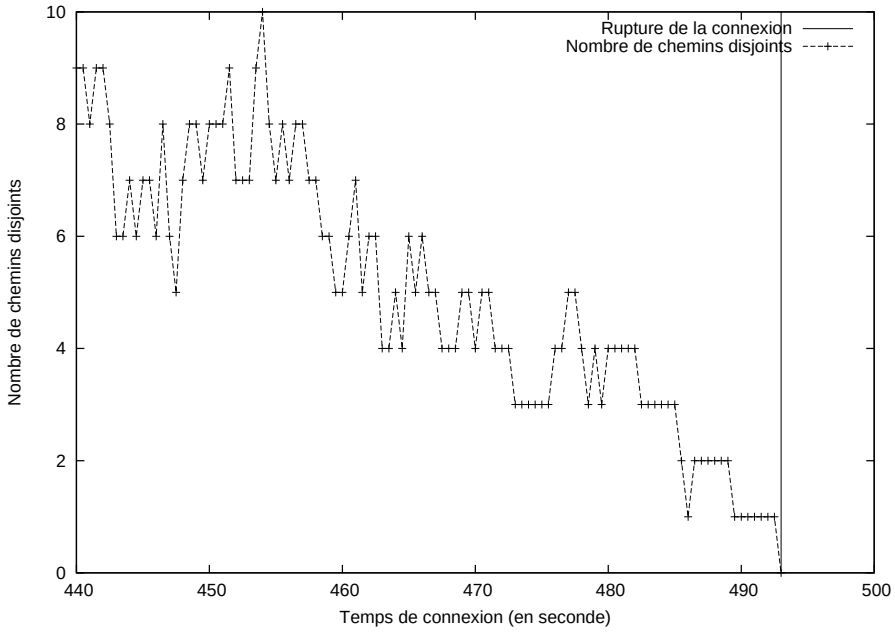


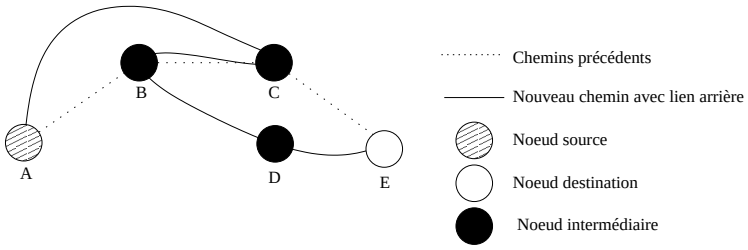
FIG. 4.4 – *Évolution du nombre de chemins disjoints entre deux nœuds.*

4.4.2 Génération des ensembles de chemins disjoints

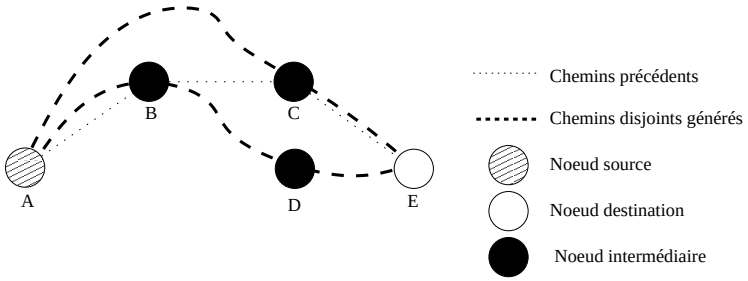
Afin de pouvoir utiliser la méthode citée précédemment, il nous faut pouvoir générer un ensemble de chemins nœud-disjoints existant entre le client et le serveur. Le problème est, en effet, non trivial et est comparable aux travaux sur le routage multi-chemins pour la qualité de service [DMB+03, PHS02, LS02, LG01, LLP+01]. Dans ce contexte, le but est de générer le maximum de chemins possible lors de la recherche de route afin de pouvoir obtenir une qualité de service maximum grâce à la distribution des paquets sur les différentes routes disponibles.

4.4.2.1 Travaux existants

Dans [PHS02], Haas *et al.* proposent un algorithme distribué pour calculer un ensemble de chemins disjoints. Ils utilisent une méthode de construction itérative qui fusionne un ensemble de chemins précédemment obtenu avec un nouveau chemin qui contient un ou plusieurs liens *arrière* (*i.e.* un lien qui va de la destination à la source, voir Figure 4.5). Quand un nœud reçoit un nouveau chemin, il le compare à l'ensemble des chemins dont il dispose. Si un des liens composant le chemin appartient également à un des chemins de l'ensemble précédent, on peut modifier les deux chemins afin de créer deux chemins disjoints. Ceci ne peut être fait que si le lien du nouveau chemin va dans le sens inverse du lien de l'ancien chemin. Dans l'exemple donné Figure 4.5, le lien entre les nœuds B et C est présent dans le nouveau chemin dans le sens $C \rightarrow B$ et dans l'ancien chemin dans le sens $B \rightarrow C$. Le chemin précédent est alors enlevé de l'ensemble et est remplacé par les deux chemins générés par la fusion. En l'occurrence, ici, le chemin ACBDE est remplacé par ACE et ABDE.



(a) Nouveau chemin



(b) Ensemble généré

FIG. 4.5 – *Construction itérative d'ensemble de chemins disjoints.*

Néanmoins, dans le contexte de notre étude, ce genre de routes (celles contenant un lien *arrière*) n'est pas généré par les algorithmes de recherche de routes classiques. Ces derniers sont le plus souvent basés sur une diffusion aveugle (*blind flooding* [PR99, TNC02]) où un numéro de séquence est utilisé pour empêcher un nœud de retransmettre un même paquet plusieurs fois. Le principal problème du protocole de Haas, vis-à-vis de ce genre de génération de routes, est qu'obtenir

un ensemble de chemins contenant des liens en commun est impossible avec un seul paquet de recherche. En effet, si on observe la Figure 4.5, on en déduit que quand le premier chemin (ABCE) est généré, B et C vont stocker le numéro de séquence associé avec le paquet de recherche de route. Quand une autre instance de ce même paquet de recherche va arriver pour la seconde fois au nœud C, il va ignorer ce dernier car il a déjà retransmit cette requête. Cela entraîne alors l'impossibilité de générer le chemin ACBDE. Pour utiliser cette méthode de recherche de chemins disjoints, il faudrait donc utiliser plusieurs requêtes ce qui pose deux problèmes majeurs. En effet, multiplier les requêtes aura pour effet d'augmenter sensiblement le temps de génération de l'ensemble de chemins disjoints et surtout de générer une charge réseau importante.

Plusieurs requêtes n'étant pas acceptable pour générer notre ensemble de chemins disjoints, il nous faut alors nous tourner vers un algorithme qui va augmenter légèrement le nombre de paquets générés afin de créer plus de chemins sans congestionner le réseau de façon trop importante.

Dans [DMB+03], Das *et al.* utilisent un algorithme de diffusion qui n'est plus basé sur un numéro de séquence, mais sur le chemin actuellement parcouru par le paquet de recherche. On note respectivement s et d les nœuds source et destination. Le paquet de recherche de route contient un nombre de saut qui est décrémenté à chaque retransmission et la liste des nœuds constituant le chemin que le paquet suit. Quand un nœud reçoit un paquet, il exécute l'algorithme suivant :

1. Si le nœud est la destination, il génère un paquet de réponse et l'envoie à la source. Le paquet suit alors le trajet pris par le paquet de recherche,
2. Si le nombre de saut atteint 0, le paquet est ignoré,
3. Si le nœud est déjà présent dans le chemin suivit par le paquet, ce dernier est également ignoré pour éviter les boucles,
4. Si aucune des conditions précédentes n'est vérifiée, l'identifiant du nœud est ajouté dans le paquet et celui ci est retransmis.

Comme le paquet n'est ignoré qu'en cas de boucle ou quand le nombre maximum de sauts est atteint, le protocole est plus apte à générer beaucoup de chemins car une même requête peut être retransmise plusieurs fois par le même nœud. Tous les chemins sans boucle de longueur inférieure ou égale au nombre maximum de saut seront découvert par ce protocole car seuls les tests de longueur et de boucle sur le chemin empêche la retransmission du paquet de recherche.

Le problème principal de cet algorithme est que le nombre de paquet généré explose avec la distance entre la source et la destination. En effet, étant données les règles de retransmission du paquet, il est clair que la complexité en nombre de paquet généré est $\mathcal{O}(d^n)$ où n est le nombre de saut entre la source et la destination

et d est la densité moyenne du réseau. Ce protocole ne peut alors être utilisé que dans des réseau de faible densité et de faible diamètre.

Lee *et al.* ont proposé une extension de la diffusion aveugle dans [LG01]. Ils utilisent toujours un numéro de séquence pour éviter les rediffusions inutiles mais ils rajoutent une condition sur l'émetteur direct du paquet afin de rendre moins restrictive la règle de non retransmission. Si un paquet de même numéro de séquence à déjà été retransmit, le nœud ayant reçu le paquet regarde si ce dernier provient d'un voisin ayant déjà émit un paquet avec le même numéro de séquence. Si oui, le paquet est ignoré, sinon il est retransmit. Un nœud recevant un paquet de recherche de chemin exécute alors l'algorithme suivant :

1. Si le nœud est la destination, il génère un paquet de réponse et l'émet par la route par laquelle le paquet de recherche est arrivé,
2. Si un paquet de même numéro de séquence et du même émetteur à déjà été reçu, le paquet est ignoré,
3. Sinon, l'identifiant du nœud est ajouté dans le paquet de recherche et celui ci est retransmis.

La complexité en nombre de retransmissions de l'algorithme dépend alors du nombre de voisins de chaque nœud est peut donc être estimée à $\mathcal{O}(d)$ où d est la densité moyenne du réseau.

4.4.2.2 Notre proposition

Le protocole que nous proposons va permettre de diminuer le nombre de retransmissions tout en améliorant le nombre de chemins disjoints découverts. De plus, la requête est exécutée en une seule diffusion ce qui assure un temps raisonnable pour une utilisation dans le cadre de prédiction de partitionnement. Pour ce faire, nous étendons également la diffusion aveugle en l'améliorant pour notre problème par deux mécanismes. En effet, ce protocole de diffusion possède deux inconvénients majeurs. Tout d'abord, comme nous l'avons dit précédemment, il ne génère pas assez de chemins. La robustesse déduite serait alors beaucoup plus faible que la robustesse réelle ce qui entraînerait une pauvre fiabilité du mécanisme de prédiction. De plus, la zone de congestion induite s'étend sur tout le réseau, y compris sur les nœuds qui n'interviennent pas dans la communication entre le client et le serveur. Comme nous allons devoir augmenter le nombre de retransmissions du paquet par chaque nœud, il est important de limiter la congestion à une zone restreinte.

Le premier mécanisme mis en place concerne l'augmentation du nombre de chemins générés. Lee *et al.* proposent de faire environ d retransmissions par nœud (où d est la densité moyenne du réseau). Nous pensons qu'il est possible de trouver plus de chemins en contrôlant de manière plus fine la retransmission. Le problème

de la diffusion aveugle est que le principe du numéro de séquence empêche un nœud d'appartenir à plusieurs routes. Même si, dans l'absolu, le fait qu'un nœud appartienne à plusieurs routes est en contradiction avec la définition des ensembles de chemins nœud-disjoints, il est tout de même important de ne pas éliminer trop tôt des routes potentielles. En effet, il est probable, voire même fréquent, qu'une route ne soit jamais trouvée.

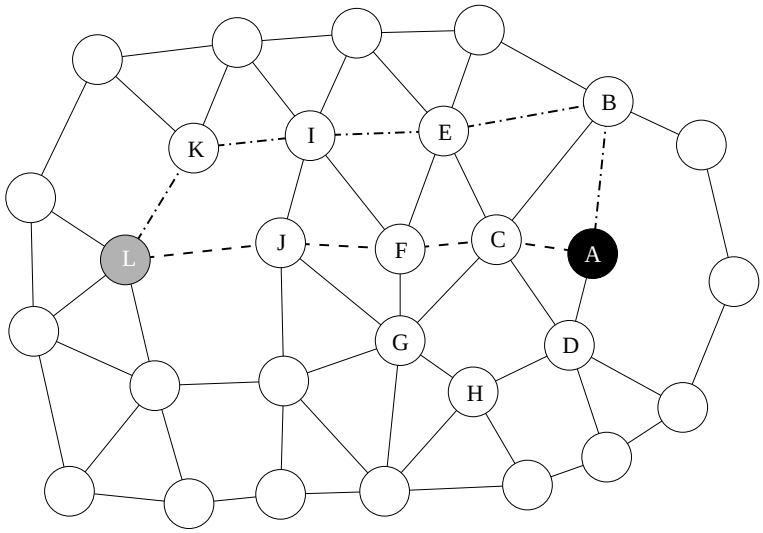


FIG. 4.6 – *Limites de la diffusion aveugle pour la génération de chemins nœud-disjoints.*

La Figure 4.6 nous montre un cas dans lequel deux routes peuvent ne pas être découvertes avec une diffusion simple. Si nous exécutons le protocole pour découvrir des routes du nœud A au nœud L, A va envoyer un message de découverte de route qui va être reçu par B, C et D. Si C est le premier à retransmettre le paquet, les nœuds B,D,E,F,G et H vont recevoir le paquet et vont donc mémoriser son numéro de séquence. De par ce fait, quand B va retransmettre le message reçu de A et que E va le recevoir, comme ce dernier à déjà reçu un paquet avec le même numéro de séquence, il va ignorer ce paquet. Ce qui empêche la découverte de la route $ABEIKL$ qui est pourtant intéressante d'un point de vue ensemble de route nœud-disjointes. Si par contre, B retransmet le paquet de A avant C, il est tout à fait possible que le problème soit reporté plus loin dans l'avancement du paquet de recherche. Il en va de même pour la partie inférieur du réseau (la route commençant par D). Les solutions de Lee ou Das permettent d'éviter ce problème. Néanmoins, leurs solutions induisent beaucoup de retransmissions (surtout dans le cas de Das).

Afin de générer plus de routes, il faudrait donc permettre de diversifier la diffusion en redonnant une chance à certains nœuds de réintégrer le processus de

recherche. Pour cela, nous proposons une solution originale basée sur l'utilisation de numéros de séquence multiples. Un nœud désirant effectuer une recherche de route va envoyer un paquet de recherche en utilisant un protocole de diffusion classique basé sur un numéro de séquence. À partir d'une certaine distance, nous proposons de changer dynamiquement le numéro de séquence afin d'introduire un renouvellement dans la diffusion. Mais, pour permettre de ne pas trop encombrer le réseau, le nouveau numéro de séquence ne sera pas choisi au hasard. Le nœud désirant effectuer sa recherche va intégrer au paquet une liste de nouveaux numéros de séquence, différents de celui utilisé pour diffuser le paquet. Au moment de changer de numéro, les nœuds vont choisir aléatoirement le nouveau numéro parmi la liste. Cela aura donc pour effet de redonner leur chance à des routes qui aurait été ignorée par une diffusion aveugle.

Même si nous tentons de limiter au maximum le nombre de retransmissions, la recherche d'ensemble de routes disjointes reste plus coûteuse qu'une simple diffusion. C'est pourquoi il faut tenter de limiter cette recherche dans la zone du réseau qui nous intéresse. En effet, il est inutile d'impliquer des nœuds dans la recherche de chemins si ils sont trop éloignés du client ou du serveur. Nous introduisons alors un deuxième mécanisme dans notre protocole de recherche de chemins qui va permettre de diriger et de confiner la diffusion. Pour cela, nous avons besoin d'information de distance entre les nœuds source, destination et le reste du réseau. Ces informations peuvent être obtenues de plusieurs manières comme par exemple grâce à des protocole de diffusion optimisés [1][ISRS04]. Chaque nœud sera alors en mesure de connaître les informations nécessaires à l'exécution de notre protocole.

La recherche de chemins se fait en deux étapes. Premièrement, le client envoie une requête au client grâce à une diffusion optimisée. Les nœuds recevant cette requête retiennent alors la distance les séparant du client (en nombre de sauts). Quand le serveur reçoit ce paquet, il génère une réponse. Le paquet de réponse contient les informations suivantes :

- l'identifiant du serveur ;
- l'identifiant du client ;
- la distance entre le serveur et le client (en nombre de sauts). Cette distance est obtenue à partir du paquet de requête ;
- le nombre de sauts actuellement parcouru par le paquet (initialisé à 0) ;
- un numéro de séquence, utilisé pour limiter la diffusion du paquet de réponse (par un algorithme de diffusion aveugle classique) ;
- un ensemble de numéro de nouveaux numéros de séquence, générés par le serveur à l'émission du paquet de réponse ;

1. dont le rôle sera alors de récupérer les informations nécessaires en diminuant au maximum le nombre de paquet nécessaire

– le chemin suivi par le paquet.

Chaque nœud u recevant le paquet de réponse exécute l'algorithme suivant. On pose alors $d(c,s)$ la distance séparant le serveur du client, h le nombre de sauts parcourus par le paquet de réponse au moment de sa réception par u et $d(c,u)$ la distance séparant le client du nœud u (obtenue à partir de la diffusion du paquet de requête) :

1. si u a déjà retransmis un paquet de même numéro de séquence, le paquet est ignoré,
2. si $h+d(c,u) > d(c,s)+k$ (ou k est un paramètre du protocole), le paquet est ignoré. Ceci constitue l'implantation du deuxième mécanisme permettant de confiner la diffusion de la réponse,
3. si $h = \frac{d(c,s)}{2}$, *i.e.* si le nombre de sauts effectués par le paquet est égal à la moitié de la distance séparant le client du serveur, le nœud choisi de façon aléatoire un nouveau numéro de séquence parmi ceux fournis dans le paquet. Le paquet est alors retransmis avec ce nouveau numéro de séquence.

La première règle de ce protocole correspond à la diffusion aveugle classique, basée sur un numéro de séquence. La deuxième constitue l'application du deuxième mécanisme que nous avons cité plus haut. Le but est alors de cibler le client lors de la réponse afin de ne pas encombrer la totalité du réseau par une diffusion qui reste relativement lourde. On peut voir Figure 4.7, où les lignes épaisses représentent les liens qui ont effectivement été utilisés pour la diffusion, que seule une petite partie du réseau est perturbée par notre protocole.

La troisième et dernière règle permet d'augmenter le nombre de retransmission par nœud afin de générer plus de routes. Comme le serveur choisi le nombre de numéros de séquence qui vont être utilisés, il peut contrôler la congestion générée par le protocole en fonction des capacités du canal. De même le paramètre k permet de limiter l'étendue de la congestion. Plus k est grand, plus le nombre de chemins généré va être grand et plus ces chemins pourront être long. Par contre, si k est grand, il y aura plus de nœuds impliqués dans la diffusion et donc un encombrement du réseau plus important.

4.4.3 Évaluation du protocole de recherche de chemins nœud-disjoints

Afin de vérifier l'efficacité du protocole de recherche de routes, nous l'avons comparé aux protocoles existants. Nous avons évalué le nombre de retransmissions par nœud ainsi que le nombre de chemins générés. Nous présentons uniquement une comparaison avec le protocole de Lee *et al.* [LG01]. La raison pour laquelle nous n'analysons par le protocole donné par Das *et al.* dans [DMB+03] est que,

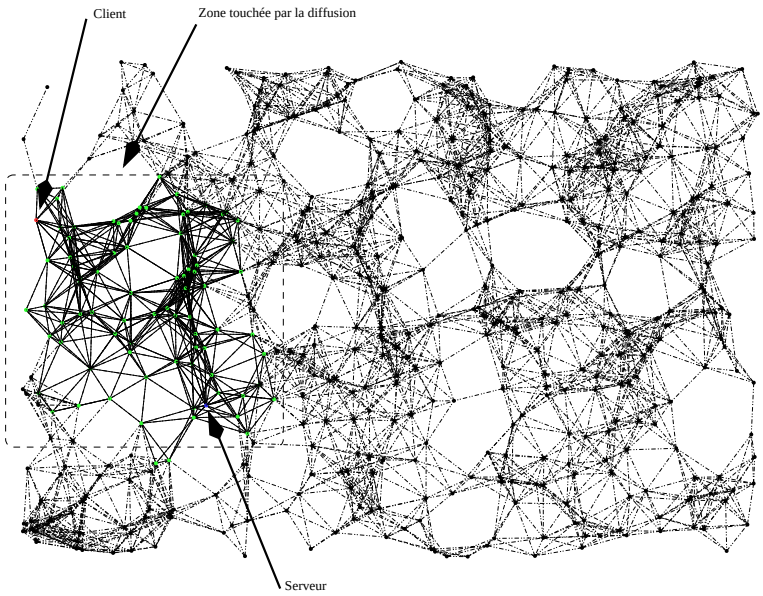


FIG. 4.7 – *Diffusion confinée.*

comme nous l'avons précisé plus haut, sa complexité en terme de retransmissions est en $\mathcal{O}(d^n)$ ce qui rend la simulation quasi impossible dans des temps raisonnables pour une densité supérieure à cinq nœuds par zone de communication.

Nous avons effectué les mesures à différentes densités en utilisant un graphe de 200 nœuds situé dans une zone rectangulaire de 600 mètres par 400 mètres. La portée de communication des nœuds est adapté afin d'obtenir la densité souhaitée.

Afin d'obtenir des résultats pertinents, nous effectuons l'expérience suivante :

1. un graphe aléatoire est généré en fonction de la densité souhaitée,
2. deux nœuds sont choisis pour être le client et le serveur. Nous les choisissons de façon à ce qu'il existe au moins 3 chemins nœuds disjoints entre eux (propriété vérifiée à l'aide d'un algorithme de type recherche en profondeur). Si on ne peut vérifier la propriété, on génère un nouveau graphe jusqu'à pouvoir ce placer dans la situation qui nous intéresse,
3. les deux protocoles sont simulés et on retient le nombre de retransmissions par nœud ainsi que le nombre de chemins générés,
4. les étapes précédentes sont répétées pour 500 graphes et on observe la moyenne des résultats.

Le paramètre k utilisé est 5 et le nombre de numéros de séquence inclus dans le paquet de réponse est égal à la densité moyenne du réseau. Les résultats de ces expériences sont donnés Figures 4.8, 4.9 et 4.10.

La Figure 4.8 donne le nombre de chemins générés par les différents protocoles. Cette dernière montre donc clairement que notre protocole est efficace en terme

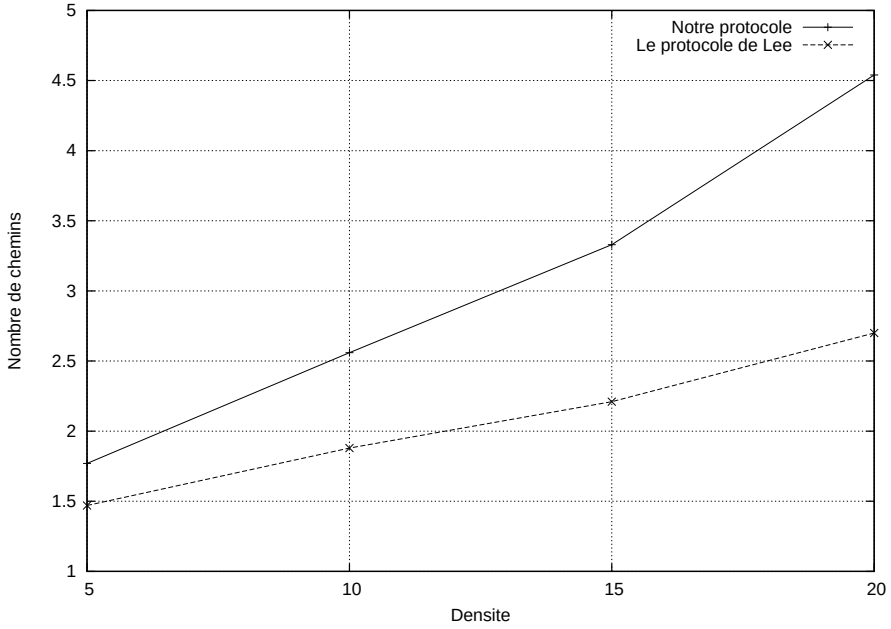


FIG. 4.8 – *Nombre de chemins nœud-disjoints générés.*

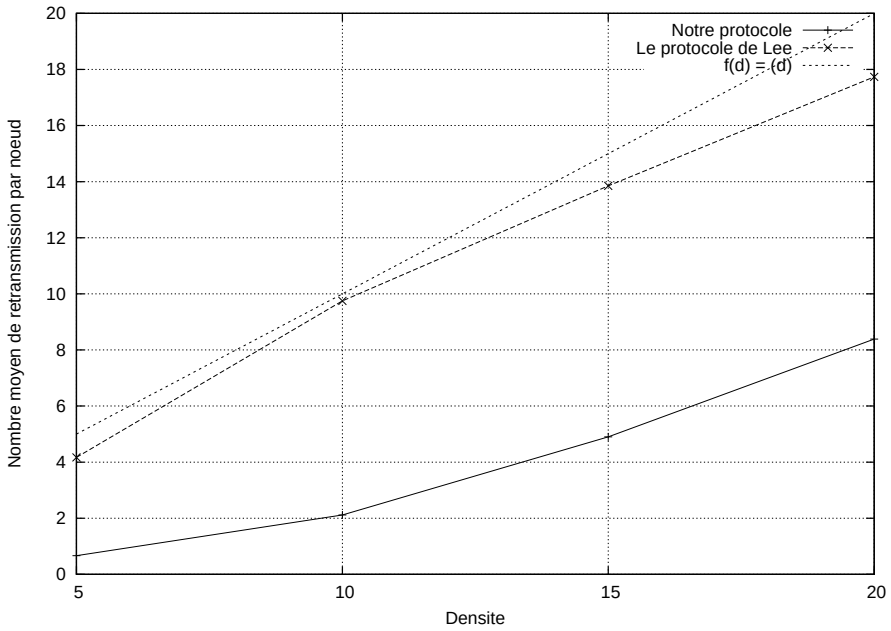


FIG. 4.9 – *Nombre moyen de retransmission par nœud.*

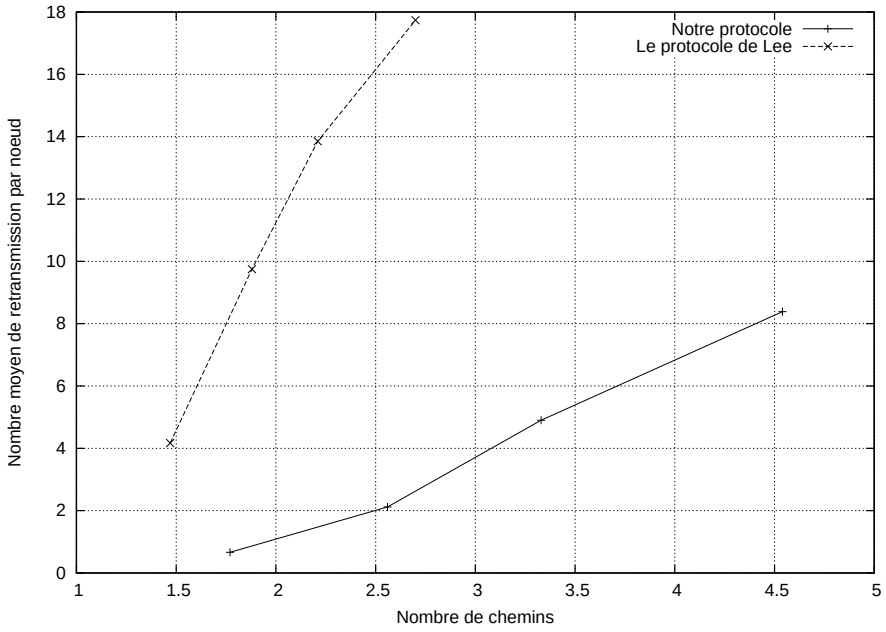


FIG. 4.10 – *Nombre moyen de retransmission par nœud sur nombre de chemins nœud-disjoints générés.*

de quantité de chemins trouvés en une passe requête/réponse. Notre protocole permet même de découvrir plus de chemins que celui de Lee.

La Figure 4.9 montre quant à elle une mesure de la charge du réseau en nombre de retransmissions moyen par nœud. On peut voir que cette courbe confirme notre estimation sur la complexité du protocole de Lee. En effet, la courbe illustrant le nombre de retransmissions de leur protocole est quasiment équivalente à la fonction $f(d) = d$. Compte tenu du nombre de numéros de séquence que nous avons intégré dans le message, le nombre de retransmissions de notre protocole est *au maximum* égal d car les d numéros de séquence intégrés au paquet (qui vont donc être utilisés à partir de la moitié de la distance entre le client et le serveur) seront tous sélectionnés uniquement dans le pire des cas. De plus, la limitation de la zone de diffusion de notre protocole permet d'obtenir d'excellents résultats si on considère l'ensemble du réseau. L'écart de performance entre les deux protocoles est encore plus clairement illustrée par la Figure 4.10.

Pour cette expérience, nous avons choisi le paramètre k et le nombre de numéros de séquence de façon à ce que les résultats obtenus par notre protocole offrent un bon compromis entre charge du réseau et nombre de chemins générés. En fonction des besoins, ces paramètres sont modifiables afin de privilégier la charge du réseau ou le nombre de chemins générés. Si on diminue k, les chemins générés seront plus courts et donc moins nombreux mais la diffusion sera plus confinée et aura donc un impact moindre sur le réseau (on rappelle que k permet de régler la

largeur de la diffusion dirigée). En jouant sur les numéros de séquences, on peut diminuer ou augmenter le nombre de retransmissions par nœud ce qui permet de contrôler la charge du réseau dans la zone de diffusion. Au plus il y aura de numéros de séquences, au plus on générera de chemins mais au plus on surchargera le réseau. Inversement, si on accepte d'avoir moins de chemins générés, on pourra diminuer la charge du réseau en diminuant le nombre de numéros de séquence.

4.4.4 Conclusion

Nous venons de montrer qu'il est possible de générer un ensemble de chemins nœud-disjoints à l'aide d'une seule et unique passe requête/réponse. En plus de pouvoir être utilisée pour évaluer la qualité du réseau entre deux nœuds, la génération d'ensemble de chemins disjoints peut également être utilisée dans le cadre d'algorithme de routage multi-chemins [DMB+03, PHS02, LS02, LG01, LLP+01].

Cependant, pour utiliser ces ensembles dans le cadre de prédiction de partitionnement, il faut qu'ils soient suffisamment riches (c'est à dire que le nombre de chemins trouvés par l'algorithme de recherche de route soit proche du nombre réel de chemins existant entre les deux nœuds). Pour cela, même si notre protocole propose des mécanismes permettant d'optimiser substantiellement le nombre de chemins générés tout en diminuant la charge induite dans le réseau, cette dernière reste tout de même importante. En effet, si il existe de nombreux couples client/serveur dans le réseau (ce qui semble tout à fait probable voire même évident dans cadre d'application grand public), nos mécanismes de suffiront probablement pas à éviter une surcharge complète du réseau.

Le théorème de Menger [Men27, BGH01] permet dans sa forme nœud-disjoints de donner un équivalent à la recherche de chemins nœud-disjoints. Dans cette forme, ce théorème s'exprime ainsi :

Théorème 1 (Menger (forme non orientée, nœud-disjoints)) *Dans un graphe non orienté le nombre maximum de chemins deux à deux nœud-disjoints d'un sommet x à un sommet y non adjacents est égal au nombre minimum de nœuds à supprimer pour déconnecter x de y.*

La recherche de chemins nœud-disjoints entre deux nœuds peut donc se ramener à la recherche de nœuds dont la suppression déconnecte ces derniers. De tels nœuds sont dits **critiques** et peuvent être détectés par des algorithmes centralisés de type recherche en profondeur [Tar72].

Dans [GJC02], Goyal et Caffery utilisent la notion de liens critiques afin de prédire des partitionnements. Cette notion est identique à celle de nœud critique à ceci près qu'on ne considère alors pas les nœuds dont la suppression déconnecte le réseau mais les liens. Ces liens peuvent également être détecté par une recherche en profondeur sur le graphe complet. C'est cette solution qu'ont retenue Goyal et Caffery.

Il nous a donc paru intéressant de proposer deux méthodes d'évaluation supplémentaires, basées sur les liens critiques et les nœuds critiques. Néanmoins, les méthodes actuelle basées sur une connaissance totale du réseau ne sont pas envisageable si l'on veut pouvoir passer à l'échelle ou supporter la mobilité par exemple.

La nature distribuée des réseaux ad hoc nécessite la définition de protocoles localisés si l'on veut pouvoir supporter le passage à l'échelle ainsi que la mobilité ou fournir des solutions robustes. Les algorithmes que nous proposons dans ce mémoire sont des algorithmes localisés qui permettent de découvrir rapidement des liens ou des nœuds critiques. Par contre, du fait de l'utilisation d'une connaissance uniquement partielle du réseau (limitée à quelques saut pour chacun des nœuds), ces algorithmes vont détecter des liens ou des nœuds comme critiques alors qu'ils ne le sont pas si l'on observe le réseau dans sa globalité. Cependant, si l'on veut fournir une qualité de service correcte aux applications en terme de bande de passante ou de temps de réponse, les routes très longues reliant deux nœuds entre eux ne sont pas intéressantes. Les algorithmes localisés vont donc permettre de fournir une information plus utile encore que le partitionnement du réseau. On va en effet pouvoir détecter quand la qualité des routes fournies aux applications se dégradent d'une façon trop importante pour offrir une qualité de service correcte. En outre, la prédiction de partitionnement se fera de façon beaucoup plus rapide et impliquera beaucoup moins de surcharge du réseau.

La suite de ce chapitre propose des méthodes originales destinées à faire la découverte de ce type de liens et de nœuds de manière localisée.

4.5 Évaluation par protocoles localisés

Les évaluations que nous proposons se basent sur les notions de nœuds et de liens critiques qui peuvent être définies de la manière suivante :

Définition 5 *Un nœud, ou un lien, est critique si le fait de le retirer du graphe partage celui-ci en deux composantes connexes distinctes (ou plus).*

Par exemple, Figure 4.11, le lien AB est critique car si on l'enlève, le graphe sera séparé en deux composantes contenant respectivement les nœuds B, C, D, E, F, G et les nœuds A, I, J, K, L, M, N. De même, les nœuds A, B, I et J sont critiques.

Il paraît naturel que si une route emprunte un tel lien ou un tel nœud, elle peut être considérée comme faible, dans le sens ou les risques de la voir casser sont grands. En effet, il suffit à ce lien ou à ce nœud de disparaître pour que la route ne puisse être reconstruite en passant par d'autres liens.

La première chose à définir quand on parle d'algorithme localisé est la notion de connaissance locale. On parle alors de connaissance à k sauts. Deux nœuds

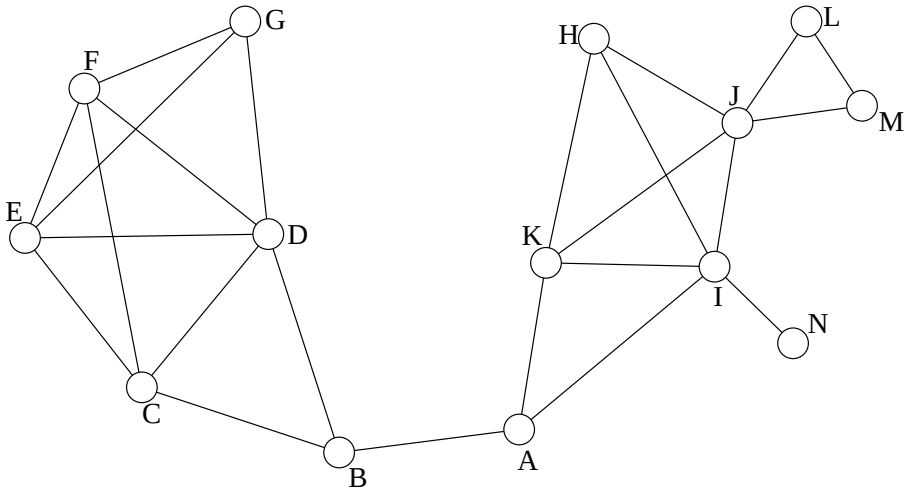


FIG. 4.11 – *Liens et nœuds critiques.*

sont considérés comme étant des voisins à k sauts si et seulement si la route la plus courte les reliant fait au maximum k sauts. Un nœud qui possède uniquement des informations le concernant à une connaissance à 0 saut.

Les nœuds obtiennent cette connaissance en envoyant des messages *hello* à leurs voisins qui contiennent le graphe de leurs voisins à $k-1$ sauts. La connaissance à 1 saut est donc une simple liste des voisins directs. L'information à 2 sauts est obtenue en diffusant la liste des voisins à 1 saut. Cette information est donc constituée des liens entre les voisins à 1 saut ainsi que des liens entre ces derniers et les voisins à 2 sauts. Par contre, aucune information n'est connue quant aux liens existants entre les voisins à 2 sauts. D'une manière plus générale, l'information à k sauts est constitué du sous graphe des voisins à k sauts, en incluant les liens entre les voisins à $k-1$ et à k sauts mais ne précisant pas si les voisins à k sauts sont connectés.

On peut tout de même noter, même si nous nous efforçons de nous en passer dans tous les travaux présentés dans ce mémoire, que si l'on dispose d'un système de positionnement ainsi que de la valeur de la portée de communication de l'interface radio, on peut savoir si les nœuds à k sauts sont connectés entre eux. Ceci permet, au prix du système de positionnement, d'obtenir de meilleurs résultats avec une connaissance à même distance. Nous décrirons cependant nos algorithmes dans le cas où l'on n'utilise pas de système de positionnement car cela sort des limites que nous nous sommes fixés. La technique reste cependant identique.

Ceci étant défini, nous pouvons maintenant décrire nos protocoles de détection localisés de nœuds et de liens critiques.

4.5.1 Algorithme localisé pour la détection de nœuds critiques

Pour approximer la notion de nœud critique de manière locale, nous définissons les nœuds k-critiques :

Définition 6 *Un nœud u est dit k-critique si le sous graphe des ses voisins à k sauts, duquel on exclut u ainsi que les liens qui y mènent est non connexe.*

L'algorithme localisé consistera alors à obtenir des informations sur le voisinage à k sauts des nœuds et d'en déduire quels sont les nœuds k critiques. Il est clair que si un nœud est critique de manière globale, il sera détecté par cet algorithme localisé (*i.e.* il sera k-critique $\forall k \geq 1$). De plus, si k est égal au diamètre du réseau, la connaissance est totale donc l'algorithme est équivalent à une version centralisée. Si on prends $k = 1$, on ne dispose pas d'information sur les liens entre les voisins et donc tous les nœuds sont déclarés critiques ce qui n'est, de manière évidente, pas intéressant. Il faudra donc au minimum se pencher sur la connaissance à deux sauts ou plus en fonction de la précision que l'on souhaite obtenir.

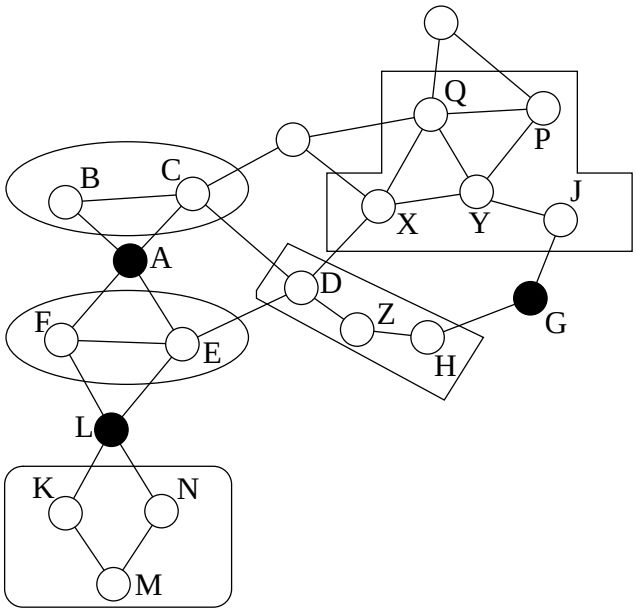


FIG. 4.12 – *Illustration de la définition locale des nœuds critiques.*

La Figure 4.12 illustre ceci ainsi que la définition locale des nœuds critiques. Les voisins à 1 saut de A peuvent être répartis en deux ensembles $\{B,C\}$ et $\{E,F\}$ (encerclés sur la figure). Grâce à une connaissance à un saut, on peut ne peut pas savoir que B est voisin de C, ni que F est voisin de E. L'ensemble des voisins

est donc déconnecté donc A est déclaré 1 critique (cela ne faisant qu'illustrer la remarque précédente concernant la détection faite par une connaissance à un saut). Par contre, si on à une connaissance à deux sauts, on sait que B et C sont voisins et que F et E le sont également. On sait également que C à comme autre voisin D et que E à également D comme autre voisin. L'ensemble des voisins à deux sauts de A est donc connecté, ce qui implique que A n'est pas 2-critique et n'est donc pas non plus critique globalement. Le nœud L est quant à lui 2-critique car l'ensemble de ses voisins à deux sauts n'est pas connecté. En effet, il est séparé en deux ensembles distincts, $\{F,E\}$ et $\{K,M,N\}$. Ce nœud est même critique de façon globale, on peu augmenter k autant que l'on veut, la composante $\{K,M,N\}$ sera toujours isolée. Le nœud G est 3-critique car l'ensemble des ses voisins à 3 sauts est également séparé en deux composantes, $\{J,P,Q,X,Y\}$ et $\{D,Z,H\}$. Par contre, si on étends à $k = 4$, on découvre que X et D sont voisins. G n'est donc pas 4-critique et n'est donc pas non plus globalement critique.

4.5.2 Algorithme localisé pour la détection de liens critiques basé sur les voisins communs à k sauts

Nous pouvons aborder d'une manière similaire la détection de liens critiques. On peut en effet également définir les liens k-critiques de la manière suivante :

Définition 7 *Un lien uv est k-critique si les ensembles des voisins à k sauts de u et de v (construits en supposant que le lien uv n'existe pas) sont disjoints.*

Comme précédemment, si un lien est critique de façon globale, il sera k-critique quelque soit $k \geq 1$.

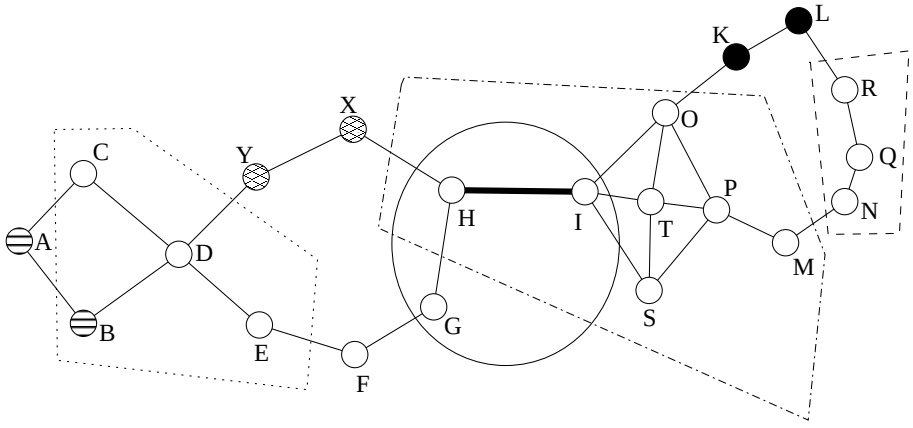


FIG. 4.13 – *Illustration de la définition locale des liens critiques.*

La Figure 4.13 illustre cette définition. Le seul lien globalement critique est le lien HI. L'algorithme localisé va le détecter mais va également en trouver d'autres.

Par exemple, le lien AB est 1-critique car les nœuds A et B n'ont pas de voisin commun à un saut. Par contre, en étendant à une connaissance à deux sauts, les deux ensembles des voisins de A et B ont un voisin à deux sauts commun, le nœud D. Le lien XY est quant à lui 2-critique. En effet, les ensembles des voisins à deux sauts de X (G,H,I) et de Y (B,C,D,E) sont disjoints. Mais, si on prends $k = 3$, on trouve bien que ce lien n'est pas critique car les ensembles des voisins à trois sauts ne sont pas disjoints (le nœud F est commun aux deux ensembles). De même, le lien KL sera déclaré 3-critique mais pas 4-critique.

4.5.3 Évaluation des performances

Afin de montrer l'efficacité de nos méthodes locales, nous avons comparé leur résultats avec ceux obtenus par un algorithme global (autrement dit, la valeur correcte). Nous présentons ici les résultats expérimentaux obtenus sur des graphes connectés. Ces graphes sont générés aléatoirement avec une densité donnée d. La taille de la zone est fixée ainsi que le nombre de nœuds n. Le rayon de communication r des nœuds est calculé à partir de la formule $d = \frac{(n-1)\pi r^2}{h \times l}$ avec l et h la largeur et la hauteur de la zone, respectivement.

Nos expériences ont été menées avec $n = 100$ et $n = 500$ nœuds pour des densités allant de 3 à 15 nœuds. Dans chaque cas, on mesure le rapport de détection comme étant la proportion de nœuds ou de liens déclarés comme critique par les méthodes locales et qui le sont également par une méthode globale. Nous avons aussi observé le nombre moyen de nœuds et de liens critiques dans le réseau.

D'une manière générale, on observe que les résultats sont légèrement meilleurs avec un nombre de nœuds plus important ce qui n'était pas, à priori, ce à quoi on pouvait s'attendre de manière intuitive. Cependant, ceci est encourageant quant aux possibilités de passage à grande échelle de nos protocoles.

4.5.3.1 Détection des nœuds critiques

La Figure 4.14 montre les résultats de l'algorithme de détection des nœuds critiques à l'aide d'une connaissance à 2 et 3 sauts sur un réseau de 500 nœuds. On peut voir que la le rapport de prédiction est toujours supérieur à 50 % ce qui veut dire que plus de la moitié des nœuds détectés comme critiques par l'algorithme local le sont également par un algorithme global (et le sont donc réellement). Comme on pouvait s'y attendre, le résultat est meilleur avec une connaissance à 3 sauts. Néanmoins, la qualité des résultats de l'algorithme à 2 sauts laisse à penser qu'il sera suffisant pour l'utilisation dans le cadre d'un algorithme de prédiction de partitionnement.

La Figure 4.15 confirme les bons résultats de l'algorithme localisé.

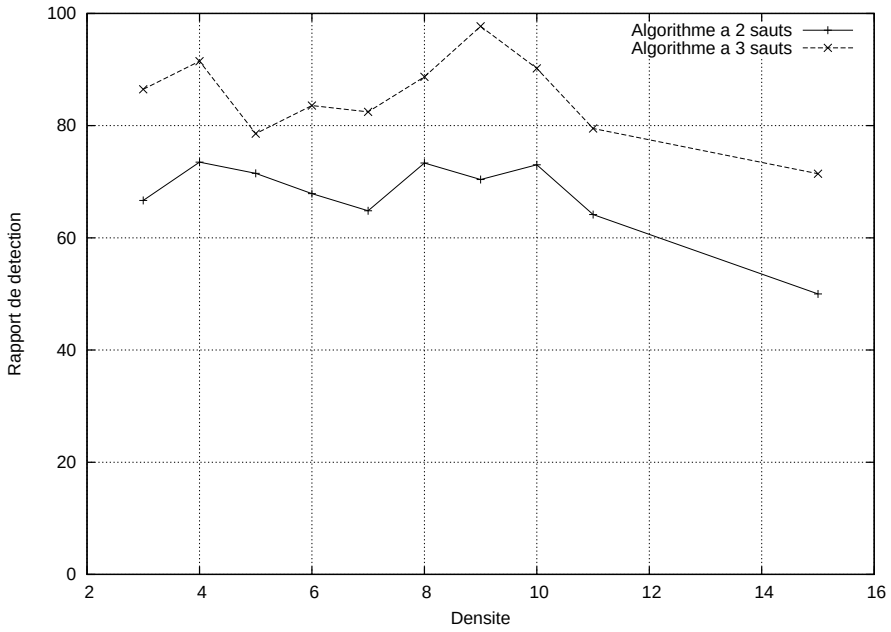


FIG. 4.14 – *Rapport de détection des nœuds critiques.*

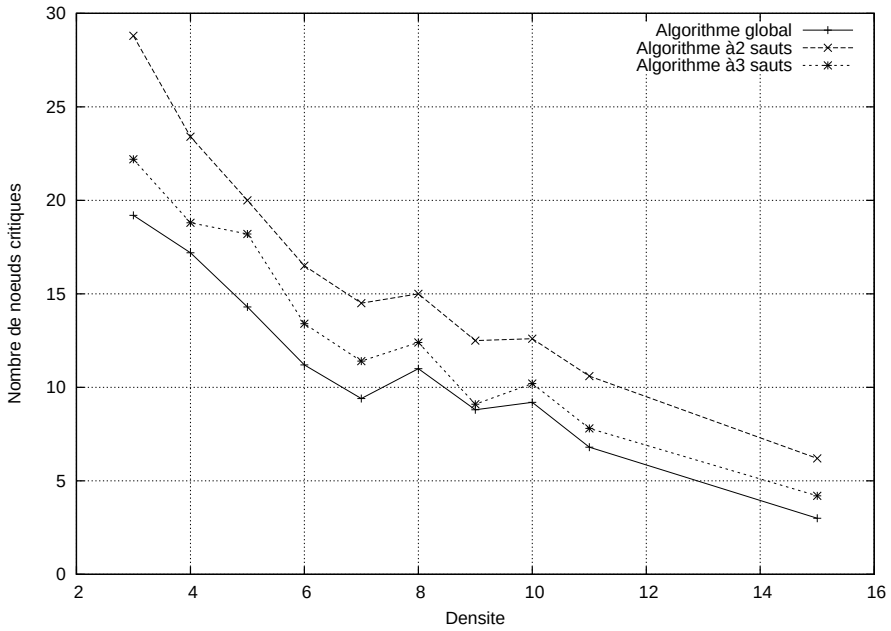


FIG. 4.15 – *Nombre moyen de nœuds critiques dans le réseau.*

4.5.3.2 Détection des liens critiques

Les Figure 4.16 et 4.17 montre des résultats similaires dans le cas de la détection de liens critiques. On peut voir que, comme dans le cas de la détection de nœuds critiques, les algorithmes localisés pour la détection de liens critiques donnent d'excellents résultats puisqu'avec une connaissance à 3 sauts, on arrive à une précision supérieure à 80 % quelque soit la densité du réseau. On observe également que les résultats obtenus par l'algorithme à 2 sauts restent très intéressants ce qui permettra de les utiliser efficacement dans la prédiction de partitionnement.

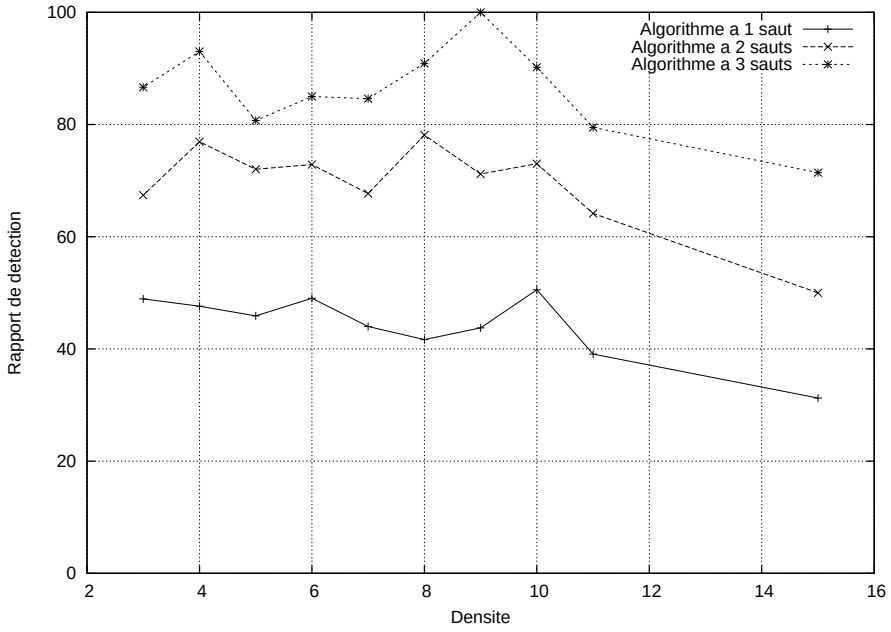


FIG. 4.16 – *Rapport de détection des liens critiques.*

4.6 Performance des algorithmes d'évaluation pour la prédiction de partitionnement

Nous allons maintenant observer à travers nos expérimentations le comportement des différents algorithmes d'évaluation que nous avons proposé précédemment. Pour prédire un partitionnement entre deux nœuds u et v, on utilise l'algorithme 4.2 en définissant les seuils critiques des trois méthodes d'évaluation de la manière suivante :

1. Il n'existe qu'un seul chemin nœud-disjoint entre u et v,
2. Il existe un chemin sans boucle entre u et v qui contient un lien critique,

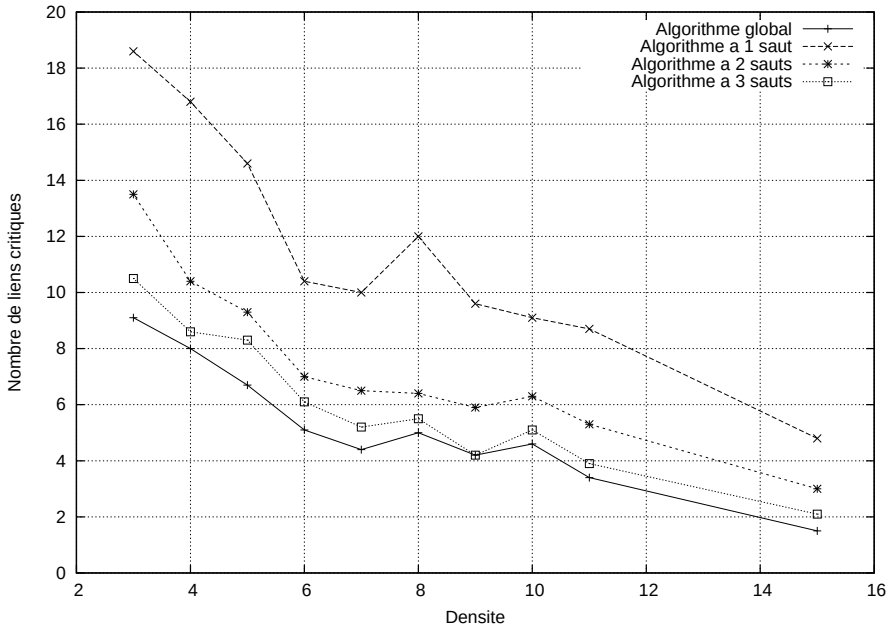


FIG. 4.17 – *Nombre moyen de liens critiques dans le réseau.*

3. Il existe un chemin sans boucle entre u et v qui contient un nœud critique.

Après analyse, il s'avère que la méthode basée sur les nœuds critique doit être modifiée pour être vraiment efficace. En effet, comme le montre la Figure 4.18, la présence d'un nœud critique sur un chemin reliant le client au serveur n'est pas forcement signe de faiblesse car ce nœud peut séparer le graphe en deux composantes tout en conservant le client et le serveur dans la même.

La méthode 3 doit donc être améliorée pour prendre en compte ce genre de situation qui mènerait à une prédiction erronée. La proposition suivante corrige le problème :

– On note $CC_u(v)$ l'ensemble des nœuds qui appartiennent à la même composante connexe que v quand le nœud u est retiré. Il y a risque de déconnexion s'il existe un chemin sans boucle entre u et v qui contient un nœud critique w tel que son prédécesseur x et son successeur y satisfont $CC_w(x) \neq CC_w(y)$.

La propriété $CC_w(x) \neq CC_w(y)$ est facilement évaluable localement car elle revient à chercher s'il existe un chemin entre x et y qui n'utilise pas le nœud w. S'il n'y en a pas, x et y n'appartiennent pas à la même composante connexe (privée de w).

D'un point de vue pratique, et afin de diminuer les alertes non pertinentes, nous introduisons dans l'algorithme de prédiction un *timeout*. L'alerte ne sera alors donnée que si l'évaluation atteint son seuil critique pendant un temps égal

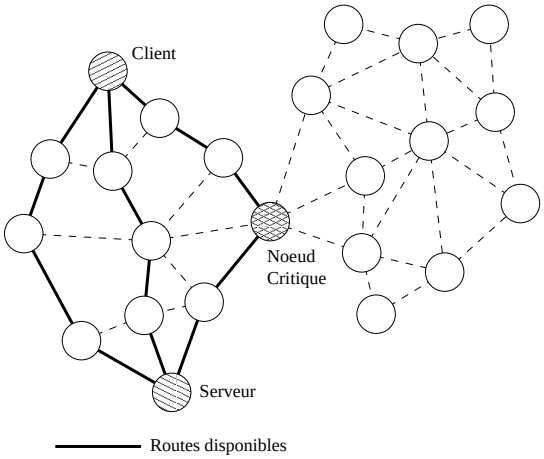


FIG. 4.18 – *Un nœud critique n'est pas forcement signe de faiblesse.*

au *timeout*. En effet, il est fréquent que l'évaluation atteigne son seuil d'alerte de façon ponctuelle pour remonter à une valeur reflétant une fiabilité suffisante. La valeur de ce *timeout* sera également évaluée.

Afin de comparer les trois méthodes, nous allons observer deux paramètres, l'efficacité, c'est à dire le nombre de déconnexions prédites avec succès, et le temps existant entre la prédiction et la déconnexion effective. Ce temps sera mis en rapport avec le temps total pendant lequel le client et le serveur auront été connectés.

Les paramètres de simulation sont une densité de 8 nœuds par zone de communication et une vitesse de 2 mètres par seconde. La densité est choisie car elle correspond à un réseau moyennement dense dans lequel les partitionnement du réseau peuvent arriver régulièrement. Choisir une densité plus important entraîne très peu de déconnexions. La vitesse corresponds quant à elle à la vitesse d'une marche rapide, ce qui se place dans le cadre de notre étude, à savoir des piétons à l'intérieur d'un bâtiment. Le nombre de nœuds est toujours fixé à 500.

Les Figures 4.19 et 4.20 nous permettent de choisir une valeur de *timeout* efficace pour les différentes méthodes. La première observation que l'on peut faire est que les trois méthodes d'évaluation offrent des résultats relativement similaires. Ceci peut s'expliquer par la forte corrélation existant entre les différentes méthodes d'évaluation, en particulier dans le cas des chemins nœud-disjoints et des nœuds critiques qui sont montrées équivalentes par le théorème de Menger. Les différences entre ces deux dernières méthodes s'expliquent simplement par le fait que les algorithmes utilisés pour récupérer l'information nécessaire ne sont pas globaux et ne fournissent donc qu'une approximation de la réalité.

Concernant le *timeout*, comme on pouvais s'y attendre, la valeur du temps de déclenchement de l'alerte influe pour beaucoup dans l'efficacité de la prédiction

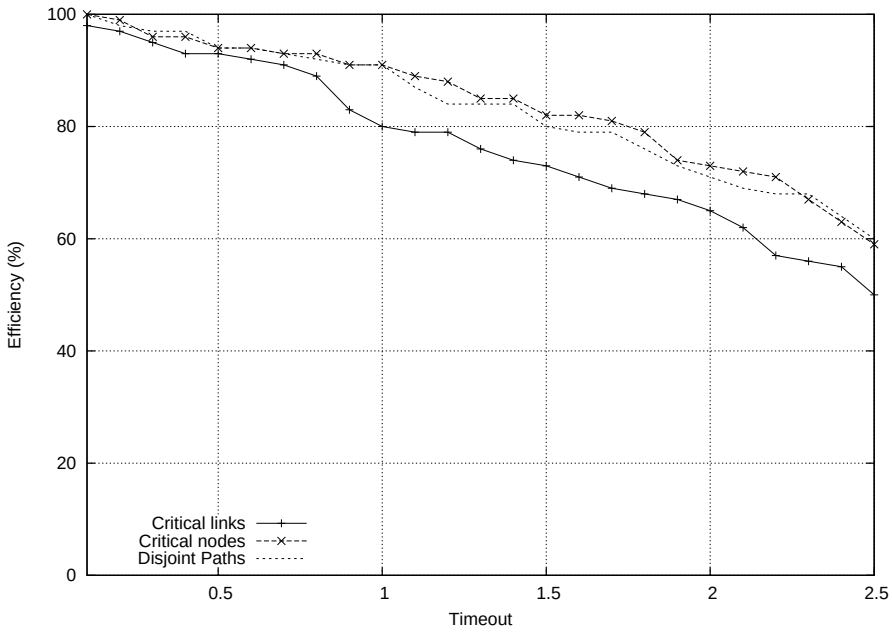


FIG. 4.19 – *Éfficacité des méthodes de prédiction en fonction de la valeur du timeout*

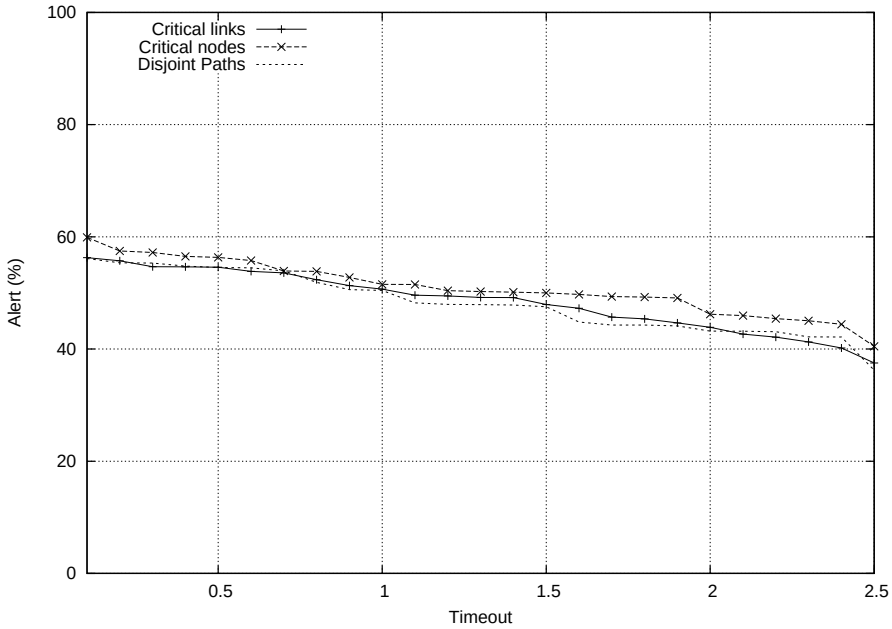


FIG. 4.20 – *Temps passé en alerte en fonction de la valeur du timeout*

ainsi que dans la pertinence des alertes. Pour qu'un algorithme de prédiction soit performant, il faut bien évidement qu'il soit fiable, c'est à dire que l'efficacité soit proche de 100 %, mais il faut aussi qu'il ne prédise pas de déconnexion quand cela n'est pas nécessaire. Il est en effet aisé de définir un protocole naïf qui aurait une efficacité totale. Il suffit de signaler une alerte dès le début de la connexion entre le client et le serveur. Il va de soi qu'une telle méthode n'est pas du tout utilisable. La prédiction doit donc être donnée peu de temps avant la déconnexion.

On observe que le plus petit temps passé en alerte[2] tout en conservant une fiabilité correcte (au delà de 80 %) est situé aux alentours de 50 % du total de connexion total en utilisant un timeout compris entre une seconde et une seconde et demi. Cette valeur peut paraître importante au premier abord. Cependant, il ne faut pas oublier qu'il faut laisser du temps aux applications pour réagir à une éventuelle déconnexion. En fonction de la solution retenue pour la réaction, le temps nécessaire peut s'avérer être important, par exemple pour la réplication d'un service de taille importante. Dans cet optique, le temps d'alerte peut être jugé correct.

Néanmoins, il serait bien évidement intéressant de pouvoir quantifier le temps restant aux applications pour réagir. Grâce à la Figure 4.21, nous donnons un premier éléments de réponse sur ce point. Nous avons observé la distribution des temps observés entre la prédiction et la déconnexion sur l'ensemble de nos expériences.

Nous pouvons donc, à l'heure actuelle, construire par simulations des abaques permettant de fournir une estimation sur le temps qu'aurait une application pour réagir après l'alerte. Dans le cas de la Figure 4.21, nous pouvons énoncer qu'en moyenne, à densité 8 pour une vitesse de 2 mètres par seconde, il y a une chance sur deux pour que la déconnexion interviennent au maximum 50 secondes après l'alerte.

4.7 Conclusion

Dans ce chapitre, nous avons proposé des solutions originales pour répondre au problème de la détection de partitionnement dans un réseau ad hoc.

Contrairement aux solutions existantes, qui utilisent des algorithmes centralisés et/ou un système de positionnement, nos solutions n'utilisent que la notion de voisinage local et peuvent donc être utilisables quel que soit l'équipement dont sont pourvus les nœuds. Le coût engendré par ces protocoles locaux ne peuvent donc qu'être bien inférieurs à des versions centralisées. L'utilisation de nos protocoles pour des grands réseaux sera donc possible. Bien que notre première solution, basée sur la notion de chemins nœud-disjoints, induisent une charge réseau plus

2. c'est à dire entre la prédiction et la déconnexion

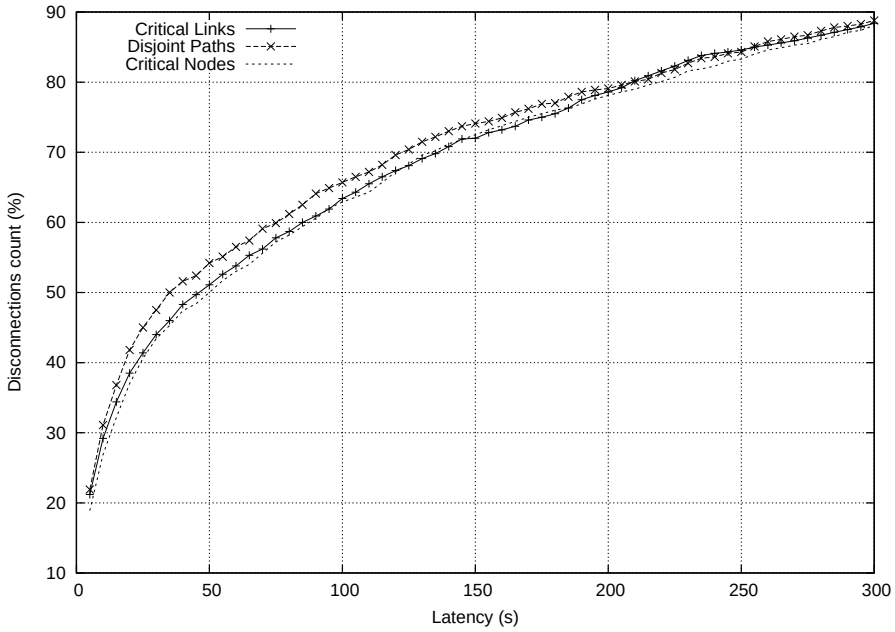


FIG. 4.21 – *Distribution du temps d'alerte à densité 8, 2m/s.*

importante que l'évaluation par nœuds ou liens critiques, le protocole de diffusion que nous proposons peut être utilisé dans le cadre du routage multi-routes.

Nous proposons de retenir l'algorithme basé sur la détection des nœuds critiques. Ces résultats sont les plus intéressants et on peut, par simulation, dresser des abaques qui pourrait permettre aux nœuds d'obtenir une information précise sur le temps qu'il reste avant la déconnexion.

Dans ce travail, nous n'avons pas examiné la façon dont les nœuds pourraient déduire leur vitesse (par exemple, en mesurant le *turn-over* de la table de routage), ainsi que la densité du réseau, ce qui serait nécessaire pour pouvoir prédire avec précision quand aura lieu la déconnexion. Nous pensons que des informations concernant la densité locale ainsi que la vitesse d'évolution des ensembles de voisins peut conduire à un protocole fiable et précis.

Chapitre 5

Conclusion

« Le danger qui menace les chercheurs aujourd'hui serait de conclure qu'il n'y a plus rien à découvrir »

La recherche passionnément, Pierre Joliot.

Dans ce document, nous avons présenté des solutions originales à quelques problèmes primordiaux liés à la gestion de service en milieu ad hoc. Pour chacune de nos solutions, nous avons cherché à nous limiter quant aux matériel à embarquer sur les objets constituant le réseau. Nous avons donc développer des protocoles qui ne nécessite que des informations sur le voisinage local, qui sont faciles à obtenir. En particulier, aucune de nos solutions n'utilise de système de positionnement ni ne se base sur une quelconque infrastructure. Nous n'utilisons pas non plus d'algorithme nécessitant une connaissance globale du réseau et le caractère totalement distribué rends nos méthodes tolérantes aux pannes et capables de fonctionner pour des réseaux de taille et de densité importante.

Tout d'abord, nous avons proposé une méthode efficace pour répartir une information de manière efficace dans le réseau [HPSR04]. Cette partie de notre travail réponds à la question « Comment publier un service dans le réseau? ».

Nous avons tout d'abord proposé un critère permettant de juger de la qualité de la répartition de l'information dans le réseau. Ce critère prend en compte le coût de la publication ainsi que le coût de la recherche.

Nous avons montré ensuite que l'utilisation d'une loi de probabilité ainsi que d'un mécanisme de réduction de graphe, les ensembles dominants, permettait d'effectuer une sélection efficace des nœuds devant mémoriser l'information. Cette sélection est efficace par l'utilisation faible en mémoire (peu de nœuds sont nécessaires à la bonne répartition de l'information) et la vérification du critère de bonne dissémination.

Ensuite, nous avons proposé une série d'algorithmes permettant de prédire le partitionnement du réseau. Une telle prédiction peut venir améliorer la qualité d'un service au niveau applicatif mais aussi permettre de fiabiliser un algorithme de routage en lui donnant la possibilité de prédire que les routes qu'il propose risquent de disparaître. Nous avons proposé trois solutions à ce problème.

Une première, basée sur l'utilisation d'ensemble de chemins nœud-disjoints [HSRC03a]. Pour cette dernière, nous avons proposé un algorithme de recherche de routes [HSRC03b] permettant de trouver, de manière efficace, un ensemble de routes possédant la caractéristique intéressante d'être disjoint.

Le coût relativement important de l'algorithme de recherche de routes nous a amené à proposé deux autres méthodes, basée sur la détection de nœuds et de liens critiques. À cet effet, nous avons proposé des algorithmes de détections localisés [JSHSR04], là où les algorithmes existant étaient tous centralisés.

Pour ces trois méthodes, nous sommes capables de fournir une probabilité de déconnexion après un temps donné, pour une vitesse et une densité donnée. Une application peut alors réagir en fonction de cette probabilité pour améliorer la qualité de son service.

Perspectives

Nos travaux offrent plusieurs voies d'exploration pour des travaux futurs. Premièrement, en ce qui concerne leur amélioration. En effet, bien que nos protocoles soient efficaces, nous pensons pouvoir encore améliorer les résultats.

Dans le cas de la dissémination, il paraît important de tenir compte également des capacités de chaque nœud. Les critères pouvant être pris en compte sont la taille de cache disponible, l'énergie disponible ainsi que la stabilité du nœud. Par stabilité, on entend stabilité du voisinage. En effet, un nœud dont le voisinage change peu fréquemment peut être un bon candidat à la mise en cache de l'information. Cette adaptation peut être faite soit par règle fixe soit par adaptation locale du facteur de rapprochement.

En ce qui concerne la prédiction de partitionnement, nous pouvons actuellement donner de bons résultats si l'on connaît à l'avance la densité et la mobilité du réseau. Le travail principal à réaliser serait alors l'approximation de ces données en fonction du voisinage. Les méthodes pouvant être proposées aux applications pour réagir à la prédiction de partitionnement sont encore à étudier.

Deuxièmement, nos protocoles offrent des possibilités intéressantes pour d'autres types d'applications.

Notre approche généraliste de la dissémination d'informations permet de l'utiliser non seulement dans la recherche de services mais également, dans toute application nécessitant l'accès rapide à des données (partage de fichiers, recherche de position...).

Enfin, notre protocole de recherche de route peut être également utile pour développer des algorithmes de routages de type multi-chemins [DMB+03, LLP+01]. L'intérêt du routage multi-chemins sera d'accentuer la fiabilité des routes par leur diversité. Un tel routage, accompagné de notre protocole de prédiction de partitionnement, doit permettre de développer de nombreuses applications fiables en s'affranchissant des problèmes topologiques liés à la mobilité des nœuds.

Bibliographie

[Abr70] N. Abramson. The ALOHA system - Another alternative for computer communications. In *Proceedings of AFPIS Conference*, volume 37, 1970.

[AML+93] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey, and B. Alberti. *RFC 1436 - The Internet Gopher Protocol (a distributed document search and retrieval protocol)*. University of Minnesota, March 1993.

[BCGSar] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors. *Ad Hoc Networking*. IEEE Press, to appear.

[BCS99] S. Basagni, I. Chlamtac, and V. R. Syrotiuk. Dynamic source routing for ad hoc networks using the global posit ioning system. In *Proceedings of the IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, New Orleans, Louisiana, September 1999.

[BCSW98] S. Basagni, I. Chlamtac, V.-R. Syrotiuk, and B.-A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, Dallas, Texas, United States, 1998.

[BDSZ94] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A media access protocol for wireless LANs. In *Proceedings of ACM SIGCOMM Symposium on Commmunication, Architectures and Protocol*, pages 212–225, London, U.K., October 1994.

[Ber96] P.A. Bernstein. Middleware: A model for distributed system services. *Communications of the ACM*, 39(2):86–98, February 1996.

[BGH01] T. Böhme, F. Göring, and J. Harant. Menger's theorem. *Journal of Graph Theory*, 37:35–36, 2001.

[BL90] T. Berners-Lee. Information management: A proposal, May 1990.

[BLMM94] T. Berners-Lee, L. Masinter, and M. McCahill. *RFC 1738 - Uniform Resource Locators (URL)*. CERN and Xerox and University of Minnesota, December 1994.

[blu99] Specification of the bluetooth system, December 1999. Available from
URL http://www.bluetooth.com.

[BMJ+98] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[bP85] IEEE Standards boards Part 3. Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD), 1985.

[bP98a] IEEE Standards boards Part 5. Local and metropolitan area networks: Token ring access method and physical layer, 1998.

[BP98b] S. Brin and L. Page. The anatomy of a large scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[bP99] IEEE Standards boards Part 11. Wireless LAN medium access control and physical layer specifications, 1999.

[bP02] IEEE Standards boards Part 3. Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation, 2002.

[BR00] C. Bettstetter and C. Renner. A comparison of service discovery protocols and implementation of the service location protocol. In *Proceedings of the 6th EUNICE Open European Summer School: Innovative Internet Applications (EUNICE'00)*, Twente, Netherlands, September 2000.

[BR04] P. Basu and J. Redi. Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, 18(4):36–44, July/August 2004.

[CCB02] D. Conan, S. Chabridon, and G. Bernard. Disconnected operations in mobile environments. In *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, International Parallel and Distributed Symposium (IPDPS 2002)*, Fort Lauderdale, Florida, USA, April 2002.

[CCJ90] B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[CIS03] J. Cartigny, F. Ingelrest, and D. Simplot. RNG relay subset flooding protocols in mobile ad-hoc networks. *International Journal of Fondations of Computer Science*, 14(2):253–265, 2003.

[CJ03] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). Request For Comments (RFC) 3626, Internet Engineering Task Force, October 2003.

[CJL+01] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE INMIC*, Pakistan, 2001.

[CR99] K.A.L. Coar and D.R.T. Robinson. *The WWW Common Gateway Interface Version 1.1*. IBM Corporation and E*TRADE UK Ltd., June 1999.

[Cri03] M. Crispin. *RFC 3501 - Internet Message Access Protocol - Version 4rev1*. University of Washington, March 2003.

[CSR04] J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. *IEEE Computer Magazine*, (37):40–46, 2004.

[CSS03] J. Cartigny, D. Simplot, and I. Stojmenović. Localized minimum-energy broadcasting in ad-hoc networks. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, USA, 2003.

[DH98] S. Deering and R. Hinden. *RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification*. Cisco and Nokia, December 1998.

[DMB+03] S.K. Das, A. Mukherjee, S. Bandyopadhyay, D. Saha, and K. Paul. An adaptive framework for QoS routing through multiple paths in ad-hoc wireless networks. *Journal of Parallel and Distributed Computing*, 63:141–153, 2003.

[DRWT97] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal stability based adaptive routing for ad hoc networks. *IEEE Personal Communication*, February 1997.

[DW03] F. Dai and J. Wu. Distributed dominant pruning in ad hoc networks. In *Proc. of IEEE International Conf. on Communications (ICC'03)*, May 2003.

[EGHK99] D. Estrin, R. Govindan, J.S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pages 263–270, 1999.

[FB96] N. Freed and N. Borenstein. *RFC 2045 - Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Innosoft and First Virtual, November 1996.

[Fee99] L.M. Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T99/07, SICS (Swedish Institute of Computer Science), Sweden, October 1999.

[FF93] L.R. Ford and D.R. Fulkerson. Improving the routing and addressing of IP. *IEEE Journal Network Magazine*, 7:10–15, 1993.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1*. UC Irvine and Compaq and W3C and MIT and Xerox and Microsoft, June 1999.

[GCJD99] E. Guttman, C.Perkins, J.Veizades, and M. Day. Service location protocol, June 1999. IETF RFC 2608.

[GGL03] S. Ghemawat, H. Gobioff, and S.T. Leung. The google file system. In *Proceedings of Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, New York, USA, October 2003.

[GGM99] J.-M. Geib, C. Gransart, and P. Merle. *CORBA : des concepts à la pratique*. Dunod, Octobre 1999.

[GJC02] D. Goyal and Jr. J. Caffery. Partition avoidance in mobile ad hoc networks using network survivability concepts. In *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, 2002.

[gps97] *The Global Positioning System FAQ*, July 1997.
URL: http://www.gpsy.com/gpsinfo/gps-faq.txt.

[GS69] K.R. Gabriel and R.R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[HGPC99] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad-hoc wireless networks. In *Proceedings of the 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems*, 1999.

[HP98] Z.J. Haas and M.R. Pearlman. The performance of query control schemes for the zone routing pro tocol. In *ACM SIGCOMM'98*, 1998.

[HPS01] Z.J. Haas, M. Perlman, and P. Samar. The Interzone Routing Protocol (IERP) for Ad Hoc Networks. draft-ietf-manet-zone-ierp-01.txt, IETF MANET Working Group, June 2001.

[HPSR04] M. Hauspie, A. Panier, and D. Simplot-Ryl. Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks. In *Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, Fort Lauderdale, Florida, USA, 2004.

[HSRC03a] M. Hauspie, D. Simplot-Ryl, and J. Carle. Partition detection in ad-hoc networks. In *Proceedings of the 2nd IFIP Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET 2003)*, Madhia, Tunisia, June 2003.

[HSRC03b] M. Hauspie, D. Simplot-Ryl, and J. Carle. Partition detection in ad-hoc networks using multiple disjoint paths set. In *Proceedings of*

the 1st International Workshop on Objects models and Multimedia technologies, Geneva, Switzerland, September 2003.

[ISRS04] F. Ingelrest, D. Simplot-Ryl, and I. Stojmenović. *Resource Management in Wireless Networking*, chapter Energy Efficient Broadcasting in Wireless Mobile Networks. Kluwer, 2004. à paraître.

[Jai00] A. Jain. Routing protocols for mobile ad hoc networks. Technical report, Department of computer science and engineering, Indian Institute of Technology, Kanpur, 2000.

[JM96] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[Joh94] D. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., 1994.

[JSHSR04] M. Jorgic, I. Stojmenovic, M. Hauspie, and D. Simplot-Ryl. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *Proceedings of the 3rd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2004)*, Bodrum, Turkey, 2004.

[Kah77] R.E. Kahn. The organization of computer resources into a packet radio network. In *IEEE Transactions on Communications*, volume COM-25, pages 169–178, January 1977.

[Kar90] P. Karn. MACA - a new channel access method for packet radio. In *Proceedings of the 9th Computer Networking Conference (ARRL/CRRL Amateur Radio)*, September 1990.

[KF02] H. Koubaa and E. Fleury. Service location protocol overhead in the random graph model for ad hoc networks. In *Proceedings of the 7th International Symposium on Computers and Communications (ISCC 2002)*, pages 49–54, July 2002.

[KK04] M. Karlsson and C. Karamanolis. Choosing replica placement heuristics for wide-area systems. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, 2004.

[KKR] M. Klein and B. Koenig-Ries. Multi-layer clusters in ad-hoc networks - an approach to service discovery. In *International Workshop on Peer-to-Peer Computing*.

[KL86] B. Kantor and P. Lapsley. *RFC 977 - Network News Transfer Protocol*. U.C. San Diego and U.C. Berkeley, February 1986.

[Kru56] J.B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society*, volume 7, pages 48–50, 1956.

[KT75] L. Kleinrock and F. Tobagi. Random access techniques for data transmission over packet-switched radio channels. In *Proceedings of NCC Conference*, pages 187–201, 1975.

[KT04] U.C. Kozat and L. Tassiulas. Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Adhoc Networks*, 2:23–44, 2004.

[LB04] H. Luo and M. Barbeau. Performance evaluation of service discovery strategies in ad hoc networks. In *2nd Annual Conference on Communication Networks and Services Research (CNSR 2004)*, Canada, may 2004.

[LG97] C.R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1265–1275, 1997.

[LG01] S. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of the IEEE Internation Conference on Communications*, pages 3201–3205, 2001.

[LG04] Y. Liangzhong and C. Guohong. Supporting cooperative caching in ad hoc networks. In *IEEE INFOCOM 2004*, 2004.

[Li01] J. Li. A scalable location service for geographic ad hoc routing. Master's thesis, Massachusetts Institute of Technology, January 2001.

[LLP+01] R. Leung, J. Liu, E. Poon, C. Chan, and B. Li. MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks. In *Proceedings of the 26th IEEE Annual Conference on Local Computer Networks (LCN 2001)*, Tampa, Florida, November 2001.

[LNBK02] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *ACM Conf. on Principles of Distributed Computing (PODC)*, Monterey, CA, July 2002.

[LQV01] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. *35th Annual Hawaii International Conference on System Sciences (HICSS'2001)*, 2001.

[LS02] X. Lin and I. Stojmenović. Location-based localized alternate, disjoint and multi-path routing algorithms for wireless networks. *Journal of Parallel and Distributed Computing*, 2002.

[Man] MANET (Mobile Ad-hoc NETwork) group of IETF (Internet Enginering Task Force).
URL: http://tonnant.itd.nrl.navy.mil/manet/.

[Man96] S. Mann. Smart clothing: The shift to wearable computing. In *Communication of the ACM*, pages 23–24, August 1996.

[Meh84] N. Mehravari. TDMA in a random-access environment: An overview. *IEEE Communications Magazine*, 22:54–59, November 1984.

[Men27] K. Menger. Zur allgemeinen kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[Mer97] Ph. Merle. *CorbaScript - CorbaWeb : Propositions pour l'accès à des objets et services distribués*. PhD thesis, Université de Lille 1, 1997.

[MGLA95] S. Murthy and J.J. Garcia-Luna-Aceves. A routing protocol for packet radio networks. In *Proceedings of ACM First International Conference on Mobile Computing & Networking (MOBICOM'95)*, November 1995.

[Mic89] Sun Microsystems. *RFC 1094 - NFS: Network File System Protocol specification*, March 1989.

[Mic95] Microsoft. *The Component Object Model Specification*, October 1995.

[Mic98] Sun Microsystems. *Java RMI Specification*, 1998.

[mic99a] Microsoft corporation, universal plug and play: Background, 1999. Available from
URL http://www.upnp.org/ressources/UPnPbkgnd.htm.

[Mic99b] Sun Microsystems. *Jini architecture specification*, November 1999.

[Moc87] P. Mockapetris. *RFC 1034 - Domain names, Concepts and Facilities*. Information Sciences Institute, University of South California, November 1987.

[MR96] J. Myers and M. Rose. *RFC 1939 - Post Office Protocol - Version 3*. Carnegie Mellon and Dover Beach Consulting, Inc., May 1996.

[NSC03] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *Proc. of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 25–34, 2003.

[Obj04] Object Management Group. *Common Object Request Broker Architecture (CORBA/IIOP) Specification*, 2004.

[oSC81] Information Sciences Institute University of Southern California. *RFC 791 - Internet Protocol*. Defense Advanced Research Projects Agency, 1981.

[PB94] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance vector (DSDV) for mobile computers. *ACM SIGCOMM '94 Computer Communications Review*, 24(4):234–244, October 1994.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Request For Comments (RFC) 3561, Internet Engineering Task Force, July 2003.

[PC97] V.D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM '97*, Japan, April 1997.

[Per01] C.E Perkins, editor. *Ad Hoc Networking*. Addison Wesley, 2001.

[PHS02] P. Papadimitratos, Z.J. Haas, and E.G. Sirer. Path set selection in mobile ad hoc networks. In *MobiHoc 2002*, June 2002.

[Plu82] D.C. Plummer. *An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. dcp@mit-mc, November 1982.

[Pos81] J. Postel. *RFC 793 - Transmission Control Protocol*. Defense Advanced Research Projects Agency, 1981.

[Pos82] J.B. Postel. *RFC 821 - Simple Mail Transfer Protocol*. Information Sciences Institute, University of South California, August 1982.

[PR85] J. Postel and J. Reynolds. *RFC 959 - File Transfer Protocol*. Information Sciences Institute, University of South California, October 1985.

[PR99] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second Annual IEEE Workshop on Mobile Computing Systems and Apllications*, pages 90–100, February 1999.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[Rob72] L. Roberts. Aloha packet system with and without slots and capture. Technical Report ASS Note 8, Stanford Research Institute, Advanced Research Projects Agency, Network Information Center, 1972.

[Ros93] M.T. Rose. *The Internet Message: Closing the book with Electronic Mail*. Prentice-Hall, 1993.

[RS96] S. Ramanathan and M. Steenstrup. A survey of routing techniques for mobile communications networks. *ACM/Baltzer Mobile Networks and Applications*, 1(2):89–104, 1996.

[RT99] E. M. Royer and C-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[sal99] Salutation consortium, salutation architecture specification, 1999. Available from
URL http://www.salutation.org/specordr.htm.

[SCN01] S.H. Shah, K. Chen, and K. Nahrstedt. Cross-layer design for data accessibility in mobile ad hoc networks. In *Proceedings of 5th World multiconference on systemics, cybernetics and informatics (SCI 2001), Orlando, Florida*, July 2001.

[Sco96] J. Scourias. Overview of GSM: The Global System for Mobile Communications, 1996.
URL: citeseer.nj.nec.com/scourias96overview.html.

[SI03] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *MDM 2003, Mobile Data Management*, 2003.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, San Diego, CA, USA, August 2001.

[Sto02] I. Stojmenović, editor. *Handbook of Wireless Networks and Mobile Computing*. John Wiley & Sons, 2002.

[Tan03] A. S. Tanenbaum. *Computer Networks 4th Edition*. Prentice Hall, 2003.

[Tar72] R. Tarjan. Depth First Search and linear graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.

[TNC02] Y.-C. Tseng, S.-Y. Ni, and Y.-S. Chen. The broadcast storm problem in a mobile ad hoc network. *ACM Wireless Networks*, 8(2):152–167, March 2002.

[Toh97] C.-K. Toh. Associativity based routing for ad hoc mobile networks. *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems*, 4(2):103–139, March 1997.

[Toh02] C.-K. Toh. *Ad Hoc Mobile Wireless Networks, Protocols and Systems*. Prentice Hall, 2002.

[Tou80] G. Toussaint. The relative neighborhood graph of finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[VdAFdR03] A.-C. Viana, M.-D. de Amorim, S. Fdida, and J.-F. de Rezende. Indirect routing using distributed location information. In *Proc. of IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, Dallas-Fort Worth, Texas, March 2003.

[VdAFdRar] A. Carneiro Viana, M. Dias de Amorim, S. Fdida, and J. Ferreira de Rezende. Self-organization in spontaneous networks: the approach of DHT-based routing protocols. *Adhoc Networks*, to appear.

[WD03] J. Wu and F. Dai. Broadcasting in ad hoc networks based on self-pruning. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, San Francisco, 2003.

[Wei91] M. Weiser. The computer for the twenty-first century. In *Scientific American*, pages 99–104, September 1991.

[WL99a] J. Wu and H. Li. A dominating-set-based routing scheme in ad hoc wireless networks. In *Proceedings of the Third Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm (DIALM)*, pages 7–14, August 1999.

[WL99b] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proc. of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DiaLM)*, pages 7–14, 1999.

[WL02] K. H. Wang and B. Li. Group mobility and partition prediction in wireless ad-hoc networks. In *Proceedings of IEEE International Conference on Communications (ICC 2002)*, volume 2, pages 1017–1021, New York, April 2002.

[ZS03] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. In *Proceedings of IEEE INFOCOM*, San Francisco, USA, 2003.