Une architecture de contrôle de mobilité pour le routage de messages dans un réseau ad hoc de grande taille

Pirro Bracka

THÈSE DE DOCTORAT

en

INFORMATIQUE

Sujet:

Une architecture de contrôle de mobilité pour le routage de messages dans un réseau ad hoc de grande taille

Pirro BRACKA

Numéro officiel
Université de Marne-la-Vallée

THÈSE

PRÉSENTÉE À l'UNIVERSITÉ DE MARNE-LA-VALLÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE MARNE-LA-VALLÉE

Dans la discipline

INFORMATIQUE

par

Pirro BRACKA

Sujet de la thèse

Une architecture de contrôle de mobilité pour le routage de messages dans un réseau ad hoc de grande taille

Presentée et soutenue publiquement à Champs sur Marne le 23 Septembre 2005 devant le jury composé de :

M.	**Jacques**	**DÉSARMÉNIEN**	Directeur
M.	**Llukan**	**PUKA**	Directeur
Mme.	**Pascale**	**MINET**	Rapporteur
M.	**Betim**	**CICO**	Rapporteur
M.	**Gilles**	**ROUSSEL**	Examinateur
M.	**Fatmir**	**HOXHA**	Examinateur
M.	**Serge**	**MIDONNET**	Examinateur

à mes parents

Remerciements

Je voudrais d'abord exprimer ma profonde gratitude aux professeurs Jacques Désarménien, Llukan Puka et Gilles Roussel, pour avoir accepté de diriger et d'encadrer ce travail, pour leur confiance, leur disponibilité et leur soutien tout au long de ma thèse. Mon parcours auprès d'eux m'a permis d'acquérir de nouvelles connaissances et un savoir-faire. Je tiens à dire un grand merci à Gilles Roussel. Il a su orienter mes travaux avec son sens intuitif de la recherche. Cette thèse lui doit beaucoup, tant du point de vue scientifique que du point de vue humain. Je voudrais lui transmettre l'expression de ma reconnaissance et ma plus profonde gratitude.

Mes très vifs remerciements pour Etienne Duris pour ses innombrables conseils tout au long de ma thèse. Son point de vue critique mais enthousiaste, son aide très précieuse et ses remarques judicieuses m'ont permis d'améliorer la rédaction de ce document.

Que Serge Midonnet trouve ici mes plus chaleureux remerciements. Ses conseils éclairés ont largement contribué à cette thèse. Sa confiance et son aide m'ont été réconfortants à de nombreuses occasions.

Je remercie tout particulièrement les membres de mon jury de thèse, qui ont accepté de juger ce travail et de participer au jury.

Je suis très reconnaissant envers Pascale Minet pour m'avoir accordé du temps pour rapporter ma thèse. Ses remarques toujours pertinentes et ses conseils précieux et éclairés m'ont permis d'améliorer la clarté et la structure de ce document.

Je suis très sensible à la présence dans ce jury de Betim Cico et je le remercie d'avoir accepté d'être rapporteur. Ses remarques constructives ont largement contribué à améliorer cette thèse.

Je remercie vivement Fatmir Hoxha pour avoir accepté de faire partie du jury et pour l'intérêt qu'il a bien voulu porter à ce travail.

Tous mes remerciements aux personnes qui ont accepté de relire cette thèse, Teresa Gomez-Diaz et Gautier Loyauté. Leurs conseils et les suggestions dont ils m'ont fait part, m'ont permis d'améliorer la rédaction de ce document.

Un grand merci à tous les membres du laboratoire de l'informatique de l'Institut Gaspard-Monge qui m'ont offert un excellent cadre de travail ainsi qu'un séjour extrêmement agréable. J'aimerais remercier aussi Nelly et Michel Roussignol pour leur chaleureux accueil et leurs précieux conseils tout au long de mon séjour en France.

Enfin, j'adresse mes chaleureux remerciements à mes parents et mon frère pour m'avoir soutenu et encouragé depuis si longtemps et depuis si loin, en particulier, tout au long de cette thèse.

Table des matières

Chapitre 1

Introduction

1.1 Présentation du sujet et problématique

Depuis quelques années, nous assistons à une véritable révolution du monde des télécommunications. En effet, de nouvelles normes de communications ont fait leur apparition et, l'augmentation des débits aidant, on peut dorénavant envisager des applications que l'on aurait jugées futuristes il n'y a pas si longtemps. En particulier, le domaine des réseaux sans fil a connu un développement vertigineux. Son succès est dû essentiellement au développement rapide des appareils sans fil ainsi qu'à la chute des coûts et à la multiplication des services offerts dans les lieux publics, comme les aéroports, les universités etc.

Bientôt les nombreux dispositifs embarqués qui forment le cœur de la révolution de l'informatique omniprésente (*ubiquitous computing*), rejoindront ce domaine et augmenteront exponentiellement le nombre de participants. Les architectures de réseau existantes qui emploient des nœuds centraux consacrés à la coordination des communications entre les participants (infrastructures centralisées), deviendront de plus en plus obsolètes du fait de l'augmentation rapide du nombre de participants et de dispositifs qui communiqueront plus probablement entre eux de manière décentralisée et autonome en mode dit « ad hoc ».

Un réseau ad hoc est un ensemble de dispositifs mobiles organisés en réseau sans l'aide d'aucune infrastructure fixe. Dans un tel réseau, les participants contribuent à l'acheminement des messages. La technologie de communica-

tion sans fil en mode ad hoc offre de nouveaux avantages dans beaucoup de situations grâce à son indépendance vis à vis d'une infrastructure fixe et à ses possibilités de déploiement et de reconfiguration autonomes. De plus, les nouvelles applications peuvent tirer profit de la caractéristique pair-à-pair de ces réseaux (échange direct d'informations entre deux nœuds qui sont à portée de communication) et offrir des services dépendants du lieu (*location based*) s'adressant à des communautés spécifiques d'utilisateurs.

Les protocoles de routage qui permettent la mise en œuvre de la communication en mode ad hoc sont essentiels pour le bon fonctionnement de ces réseaux. De nombreux travaux de recherche [54, 55, 31, 51, 52, 50, 43, 18] s'intéressent aux protocoles de routage dans les réseaux ad hoc. Dans ces protocoles, chaque nœud participant exécute un algorithme de routage qui permet de déterminer le prochain nœud vers lequel transmettre un message allant d'une source à une destination. Les protocoles qui assurent la communication dans les réseaux ad hoc doivent également tenir compte de la *mobilité* des participants et de la variation de connectivité.

La plupart des travaux sur le routage s'intéressent à des réseaux de taille moyenne avec, dans beaucoup de situations, une centaine de nœuds. Dans ces réseaux, des protocoles de routage différant par leur mode de fonctionnement peuvent être utilisés pour établir une communication entre les nœuds. Ainsi, les protocoles de routage proactifs tels que DSDV (*Destination Sequenced Distance Vector*) ou OLSR (*Optimized Link State Routing*) maintiennent des routes vers toutes les destinations possibles. Dans ces protocoles, des messages sont échangés périodiquement ou sur changement dans la topologie du réseau. En revanche, les protocoles de routage réactifs, tels que DSR (*Dynamic Source Routing*) et AODV (*Ad hoc On Demand Distance Vector*) créent et maintiennent des routes vers une destination « à la demande ». Ainsi, les messages de contrôle sont envoyés pour découvrir les routes uniquement lorsque cela est nécessaire.

Néanmoins, ces protocoles de routage ont des limitations quand il s'agit de les mettre en œuvre dans un réseau à très grand nombre de nœuds ou dans des réseaux dont la topologie est très étendue géographiquement. Dans ce type de réseaux, que l'on appelle réseau de grande taille, la complexité algorithmique et la quantité de l'information de routage peut croître exponentiellement. Ainsi, sans tenir compte des difficultés du passage à l'échelle des structures

des réseaux ad hoc, l'utilisation directe des protocoles classiques dans des réseaux de grande taille entraîne une dégradation du routage [27]. Cette dégradation, significative dans les applications ad hoc à grande échelle où des centaines ou même des milliers de dispositifs mobiles doivent être servis, peut être due à plusieurs raisons.

La première d'entre elles est l'existence de chemins longs entre source et destination, ce qui se produit naturellement dans un réseau de grande taille. Dans un cas typique, avec des centaines de nœuds déployés dans un large et vaste terrain, le nombre moyen de sauts entre une source et une destination pourra facilement dépasser 10. Une rupture de n'importe quel lien sur un tel chemin peut entraîner un échec.

La connectivité des nœuds est un autre problème des réseaux ad hoc de grande taille. Si la topologie du réseau est très étendue géographiquement et que les nœuds qui y participent ne sont pas assez nombreux, la distance les séparant peut dépasser largement la portée radio. Dans ce type de réseau les nœuds sont souvent déconnectés et l'information du routage peine à atteindre les nœuds les plus éloignés. Et, même si cette information arrive, elle peut déjà être obsolète.

Une autre raison est le surcoût généré par les protocoles de routage. Les protocoles de routage proactifs génèrent un surcoût en terme de bande passante et d'énergie chaque fois qu'un changement a lieu dans la topologie du réseau. En effet, chaque nœud doit recalculer sa table de routage tenant compte du changement. De plus, en raison de la mobilité il faut réduire la période des mises à jour pour garder un bon taux de remise de messages. Plus précisément, des messages de contrôle plus fréquents sont exigés pour maintenir l'information sur la topologie ; ces messages produisent un surcoût de contrôle des liens.

Dans un environnement mobile de grande taille, les protocoles de routage réactifs exigent qu'un chemin soit découvert complètement avant une éventuelle transmission de message. Ainsi, un nœud émetteur peut attendre longtemps pour que le chemin soit trouvé. De plus, l'existence de longs chemins, peut exiger de nouvelles recherches de routes et produire ainsi un surcoût de la bande passante et de l'énergie.

Pour résoudre ces problèmes, on peut exiger par exemple l'augmentation de la zone de transmission radio de tous les nœuds afin de réduire le nombre de sauts qu'un message doit faire pour atteindre sa destination. Mais augmenter la zone de transmission des nœuds peut s'avérer difficile voir impossible dans certains cas. En plus, l'augmentation de la zone de transmission radio des

nœuds va accroître la consommation d'énergie.

Connaissant les limitations des performances dans les réseaux ad hoc de grande taille, nous nous sommes intéressés à des techniques qui peuvent offrir des solutions à certains de ces problèmes. Nous pensons que les protocoles basés sur la position et la mobilité des nœuds sont mieux adaptés aux réseaux de grande taille que ceux se basant uniquement sur l'état de chaque lien du réseau complet, ou même d'une route particulière. Ainsi, nous proposons dans cette thèse une solution basée sur la mobilité des nœuds. Nous pensons que le mouvement des nœuds présente une opportunité : **si les nœuds se déplacent d'une manière « appropriée », alors les algorithmes de routage peuvent en tirer profit**. Ainsi, les nœuds vont se déplacer selon un schéma prédéfini afin de mettre en place un mécanisme de rendez-vous entre eux, dont le but final est d'assurer une communication fiable dans le réseau. Ce mécanisme n'empêche pas les nœuds d'accomplir d'autres tâches, mais exige seulement qu'ils se déplacent selon un schéma quand il s'agit de communiquer. Notre architecture, basée sur ce mécanisme, est robuste vis-à-vis de la mobilité des nœuds et assure un passage à l'échelle efficace.

Dans le cadre des réseaux mobiles ad hoc de grande taille, nous proposons dans cette thèse une architecture de contrôle de mobilité et des algorithmes de routage, conçus pour assurer une bonne qualité de communication, dans un environnement où les déconnexions sont fréquentes et où les protocoles de routage existants en mode ad hoc s'avèrent difficilement applicables. Ce travail essaye de résoudre certains des problèmes de ces réseaux par une approche qui concerne la mobilité de ces participants.

1.2 Démarche suivie et contributions

Les travaux au sein du groupe MANET (*Mobile Ad-hoc NETwork*) [40] de IETF (*Internet Engineering Task Force*) proposent des solutions pour le routage dans les réseaux mobiles ad hoc. Cependant ces travaux considèrent souvent des réseaux avec une topologie totalement connectée. Or, dans certains cas, ces solutions sont difficiles à appliquer. C'est le cas notamment des réseaux de grande taille à faible connectivité entre les nœuds.

Dans cette thèse, nous considérons le cas de réseaux où les nœuds sont souvent très éloignés les uns des autres créant ainsi une topologie déconnectée.

Notre but principal est d'assurer une communication entre les participants du réseau.

Dans ce type de réseau, la connaissance des positions des nœuds est essentielle pour beaucoup d'applications dans les réseaux sans fil. Ainsi on s'attend à ce que les nœuds du réseau soient équipés avec des dispositifs de localisation. En général, plusieurs systèmes de localisations sont proposés dans la littérature, parmi lesquels : le GPS (*Global Positioning System*), les systèmes de localisation basés sur l'infrastructure [56], et les systèmes de localisation ad hoc [61].

Afin de proposer une architecture de routage efficace, nous avons étudié le mouvement des nœuds. Nous pensons que le mouvement représente un facteur important pour l'amélioration des performances d'un réseau ad hoc. Ainsi, nous proposons une architecture de contrôle des mouvements des nœuds mobiles. Cette architecture est complètement distribuée et asynchrone, c'est-à-dire que les mouvements des nœuds ne dépendent pas d'une horloge commune. En contrepartie, nous avons mis en place un mécanisme de rendez-vous entre les nœuds du réseau. Les rendez-vous ont lieu dans certaines zones de l'environnement vers lesquelles les nœuds se déplacent pour communiquer. Le mouvement des nœuds vers ces zones est défini par un algorithme. Dans ces zones, les nœuds seront à portée radio les uns des autres et capables de communiquer. Grâce à l'utilisation des zones de rendez-vous, l'information de localisation des nœuds pourra être approximative. Ainsi, notre architecture est plus robuste et exige seulement que les nœuds connaissent leur position de façon approximative.

Dans notre architecture, les nœuds ne se déplacent pas dans toute la topologie du réseau mais seulement dans des zones prédéfinies à l'avance. Ceci réduit les effets de la mobilité car le surcoût dû aux mises à jour est ainsi limité. Une autre conséquence de l'utilisation de ces zones est la réduction de la consommation de l'énergie et de la bande passante car la zone de transmission des nœuds peut être réduite. Cette contrainte peut toutefois être relâchée.

Pour montrer la faisabilité de nos algorithmes nous avons implanté un simulateur en langage Java. Cela nous a permis de tester et de mesurer les performances de notre architecture et ensuite d'analyser les résultats obtenus. De plus, des tests réels ont été effectués avec des robots Lego Mindstorms [48] conçus et programmés pour simuler nos algorithmes.
Les contributions principales de cette thèse peuvent être résumés comme suit :

- conception d'une architecture de contrôle des mouvements afin d'améliorer les performances de communication dans un réseau mobile ad hoc ;
- étude d'une architecture tolérante aux pannes ;
- une plate-forme de tests pour des mesures de performance.

1.3 Structure du document

Pour étudier le routage dans les réseaux ad hoc et présenter nos solutions, nous avons d'abord étudié différents travaux existants. Ainsi, le chapitre 2 présente une classification des réseaux sans fil et une description des technologies utilisées. Le chapitre 3 présente un état de l'art sur la problématique du routage dans les réseaux ad hoc. Les protocoles de routage y sont présentés selon leur mode d'opération proactive, réactive ou hybride. Par ailleurs, la mobilité est une caractéristique importante des réseaux ad hoc. Ainsi, le chapitre 4 donne un état de l'art sur la mobilité dans les réseaux ad hoc et plus précisément sur les modèles de mobilité utilisés pour les simulations des protocoles de routage.

Dans la suite de la thèse, nous avons étudié la mise en œuvre d'une architecture de contrôle de mouvements pour améliorer les performances de routage. Pour cela, le chapitre 5 introduit la problématique de la mobilité et des effets qu'elle peut avoir sur le routage. Ce problème n'est pas encore très étudié et suscite de plus en plus intérêt. Ce chapitre présente aussi notre approche du problème et situe notre travail par rapport aux travaux présentés dans les chapitres précédents, notamment dans la classe des protocoles de contrôle des mouvements (*compulsory protocol*), définie par Hatzis et al. dans [37]. Le chapitre 5 sert en même temps de lien entre, d'une part, les chapitres qui présentent le routage et la mobilité et, d'autre part, les chapitres suivants qui présentent notre contribution.

Notre architecture de contrôle de mouvements est présentée à partir du chapitre 6. Ce chapitre décrit l'environnement, le type de réseau ad hoc et toutes les hypothèses de notre étude. Le chapitre 7 présente la modélisation de la topologie et un premier algorithme d'ordonnancement de mouvements dans cette architecture. Différentes propriétés émanant de cette modélisation et leurs preuves y sont aussi données. Ensuite, différentes améliorations de l'algorithme initial sont présentées dans le chapitre 8. Ces améliorations

aboutissent à un nouvel algorithme enrichi pour être tolérant aux pannes.

Pour évaluer notre travail, nous avons effectué des tests. Le chapitre 9 présente ces tests et les mesures de performances. Une description de la plate-forme que nous avons mise en place et des simulations y est faite, ainsi qu'une analyse des résultats.

Enfin, le chapitre 10 donne une conclusion sur notre travail et inclut des suggestions et perspectives pour les travaux futurs.

Chapitre 2

Les réseaux sans fil

Introduction

Dans le contexte du développement des réseaux de communication, les réseaux mobiles connaissent une très grande popularité : dans les bureaux avec les réseaux sans fil WiFi ou Bluetooth, dans des lieux publics avec les hotspots ou en déplacement grâce au GPRS et, bientôt, grâce aux réseaux de téléphonie de 3ème génération. Les prix deviennent de plus en plus abordables, les performances et les débits augmentent, tout comme les réseaux domestiques et le nombre d'utilisateurs.

Le domaine des réseaux sans fil ad hoc suscite un intérêt particulier. La spécificité de ce type de réseaux est qu'il n'a besoin d'aucune infrastructure fixe à l'inverse d'autres types de réseau sans fil comme le GSM/GPRS par exemple. Dans le principe, un réseau sans fil ad hoc est facile et rapide à déployer et ses composants sont tous libres de se déplacer. De ces déplacements, souvent imprévisibles, résultent des topologies dynamiques susceptibles de changer souvent et rapidement.

Lorsque deux composants du réseau n'arrivent pas à communiquer directement, des nœuds intermédiaires peuvent aider à acheminer des messages. Pour ce type de réseaux, les protocoles de routage classiques appliqués dans les réseaux filaires deviennent inefficaces. D'où la nécessité de proposer de nouveaux protocoles de routage capables de répondre à ces besoins spécifiques et qui prennent en compte des paramètres tels que la mobilité, une possible asymétrie des liens, des nœuds cachés, etc.

Dans ce contexte, MANET (Mobile Ad-hoc NETwork) [40] est un groupe

de travail dont le but est de normaliser les fonctionnalités au niveau IP des protocoles de routage adaptés aux topologies statiques et dynamiques des réseaux sans fil. Les issues fondamentales de conception sont basées sur certaines caractéristiques propres aux interfaces sans fil, et sur le fait que la topologie des nœuds peut faire face à une dynamique accrue, due au mouvement ou à d'autres facteurs.

Ce chapitre présente succinctement une classification des réseaux mobiles. Cette classification est faite selon des critères géographiques et topologiques. Il représente aussi les principales technologies et les modes de communication utilisés dans les réseaux ad hoc. Parmi ces technologies, Bluetooth et IEEE 802.11 peuvent être utilisés pour implémenter les algorithmes, présentés dans les chapitres 6 et 8, dans des robots comme dans [67, 48].

2.1 Classification des réseaux sans fil

Les réseaux sans fil peuvent être classés selon plusieurs critères. Parmi ceux-ci, une classification habituelle concerne l'étendue géographique. Cette classification est la suivante :

- les WPAN (*Wireless Personal Area Network*) se restreignent à des portées de quelques mètres pour mettre en relation des objets personnels (téléphone, ordinateur, clavier, souris, PDA, oreillette sans fil...) ;
- les WLAN (*Wireless Local Area Network*) couvrent quelques dizaines de mètres (un entrepôt, une partie de bâtiment...) et sont souvent utilisés pour raccorder un équipement mobile au réseau de l'entreprise (PC portable, PDA...) ;
- les WMAN (*Wireless Metropolitan Area Network*) couvrent quelques kilomètres pour offrir un service d'accès au niveau d'une ville ;
- les WWAN (*Wireless Wide Area Network*) vont au delà.

Une deuxième classification, moins classique et orthogonale à la précédente, consiste à classifier les réseaux sans fil selon la topologie et l'infrastructure utilisée. Ces topologies illustrées dans la figure 2.1, aboutissent à la classification suivante :

- **topologie totalement connectée :** chaque nœud peut communiquer avec chacun des autres nœuds *via* un lien direct. Cependant, en cas

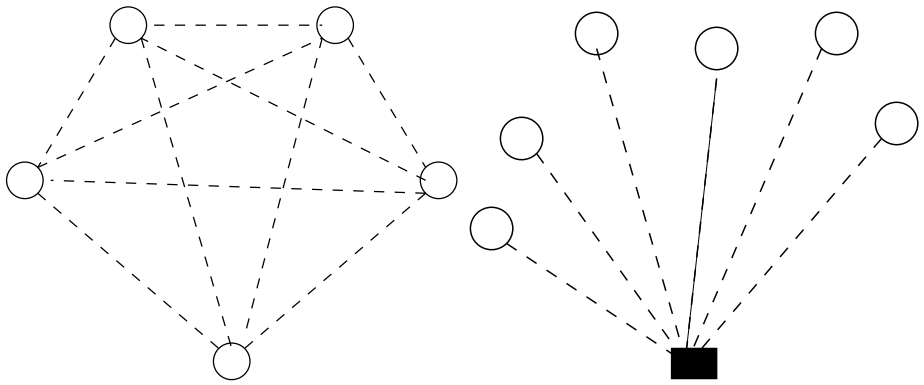
de rupture d'un lien entre deux nœuds, la communication entre eux devient impossible. Dans ce cas, quand un nœud se déplace, il faut que tous les autres le fassent aussi pour que le réseau reste connexe ;
- **topologie à station de base :** tous les nœuds ont un lien vers un nœud principal (station de base) par lequel passent les communications. Ce nœud est le point de passage pour communiquer avec un autre nœud. En cas de panne de la station de base, le réseau devient inopérant. En cas de déplacement d'un nœud, sa communication avec les autres peut être interrompue ;
- **topologie à routage interne :** le réseau est basé sur un routage interne entre ses membres. Cette architecture permet d'éviter les inconvénients des deux topologies précédentes ;
- **topologie hybride :** c'est une topologie qui combine une topologie filaire comme topologie principale et des liens sans fil formant la topologie secondaire. Comme exemple de cette topologie, on peut citer le réseau GSM.
- **topologie déconnectée :** dans cette architecture, les nœuds sont fréquemment hors de portée radio les uns des autres et le réseau est en général déconnecté. Par exemple, dans les réseaux ad hoc de grande taille, les nœuds sont souvent déconnectés à cause de la grande distance qui les sépare, mais aussi de leur mobilité.

Dans cette thèse, nous étudions le cas des réseaux de grande taille avec une topologie déconnectée. Ainsi, nous nous intéressons aux possibilités de faire communiquer entre eux n'importe quels nœuds du réseau dans un temps borné, ou réduire le nombre de sauts nécessaires pour qu'un message atteigne sa destination.

2.2 Technologies de réseaux sans fil

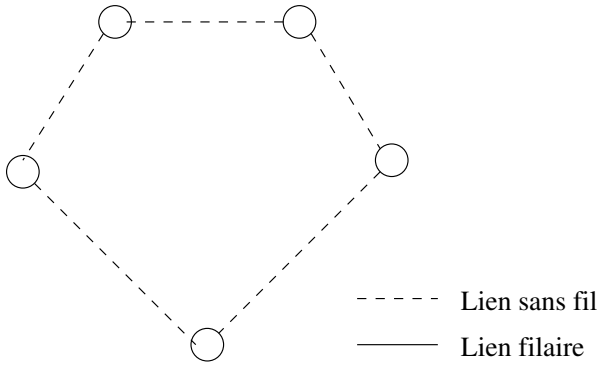
Cette section présente les principales technologies sans fil qui peuvent mettre en œuvre les idées développées dans cette thèse.

Le concept de réseau local sans fil est né du besoin d'offrir les mêmes services que les réseaux filaires. Entre les WPAN et WLAN il existe plusieurs standards tels que Bluetooth, IEEE 802.11 ou HIPERLAN, qui permettent aux nœuds mobiles des réseaux sans fil de communiquer entre eux dans un même réseau d'accès, c'est-à-dire que deux nœuds utilisant Blue-

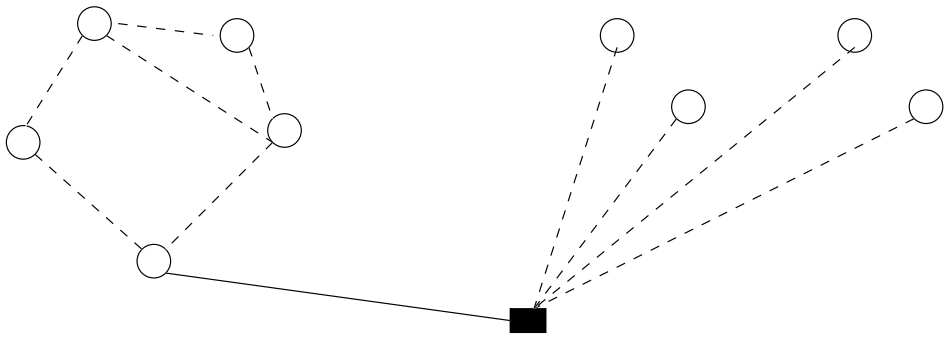


(a) Topologie connectée

(b) Topologie à station de base



(c) Topologie à routage interne



(d) Topologie hybride

FIG. 2.1 – Exemples de topologies de réseaux sans fil

tooth peuvent communiquer entre eux, mais pour communiquer avec des nœuds appartenant à un réseau qui utilise IEEE 802.11, il faut passer par les protocoles des couches supérieures comme TCP/IP ou UDP.

2.2.1 Bluetooth

Le standard Bluetooth [1] a été mis en place initialement par Ericsson rejoint en 1998 par un groupe de travail réunissant plusieurs grands industriels dont Agere, Nokia, IBM, Microsoft, Intel, Motorola et Toshiba. Bluetooth est une technologie sans fil qui fonctionne dans la bande de fréquence des 2,4 GHz. Elle permet d'atteindre en full duplex 1600 échanges par secondes ce qui donne au final, quand on enlève les informations de contrôle, un débit d'environ 1 mega bit par seconde (Mbps), mais avec une portée faible de quelques mètres seulement. Bluetooth repose sur les couches suivantes :

- couche physique radio. Au niveau physique, Bluetooth utilise la technologie par saut de fréquence FHSS (*Frequency Hoping Spread Spectrum*). Bluetooth permet de construire de petits réseaux personnels. Le réseau de base appelé *piconet* est formé d'un maître qui peut administrer plusieurs esclaves. La communication est directe entre le maître et un esclave et les esclaves ne peuvent pas communiquer entre eux. Les périphériques esclaves peuvent avoir plusieurs maîtres, et les différents piconets peuvent donc être reliés entre eux. Le réseau ainsi formé est appelé un *scatternet* (réseau chaîné) ;
- couche *Baseband*. Cette couche permet de définir trois types de liens : les liaisons SCO (*Synchronous Connection-Oriented*) pour l'audio (ou audio et données), les liaisons ACL (*Asynchronous Connectionless*) pour les données et enfin les liaisons de base pour toute la gestion des connexions au sein du piconet ;
- *Link Manager Protocol*. Ce protocole est responsable de la supervision des différentes connexions, de l'authentification des appareils, et du chiffrement. Il gère également les mises en veille des différents appareils ;
- *Link Layer Control and Adaptation (L2CAP)*. Cette couche permet l'adaptation des protocoles des couches supérieures (comme TCP/IP) au réseau Bluetooth.

La proposition Bluetooth a été prise en compte par l'IEEE dans le groupe de travail 802.15 qui s'occupe de la standardisation des réseaux personnels,

WPAN (*Wireless Personnal Area Network*).

2.2.2 IEEE 802.11

La norme IEEE 802.11 (ISO/IEC 8802-11, connue sous le nom WiFi (*Wireless Fidelity*) [5] est un standard international décrivant les caractéristiques d'un réseau local sans fil (WLAN). La norme initiale IEEE 802.11 offre des débits de 1 ou 2 Mbps. Les révisions apportées à la norme initiale permettent pour la norme 802.11b d'atteindre des débits de 5.5 et 11 Mbps. La norme 802.11a offre un débit de 54 Mbps dans la gamme des fréquences 5 GHz tandis que la norme 802.11g offre un débit de 54Mbps dans la gamme de fréquences de 2.4 Ghz. La norme 802.11 s'attache à définir les couches basses du modèle OSI pour une liaison sans fil radio utilisant des ondes électromagnétiques.

La couche physique définit la modulation des ondes radioélectriques et propose trois types de codage de l'information. Elle définit aussi les caractéristiques de la signalisation pour la transmission de données. La méthode d'accès utilisée est proche du standard Ethernet. Le standard 802.11 définit deux modes opératoires :

- le mode infrastructure dans lequel les nœuds sont connectés à un point d'accès nécessaire pour que deux mobiles communiquent. En fait ces points d'accès servent de point de passage pour les communications entre les nœuds. Il s'agit généralement du mode par défaut des cartes 802.11b [49] et qui correspond à la topologie à station de base présentée dans la section 2.1 ;
- le mode ad hoc, dans lequel les nœuds peuvent se connecter les uns aux autres sans aucun point d'accès à condition qu'ils soient à portée radio les uns des autres. C'est le mode qui correspond le mieux aux architectures qui nous intéressent dans cette thèse.

2.2.3 HIPERLAN

HiperLAN est une technologie développée par l'ETSI (*European Telecommunication Standard Institute*). Deux versions de ce standard existent, HiperLAN1 [25, 20] et HiperLAN2 [21, 22, 23, 24], qui peuvent fonctionner ensemble. Ce standard utilise une bande de fréquence proche des 5 GHz. Le débit théorique proposé par HiperLAN1 est proche de 20 Mbps et celui de HiperLAN2 est de 54 Mbps. La zone de couverture dépend du milieu. En effet,

la fréquence ayant une longueur d'onde plus petite, celle-ci est plus sensible aux obstacles. L'architecture d'HiperLAN repose sur les couches suivantes :

- la couche physique qui utilise des types de codage et de modulation différentes pour les normes HiperLAN1 et HiperLAN2.
- la couche liaison de données qui fait le lien entre les points d'accès et les terminaux mobiles. Elle inclut les fonctions d'accès au média, de transmission et de gestion de la connexion. Cette couche est divisée en deux sous-couches, la sous-couche CAC (*Channel Access Control*) qui correspond à la partie physique de la technique d'accès (gestion des problèmes liés au canal hertzien ainsi que toute la transmission et réception) et la sous-couche MAC qui correspond à la partie logique, soit la mise en forme de la trame, le routage interne, les algorithmes de confidentialité, la gestion de priorité (QoS) et l'insertion et le retrait des stations.
- la couche dite de convergence. Elle a deux fonctions principales : l'adaptation des services demandés par les couches hautes aux services proposés par la couche de liaison des données, et toute la partie fragmentation, réassemblage et bourrage, afin d'adapter les trames de n'importe quel réseau pour le transport sur le réseau HiperLAN2.

2.3 Modes de communication

Les sections précédentes présentent les topologies et les technologies qui peuvent être mises en œuvre dans un réseau sans fil afin que ses participants puissent communiquer. Pour établir une communication dans de tels réseaux, on peut distinguer deux approches.

La première consiste à avoir recours à des stations de base comme dans les réseaux à infrastructure cellulaire. Un problème dans ce type de communication est celui du *handoff*, qui conduit à la perte de données ou à l'augmentation des délais de transmission lors du changement de station de base (changement de cellule). Un autre problème est dû au fait que dans certains cas, il n'est pas possible de déployer une infrastructure fixe et dans d'autres cas ce déploiement peut s'avérer trop coûteux. Dans cette thèse nous ne nous sommes pas intéressés à ce type de réseaux.

Une autre solution consiste à former un réseau ad hoc. Cette seconde approche implique que les utilisateurs doivent être prêts à acheminer des pa-

quets qui ne leur sont pas destinés, afin d'assurer la communication entre une source et une destination. Les réseaux mobiles ad hoc sont idéalement appropriés pour des applications telles que des missions militaires, d'exploration ou de secours, où une infrastructure fixe est indisponible ou détruite. Mais indépendamment de ces situations de secours ou d'urgence, il y a beaucoup d'exemples qui s'adaptent naturellement au modèle « sans infrastructure fixe ». C'est notamment à ce type de communication que nous nous sommes intéressés dans cette thèse. Puisque les composants de tels réseaux acheminent des paquets, il leur faut donc des protocoles de routage adaptés pour leur permettre de prendre des décisions. Le chapitre suivant présente les principaux protocoles existants en la matière.

Chapitre 3

Le routage dans les réseaux mobiles ad hoc

Introduction

Un réseau mobile ad hoc (MANET) est un ensemble de nœuds sans fil constituant un réseau sans l'aide d'une infrastructure fixe ou d'une administration centralisée. D'une façon générale un MANET (RFC 2501) [26] se compose de dispositifs (nœuds) portatifs qui sont caractérisés par des limitations de capacité de traitement et de mémoire. Ainsi, il ne sera pas possible qu'un nœud maintienne les informations de routage pour tous les nœuds dans un grand réseau. Dans un réseau ad hoc, si deux nœuds veulent communiquer et qu'ils ne sont pas dans la zone de transmission sans fil l'un de l'autre, ils ne pourront communiquer que si d'autres nœuds, participant également au réseau ad hoc, sont disposés à acheminer des paquets pour eux : ces nœuds fonctionnent alors comme routeurs comme il est illustré dans la figure 3.1.

Ce chapitre présente la problématique des réseaux ad hoc et certains protocoles qui sont proposés et développés par le groupe de travail MANET au sein de l'IETF. Récemment, ce groupe de travail a retenu les protocoles AODV (RFC 3561) [53], DSR [42], OLSR (RFC 3626) [18] et TBRPF (RFC 3684) [50].

Dans cette thèse, nous nous intéressons au routage dans un type particulier de réseau ad hoc. Ainsi, il était intéressant d'étudier la possiblité d'utiliser les protocoles de routage existants pour voir s'ils s'adaptaient à notre situation.

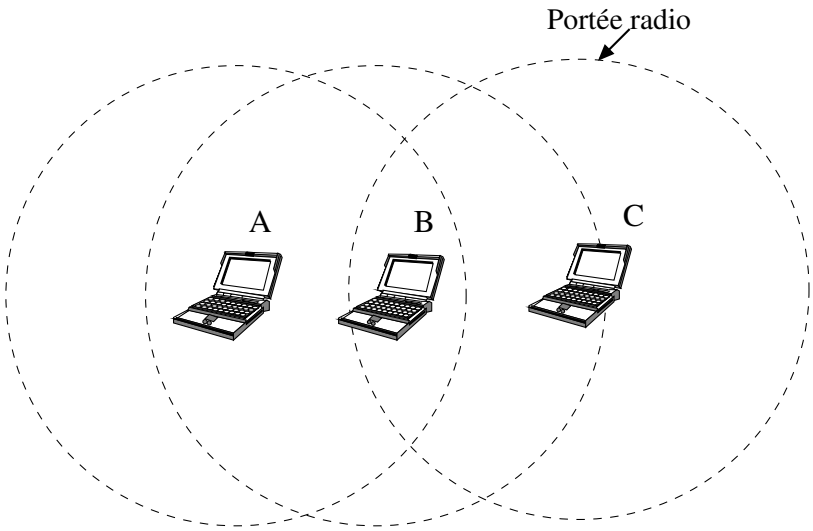


FIG. 3.1 – Exemple de réseau ad hoc

3.1 Problématique des réseaux ad hoc

Les MANETs ont plusieurs caractéristiques qui doivent être prises en compte lorsqu'on considère leur conception et leur déploiement :

- topologies dynamiques : la topologie du réseau (qui est en général multi-sauts[1]) peut changer aléatoirement, rapidement, et de manière imprévisible. Elle peut être composée de liens bidirectionnels et unidirectionnels ;
- liens de capacités variables et contraintes sur la taille de la bande-passante : les liens sans fil ont des capacités sensiblement inférieures à leurs contreparties câblées. En outre, le débit des communications sans fil est très variable en fonction de la technologie utilisée, de la topologie du réseau etc. ;
- contraintes d'énergie : certains ou tous les nœuds dans un MANET peuvent fonctionner avec des batteries ou d'autres moyens épuisables pour leur énergie. Pour ces nœuds, une capacité limitée d'énergie peut être la contrainte la plus significative et son utilisation devra être regardée ainsi comme paramètre primaire du réseau ;

[1] Un saut est une connexion directe entre deux nœuds

– sécurité physique limitée : les réseaux mobiles sans fil sont généralement plus vulnérables aux menaces physiques de sécurité que le sont les réseaux câblés. La possibilité d'écoute clandestine (*eavesdropping*), d'intrusion (*spoofing*), et d'attaques de dénis de service (*denial of service*) devront être soigneusement considérées. Les techniques existantes de sécurité sont souvent appliquées dans les réseaux sans fil pour réduire les menaces de sécurité.

Tandis que les caractéristiques ci-dessus sont communes à n'importe quel genre de réseau sans fil, les MANET sont encore distingués par leur propriété « sans infrastructure fixe » : il n'y a aucune administration centralisée ou infrastructure préalable qui prend soin de la gestion et de l'existence du réseau. Les nœuds mobiles sont eux-mêmes responsables d'établir et de maintenir la connectivité entre eux. Les opérations robustes et efficaces dans les réseaux mobiles sans fil sont assurées en incorporant la fonctionnalité de routage dans tous les nœuds mobiles, contrairement aux réseaux câblés tels que l'Internet où seulement quelques nœuds dans le réseau exécutent la fonction de routage. Il est évident que la fonction de routage est primordiale pour la viabilité d'un réseau ad-hoc.

De nombreux protocoles sont proposés pour résoudre le problème de routage multi-saut, chacun fondé sur différents concepts et reposant sur différentes hypothèses et intuitions. Les protocoles de routage peuvent être séparés en deux classes principales, à savoir *proactif* ou *réactif*, selon les manières dont les routes sont créées et maintenues. On pourra ajouter une troisième classe, celle des protocoles hybrides qui sont une combinaison des protocoles proactifs et réactifs. Les sections suivantes présentent plus en détail les principaux protocoles retenus par le groupe de travail MANET, mais aussi quelques autres protocoles appartenant à ces classes.

3.2 Les protocoles de routage proactifs

Les protocoles de routage proactifs pour les réseaux mobiles ad hoc sont basés sur la philosophie des protocoles de routage traditionnels utilisés dans les réseaux de commutation de paquets (*packet switching networks*). Dans ces protocoles, deux principales méthodes de routage sont utilisées : le routage par état de liens (*link state*) et le routage par vecteur de distance (*distance vector*). Les protocoles proactifs ont l'avantage de la disponibilité immédiate des routes vers tous les nœuds du réseau. Cependant, un trafic de contrôle

important est nécessaire pour mettre à jour les routes.

3.2.1 Destination Sequenced Distance Vector Protocol

Le protocole *Destination Sequenced Distance Vector* noté DSDV [55, 14] est un protocole de routage à vecteur de distances qui a été spécifiquement developpé pour les réseaux mobiles. L'avantage principal de DSDV sur les protocoles traditionnels à « vecteur de distances » est qu'il garantit un routage sans boucle. L'algorithme sur lequel DSDV est basé étend l'algorithme classique de Bellman-Ford [69] en étiquetant chaque entrée du vecteur de distances par un numéro de séquence assigné par le nœud destination. De cette façon, les nœuds peuvent rapidement distinguer les routes périmées (*stale routes*) des routes neuves et éviter ainsi la formation des boucles. Chaque nœud de DSDV maintient une table de routage présentant le prochain saut pour chaque destination accessible. Quand un nœud mobile reçoit une nouvelle information de routage, cette information est comparée à l'information déjà fournie par les paquets précédents. Une route R est meilleure que R' si R a un plus grand numéro de séquence ou si les deux routes ont des numéros de séquence égaux mais R a une métrique inférieure. La métrique employée dans l'algorithme pour déterminer les chemins les plus courts est le « comptage des sauts ». Pour maintenir la cohérence des tables de routage dans une topologie variable dynamiquement, chaque nœud transmet périodiquement et sur changement de topologie des mises à jour. Ces mises à jour concernent seulement les changements et pas toute la table de routage du nœud. La diffusion (*broadcast*) par chaque nœud mobile contient son nouveau numéro de séquence incrémenté et l'information suivante pour chaque nouvelle route : l'adresse de destination, le nombre de sauts requis pour atteindre la destination et le numéro de séquence de l'information reçue concernant la destination. Quand un nœud N décide que sa route vers une destination D n'existe plus, il annonce la route à D avec une métrique infinie et un numéro de séquence égal à son numéro de séquence plus 1, pour la route qui n'existe plus. Ainsi n'importe quel nœud A qui achemine des paquets par N, inclut la métrique infinie de la route à sa table de routage, jusqu'à ce que le nœud A trouve une route vers D avec un numéro de séquence plus élevé. En plus de l'élimination des boucles, il est demontré que DSDV évite aussi le problème du comptage à l'infini.

3.2.2 Optimized Link State Routing

Le protocole *Optimized Link State Routing* noté OLSR [18, 19] est un protocole à état de liens optimisé. Alors que dans un protocole à état de liens classique, chaque nœud déclare ses liens directs avec ses voisins à tout le réseau, dans le cas de OLSR, les nœuds ne déclarent qu'une sous-partie de leur voisinage grâce à la technique des relais multipoints. Les relais multipoints consistent essentiellement en un nœud donné, à ignorer un ensemble de liens et de voisins directs, qui sont redondants pour le calcul des routes de plus court chemin : plus précisément, dans l'ensemble des voisins d'un nœud, seul un sous-ensemble des ces voisins est considéré comme pertinent. Cet ensemble, appelé l'ensemble des relais multipoints, est choisi de façon à pouvoir atteindre tout le voisinage à deux sauts (tous les voisins des voisins). Les relais multipoints sont utilisés :

- pour diminuer le trafic dû à la diffusion des messages de contrôle dans le réseau, et aussi
- pour diminuer le sous-ensemble des liens diffusés à tout le réseau puisque les routes sont construites à l'aide de relais multipoint comme il est montré dans la figure 3.2.

La diffusion d'un message par relais multipoints diminue le nombre de retransmissions en utilisant la règle de suivante : un nœud retransmet un message si et seulement si :

- il ne l'avait pas déjà reçu, et
- il vient de le recevoir d'un nœud dont il est un relais multipoint.

Pour maintenir à jour toutes les informations nécessaires au choix des relais multipoints, mais aussi au calcul de la table de routage, les nœuds OLSR ont besoin de s'échanger des informations périodiquement :

- pour s'informer du proche voisinage. Les nœuds OLSR envoient périodiquement des messages dits HELLO contenant la liste de leurs voisins. Ces messages permettent à chacun de choisir son ensemble de relais multipoints ;
- pour déclarer périodiquement dans le réseau les sous-ensembles de voisinage que constituent les relais multipoints. Ce deuxième type de message de OLSR sont appelés messages TC (*Topology Control*). Leur dif-

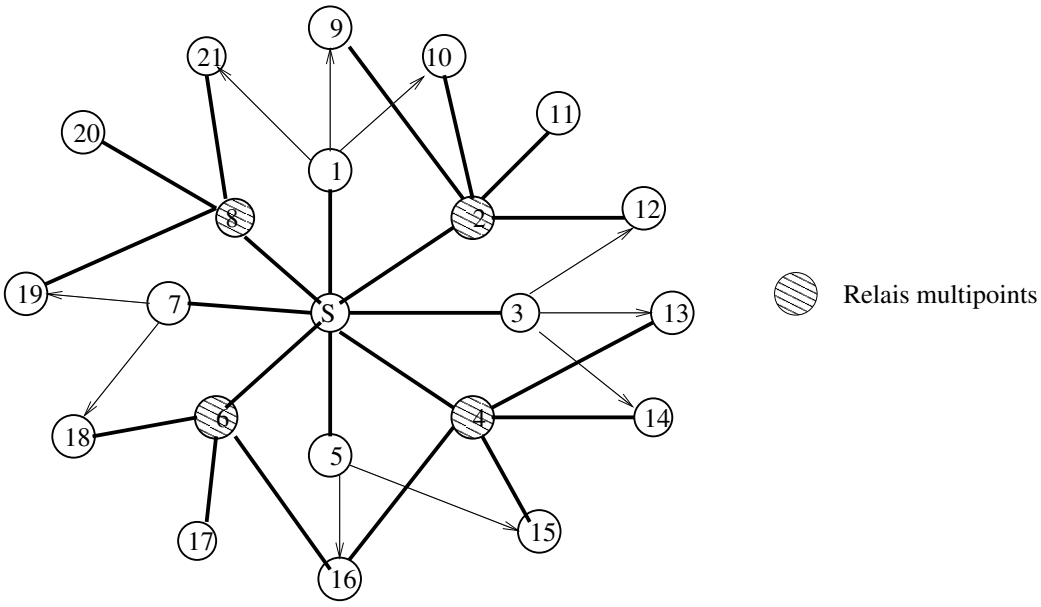


FIG. 3.2 – Relais multipoints

fusion est optimisée par les relais multipoints.

Ces informations offrent une carte du réseau contenant tous les nœuds et un ensemble partiel des liens, mais suffisant pour la construction des tables de routage. Chaque table de routage est calculée par chacun des nœuds et le routage des données s'effectue saut par saut sans l'intervention de OLSR dont le rôle s'arrête à la mise à jour des tables de routage de la pile IP.

3.2.3 Topology Dissemination Based on Reverse-Path Forwarding

Le protocole *Topology Dissemination Based on Reverse-Path Forwarding* noté TBRPF [50] est un protocole proactif à état de liens qui assure un routage par sauts sur les chemins les plus courts vers une destination. TBRPF est basé sur deux mécanismes :

- un pour la découverte des voisins qui se fait par un échange de messages HELLO dits différentiels

– et un autre pour la découverte de la topologie et le calcul de routes.

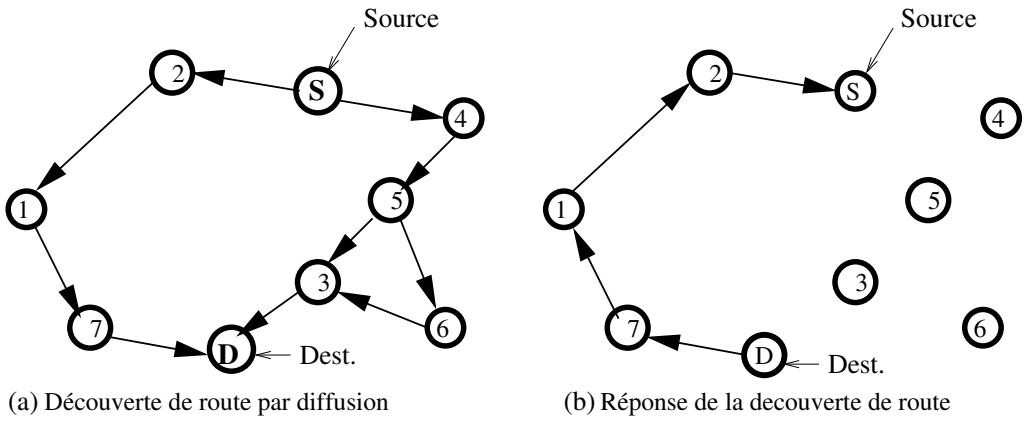
Pour la gestion de la topologie, chaque nœud calcule un arbre fournissant des chemins vers tous les nœuds accessibles. Le calcul se fait à partir d'une information partielle stockée dans sa table de topologie, et en utilisant une variante de l'algorithme de Dijkstra [29]. Ainsi, les arbres de deux voisins peuvent avoir certaines similitudes. Pour éviter cela, mais aussi pour réduire le trafic de contrôle, chaque nœud propage seulement une partie de son arbre à ses voisins.

Un nœud TBRPF emploie des messages différentiels de mises à jour pour tenir tous les voisins au courant sur les changements de son arbre. Chaque nœud peut également fournir des informations additionnelles sur la topologie, et éventuellement sur toute la topologie. À l'instar de OLSR où chaque nœud choisit un sous ensemble de ses voisins comme des relais multipoint, TBRPF exige de chaque nœud qu'il détermine s'il est un relais (père dans l'arbre) pour un autre nœud.

3.3 Les protocoles de routage réactifs

Au contraire des protocoles proactifs qui mettent à jour leurs informations de routage systématiquement, les protocoles réactifs établissent les routes uniquement à la demande. Ainsi, les protocoles de routage réactifs ou les protocoles « à la demande » créent et maintiennent des routes seulement si cela est nécessaire. Ils essayent de garder et de maintenir les routes les plus courtes vers toutes les destinations utilisées. La découverte des routes peut être très gourmande, particulièrement pour les réseaux de grande taille. Un schéma de fonctionnement de ce type de protocoles est montré dans la figure 3.3.

Quand une route est nécessaire, une procédure globale de découverte de route est lancée pour trouver le chemin vers une destination spécifique dont la route est inconnue. La découverte des routes est habituellement faite en utilisant les mécanismes classiques de diffusion (*broadcast*). Par exemple, dans la figure 3.3(a), si le nœud S veut communiquer avec le nœud D et qu'il ne connaît pas de route vers D, alors S fait une diffusion de paquets de découverte dans le réseau. Dès que le nœud D reçoit un paquet de diffusion, alors il renvoie un paquet réponse pour annoncer au nœud S qu'une route a été trouvé vers D (illustré dans la figure 3.3(b)).



(a) Découverte de route par diffusion (b) Réponse de la decouverte de route

FIG. 3.3 – Découverte de route utilisée par les protocoles reactifs

3.3.1 Dynamic Source Routing

Le protocole *Dynamic Source Routing* noté DSR [42, 14], utilise le routage par la source (*source routing*). Selon cette technique, si un nœud veut envoyer un paquet à un autre nœud, la source du paquet construit une route dans l'en-tête du paquet, donnant l'adresse de chaque nœud dans le réseau par lequel le paquet devrait être acheminé afin d'atteindre la destination. La source met explicitement cette route dans l'en-tête du paquet et transmet alors le paquet via son interface réseau au premier nœud à un saut. Quand un nœud reçoit le paquet, si ce nœud n'est pas la destination finale, il enlève de l'en-tête du paquet sa propre adresse et transmet simplement le paquet au prochain saut identifié dans l'en-tête du paquet. Ce processus continue jusqu'à ce que le paquet atteigne sa destination. Chaque nœud mobile participant au réseau ad hoc maintient un cache de routes dans laquelle il mémorise les routes qu'il a apprises pendant les différents acheminements qu'il a faits. Quand un nœud envoie un paquet à un autre nœud, la source contrôle d'abord sa mémoire cache pour trouver une route vers la destination. Si une route est trouvée, la source utilise cette route pour transmettre le paquet. Sinon, l'émetteur utilise le mécanisme de découverte de route. Le protocole DSR se compose donc de deux mécanismes : découverte de routes et maintenance de routes.

Ce protocole découvre des routes à la demande en inondant le réseau d'une façon contrôlée (par exemple en employant un champ *Time to Live*

(TTL) dans l'en-tête de paquet). Dans la découverte des routes, le nœud source diffuse un paquet *ROUTE REQUEST* dans lequel les nœuds qu'il a traversés s'enregistrent. Une fois qu'un tel paquet atteint la destination, celle-ci répond avec un paquet *ROUTE REPLY* qui copie simplement la route du paquet *ROUTE REQUEST* et l'achemine en sens inverse. La mémoire cache des routes de chaque nœud réduit le coût et la fréquence de la découverte des routes. Chaque nœud stocke les routes connues quand il achemine des paquets de réponse *ROUTE REPLY*. Ces routes sont utilisées par les paquets de données.

La procédure de maintenance des routes surveille le fonctionnement des routes et informe la source de toutes les erreurs de routage. La détection des erreurs se fait dans la couche liaison de données. Si cette couche signale un problème de transmission qu'il ne peut pas résoudre, ce nœud envoie un paquet *ROUTE ERROR* à la source du paquet. Ce paquet d'erreur de route, *ROUTE ERROR*, contient les adresses des nœuds aux deux extrémités du saut qui génére l'erreur, le nœud qui a détecté l'erreur et le nœud auquel il essayait de transmettre ce paquet sur ce saut. Quand un paquet *ROUTE ERROR* est reçu par le nœud source, le saut de l'erreur est enlevé de la mémoire cache de ce nœud et toutes les routes qui contiennent ce saut sont tronquées à ce point. Puis, une nouvelle découverte de route vers la destination est lancée par l'émetteur.

DSR a un avantage majeur lié au routage par la source qui est que les nœuds intermédiaires n'ont pas besoin de maintenir l'information de mise-à-jour afin de router les paquets, puisque les paquets eux-mêmes contiennent déjà toutes les décisions de routage. Cette caractéristique, ainsi que la nature à la demande du protocole, élimine le besoin d'envoyer des annonces périodiques des routes et des paquets pour détecter les voisins du nœud, présents dans d'autres protocoles. Une autre particularité de DSR est le fait que la route fait partie du paquet lui-même. Ainsi les boucles sont évitées, car elles sont immédiatement détectées et éliminées. Ces propriétés rendent le protocole ouvert à une variété d'optimisations. Par exemple, une inondation de requêtes peut être éliminée dès le début en faisant répondre n'importe quel nœud non-destination, s'il sait qu'il n'est pas possible d'atteindre la destination prévue. Un nœud peut connaître une route vers une destination en acheminant des paquets de *ROUTE REPLY*. En outre les routes peuvent être améliorées en échangeant des informations entre les nœuds voisins.

3.3.2 Ad Hoc On Demand Distance Vector Protocol

Le protocole *Ad Hoc On Demand Distance Vector Protocol* noté AODV [53, 14] est essentiellement une combinaison de DSR et de DSDV. Il emprunte le mécanisme de découverte des routes à la demande et la maintenance des routes à DSR auxquels il ajoute l'utilisation du routage par sauts, les numéros de séquence et les messages périodiques de DSDV. Le protocole AODV assure la non-existence des boucles. En évitant le problème du comptage à l'infini, il offre une convergence rapide quand la topologie du réseau ad hoc change (typiquement, quand un nœud se déplace dans le réseau). Quand il y a une rupture de lien, AODV nécessite que l'ensemble des nœuds affectés soient avertis afin qu'ils puissent invalider les routes utilisant le lien cassé. AODV emprunte à DSDV l'utilisation d'un numéro de séquence pour chaque route. Le numéro de séquence est créé par le nœud destination pour chaque information (sur les routes), qu'il envoie à tous les nœuds qui lui demandent. L'utilisation des numéros de séquence assure un réseau sans boucle et un protocole simple à programmer. Étant donné le choix entre deux routes vers une destination, un nœud choisit toujours celle avec le numéro de séquence le plus grand.

Le mécanisme de AODV opère comme suit : quand un nœud a besoin d'une route vers une destination, il diffuse un message *ROUTE REQUEST* à ses voisins. Le dernier numéro de séquence connu de la destination est inclus dans ce message *ROUTE REQUEST*, qui est diffusé par le réseau jusqu'à ce qu'il atteigne la destination ou un nœud qui a une route vers la destination. Chaque nœud recevant la demande stocke une route de retour vers la source de la demande. La route est rendue disponible par l'envoi d'un *ROUTE REPLY* en sens inverse vers la source du message *ROUTE REQUEST*. Comme dans le protocole DSR, chaque nœud se rappelle seulement du prochain saut et pas de la route toute entière. Pour chaque route valide (contenant une métrique finie de sauts) maintenue par un nœud, celui-ci maintient également une liste de prédécesseurs qui peuvent acheminer des paquets sur cette route. Ces prédécesseurs recevront des annonces de ce nœud en cas de détection de rupture du prochain saut. La liste de prédécesseurs dans une entrée de la table de routage contient les nœuds voisins pour lesquels une réponse de route (*ROUTE REPLY*) a été générée ou acheminée. Ainsi, AODV maintient les routes d'une manière distribuée. Les entrées de la table de routage sont sous la forme de *(destination, prochain saut, distance)*. Les tables de routage distribuées d'une telle manière sont plus petites que celles du protocole

DSR où chaque nœud doit maintenir le chemin complet dans sa mémoire cache. Cependant, afin de maintenir les routes, AODV exige normalement que chaque nœud transmette périodiquement un message *HELLO*. Le fait de ne pas recevoir trois messages *HELLO* consécutifs d'un voisin est consideré comme une indication que le lien vers la destination en question n'existe plus. À la réception d'un message *ROUTE ERROR*, un nœud doit découvrir une nouvelle route vers la destination en utilisant le mécanisme de la découverte des routes décrit précédemment. Un nœud lance un message *ROUTE ERROR* dans trois situations :

- s'il détecte une rupture de lien vers le prochain nœud d'une route active dans sa table de routage ;
- s'il obtient un paquet de données destiné à un nœud pour lequel il n'a pas de route active ;
- s'il reçoit un message *ROUTE ERROR* d'un voisin pour une ou plusieurs routes actives.

Pour les deux premiers cas, les numéros de séquence de destination dans la table de routage pour la(s) destination(s) inaccessible(s) sont incrémentés de 1. Alors, le message *ROUTE ERROR* est diffusé avec la(s) destination(s) inaccessible(s) et leurs numéros de séquence incrémentés.

Dans le premier cas, les destinations inaccessibles sont le prochain nœud vers lequel il n'existe plus ce saut, et toutes les destinations qui sont maintenant inaccessibles suite à la perte de ce saut. Ces destinations additionnelles sont celles qui utilisent également le lien rompu en tant que saut suivant dans leur table de routage.

Dans le deuxième cas, il y a seulement une destination inaccessible, qui est la destination du paquet de données qui ne peut pas être livré. Le champ *DestCount* du paquet *ROUTE ERROR* indique le nombre de destinations inaccessibles incluses dans le paquet.

Dans le troisième cas, quand un nœud reçoit un message *ROUTE ERROR*, pour chaque destination inaccessible incluse dans le paquet, le nœud détermine si le nœud source du paquet *ROUTE ERROR* est son prochain saut vers cette destination.

Tout comme DSR, AODV fait en sorte que chaque nœud ayant une route vers la destination puisse répondre à une demande de route. AODV utilise également une technique appelée échéance de route (*route expiry*) selon laquelle une entrée de la table de routage expire après une période prédétermi-

née. À la fin de celle-ci une découverte de route doit être lancée. À la différence du protocole OLSR, les protocoles DSR et AODV ne garantissent pas des chemins les plus courts. Cependant, une évaluation des performances [14, 17] prouve que les longueurs des routes découvertes sont habituellement proches de celles trouvées dans les protocoles qui utilisent les chemins les plus courts.

3.4 Les protocoles hybrides

Les protocoles hybrides combinent les mécanismes de routage des protocoles proactifs et réactifs. Ils utilisent un protocole proactif pour connaître les voisins (à deux ou trois sauts) et au delà de cette zone prédéfinie, le protocole hybride utilise des techniques de protocoles réactifs pour découvrir des routes. Le réseau est découpé en plusieurs zones et la découverte des routes est ainsi améliorée. Si un nœud reçoit une requête de recherche réactive, il peut déterminer si la destination est un de ses voisins grâce à la technique proactive utilisée pour connaître ses voisins. Sinon, il peut acheminer la requête vers les autres zones sans impliquer les nœuds participants dans sa zone. Ces protocoles s'adaptent bien aux grands réseaux mais ils ont aussi en partie les inconvénients des protocoles proactifs et réactifs : par exemple le coût de découverte de nouvelles routes reste important.

3.4.1 Zone Routing Protocol

Le protocole *Zone Routing Protocol* [35] peut être classé comme protocole de routage hybride réactif/proactif. Comme son nom l'indique, il est basé sur le concept des zones (figure 3.4). Une zone de routage est définie pour chaque nœud et les zones des nœuds voisins se recouvrent. Ainsi, la zone d'un nœud inclut les nœuds qui sont à moins de ρ sauts. Les nœuds dans une zone sont divisés en nœuds périphériques et nœuds intérieurs. Par exemple, dans la figure 3.4, les nœuds en gris 6, 7 et 8 sont des nœuds périphériques et les nœuds 1, 2, 3, 4 et 5 sont des nœuds intérieurs pour la zone du nœud S.

Le protocole ZRP utilise un protocole proactif local appelé *IntrA-zone Routing Protocol (IARP)* pour le routage vers les destinations qui se trouvent à l'intérieur de la zone et un autre protocole appelé *IntEr-zone Routing Protocol (IERP)* pour le routage réactif entre les zones. Le fait que la topologie d'une zone locale soit connue, permet à ZRP de ne pas utiliser de diffusion pour découvrir une route locale. Mais quand une découverte globale de route

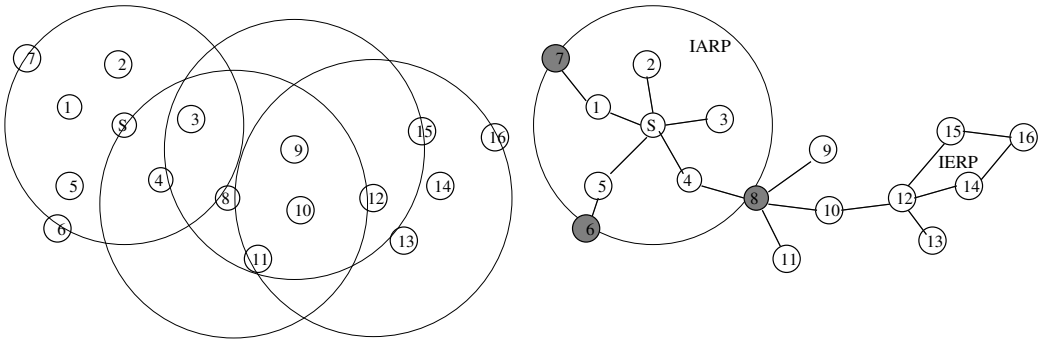


FIG. 3.4 – Zone de routage avec $\rho = 2$

est nécessaire, ZRP utilise le concept de *bordercast*. Ce service de routage est fourni par le protocole de résolution *Bordercast (BRP)*. Celui-ci utilise les informations sur la topologie fournies par IARP pour construire la liste des nœuds périphériques ainsi que la façon de les atteindre. BRP est utilisé pour acheminer la diffusion des requêtes de découverte de routes dans le réseau.

Dans un réseau ad hoc, on peut supposer que la plus grande partie du trafic est dirigée vers des nœuds voisins. Dans une zone limitée, l'entretien de l'information de routage est plus facile. De plus, la quantité de l'information de routage qui n'est jamais utilisée est réduite au minimum. Les nœuds plus lointains peuvent être atteints avec le routage réactif. Puisque tous les nœuds stockent de manière proactive l'information locale de routage, les demandes de routes peuvent être faites plus efficacement sans interroger tous les nœuds du réseau. Le comportement de ZRP est adaptatif et il dépend de la configuration courante du réseau et du comportement des utilisateurs.

3.5 Résumé

Tous les protocoles présentés dans ce chapitre sont distribués. Ainsi, aucun n'est dépendant d'une infrastructure fixe et ils peuvent s'adapter aux changements de topologie. Parmi ces protocoles, OLSR, AODV et ZRP sont les premiers qui ont fait l'objet d'un RFC (*Request For Comments*). OLSR a besoin d'échanges périodiques de messages de contrôle, pour maintenir les routes vers chaque nœud OLSR du réseau.

AODV peut être considéré comme la version réactive du protocole DSDV.

AODV utilise un mécanisme de découverte de routes similaire à DSR, mais un paquet DSR doit contenir toute l'information du routage, tandis qu'un paquet AODV contient seulement l'adresse de la destination. DSR de son coté est prévu pour des réseaux où les nœuds mobiles se déplacent avec des vitesses modérées par rapport à la latence des transmission des paquets. Enfin, ZRP divise le réseau en différentes zones. Cette approche peut être appropriée pour les topologies de grande taille. À l'intérieur des zones, le routage est proactif tandis qu'entre les zones il est réactif avec beaucoup de similarités avec AODV et DSR.

La table 3.1 résume les principales caractéristiques des protocoles présentés dans ce chapitre.

	Protocoles					
Classe	Proactifs			Réactifs		Hybride
Nom	DSDV	OLSR	TBRPF	DSR	AODV	ZRP
Distribué	oui	oui	oui	oui	oui	oui
Sans boucle	oui	oui	oui	oui	oui	oui
Plusieurs routes possibles	non	non	non	oui	non	oui
Messages de contrôle périodiques	oui	oui	oui	non	non	oui
Choix des routes	Plus court chemin	Plus court chemin	Plus court chemin	Plus court chemin[1]	Plus court chemin[1]	Plus court chemin[2]
Réaction à la mobilité	Informer d'autres nœuds pour maintenir une table de routage cohérente			Utilisent des découvertes de routes		Combinaison des deux

TAB. 3.1 – Caractéristiques des protocoles de routage ad hoc

[1] Pas forcément obtenu.

[2] Oui dans la zone locale, mais pas forcément en dehors.

Chapitre 4

La mobilité dans les réseaux ad hoc

Introduction

Un protocole doit être examiné dans des conditions réalistes prenant en compte (mais pas limitées à) une zone de transmission variable en raison d'une possible présence d'obstacles, l'espace mémoire limité pour le stockage des messages, des modèles de trafic de données, et des mouvements réalistes des nœuds mobiles, c'est-à-dire, un modèle de mobilité. Il existe plusieurs modèles de mobilité qui représentent les nœuds mobiles dont les mouvements sont indépendants les uns des autres (e.g. modèles de mobilité d'entité (*entity mobility models*) et plusieurs modèles représentant les nœuds mobiles dont les mouvements dépendent les uns des autres appelés modèles de mobilité de groupe (*group mobility models*).

Ce chapitre présente les modèles de mobilité existants utilisés pour l'évaluation des performances d'un protocole de routage ad hoc. La section 4.2 présente les modèles de mobilité d'entité, tandis que la section 4.3 présente les modèles de mobilité de groupe. Tous ces modèles sont illustrés par des figures. Enfin, une discussion sur ces modèles est présentée dans la section 4.4 ainsi qu'un tableau récapitulatif des caractéristiques de ces modèles.

4.1 Modèles de mobilité

Afin de simuler complètement un nouveau protocole pour un réseau ad hoc, il est impératif d'utiliser un modèle de mobilité qui représente exactement les nœuds mobiles qui l'utiliseront par la suite. C'est seulement dans ce type de scénario qu'il est possible de déterminer si le protocole proposé sera utile quand il sera mis en œuvre. Actuellement il y a deux types de modèles de mobilité utilisés dans la simulation des réseaux (pas seulement des réseaux ad hoc) :

- modèles de traces (*trace models*),
- modèles synthétiques (*synthetic models*).

Les modèles de traces sont les modèles de mobilité qui sont observés dans les systèmes réels. Les traces fournissent des informations précises, particulièrement quand elles impliquent un grand nombre de participants pendant une période d'observation longue. Cependant, de nouveaux environnements de réseau (par exemple les réseaux ad hoc) ne sont pas facilement modélisés si les traces n'ont pas été encore créées. Dans ce type de situation, il est nécessaire d'utiliser les modèles synthétiques qui essaient de représenter les comportements des nœuds mobiles sans l'utilisation des traces.

Un modèle de mobilité doit essayer d'imiter les mouvements de vrais nœuds mobiles. Les changements de vitesse et de direction doivent se produire dans des périodes de temps raisonnables. Par exemple, nous ne voudrions pas qu'un nœud mobile se déplace sur des lignes droites à vitesse constante au cours de la simulation toute entière car les vrais nœuds mobiles ne se déplaceraient pas d'une façon si restreinte. Voici les sept modèles synthétiques de mobilité d'entité pour les réseaux ad hoc (*synthetic entity mobility models*) qui seront présentés :

- *Modèle de mobilité de promenade aléatoire* : un modèle de mobilité simple qui choisit des directions et des vitesses de façon aléatoire.
- *Modèle de mobilité de promenade aléatoire avec pause* : un modèle qui inclut des pauses entre les changements de direction et de vitesse.
- *Modèle de mobilité de direction aléatoire* : un modèle qui force les nœuds mobiles à se déplacer vers les bords du secteur de simulation avant de changer de direction et de vitesse.
- *Le modèle de mobilité dans une région de simulation illimitée* : un mo-

dèle qui convertit un secteur 2D rectangulaire de simulation en secteur de simulation en forme de tore (*tore-shaped*).
- *Le modèle de mobilité de Gauss-Markov* : un modèle qui utilise un paramètre d'accord pour changer l'aspect aléatoire dans le modèle de mobilité.
- *La version probabiliste du modèle de promenade aléatoire* : un modèle qui utilise un ensemble de probabilités pour déterminer la prochaine position d'un nœud mobile.
- *Le modèle de mobilité des sections de ville* : un secteur de simulation qui représente les rues dans une ville.

Pour simuler un réseau ad hoc, il existe aussi des modèles de mobilité de groupe (*group mobility models*) qui permettent de simuler des situations où les décisions des nœuds dépendent des autres nœuds mobiles du même groupe. Voici quelques uns de ces modèles :

- *Le modèle exponentiel aléatoire corrélé* : c'est un modèle de mobilité de groupe qui utilise une fonction mathématique pour générer les mouvements de façon corrélée.
- *Le modèle de mobilité de colonne* : c'est un modèle où l'ensemble des nœuds mobiles forment une colonne et se déplacent uniformément en avant dans une direction particulière.
- *Le modèle de mobilité de communauté nomade* : dans ce modèle, un ensemble de nœuds mobiles se déplacent ensemble d'un endroit à un autre.
- *Le modèle de mobilité de poursuite* : un modèle où un ensemble de nœuds mobiles suivent une cible particulière.
- *Le modèle de mobilité d'un groupe avec point de référence* : dans ce modèle, un groupe de nœuds suit le chemin parcouru par un centre (nœud) logique du groupe.

4.2 Les modèles de mobilité d'entité

Cette section, présente les sept modèles de mobilité d'entité qui ont été proposés pour l'évaluation des performances d'un protocole de réseau ad hoc. Les deux premiers modèles présentés, le modèle de promenade aléatoire et le modèle de mobilité de promenade aléatoire avec pause sont parmi les plus

utilisés actuellement.

4.2.1 Le modèle de mobilité de promenade aléatoire

Le modèle de mobilité de promenade aléatoire noté RWMM (*Random Walk Mobility Model*) [15] a été décrit pour la première fois mathématiquement par Einstein en 1926. Puisque beaucoup d'entités dans la nature se déplacent de manière extrêmement imprévisible, le RWMM a été développé pour imiter ce mouvement erratique. Dans ce modèle, un nœud mobile se déplace de son endroit actuel à un nouvel endroit en choisissant aléatoirement une direction et une vitesse de déplacement. Les nouvelles vitesse et direction sont toutes les deux choisies dans des intervalles prédéfinis, $[speedmin..speedmax]$ et $[0..2\pi]$ respectivement. Chaque mouvement se produit au cours d'un intervalle de temps constant t ou sur une distance constante d traversée au delà desquels une nouvelle direction et une nouvelle vitesse sont choisies. Si un nœud mobile qui se déplace selon ce modèle arrive à la frontière de la zone de simulation, il rebondit contre elle avec un angle déterminé par la direction d'incidence. Le nœud mobile continue son déplacement suivant ce nouveau chemin comme le montre la figure 4.1.

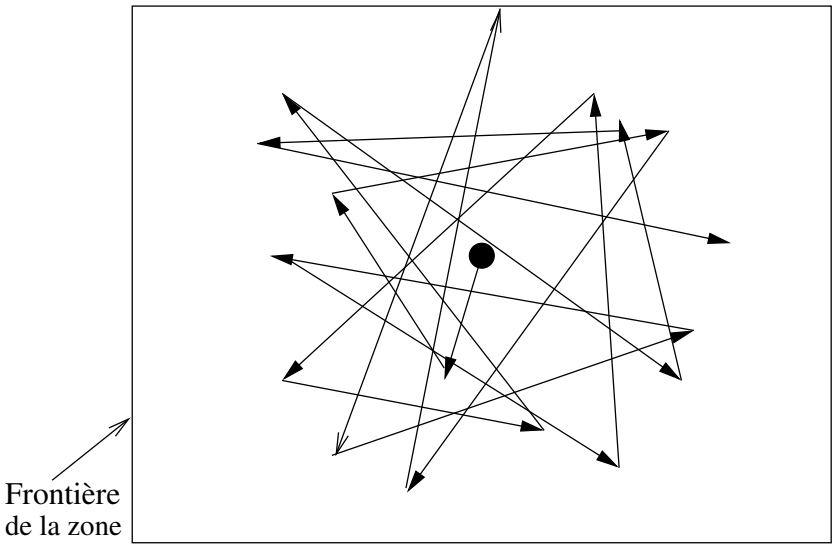


FIG. 4.1 – Modèle de mobilité de promenade aléatoire

Le modèle 2-D de RWMM est un modèle largement répandu et utilisé [30], et connu parfois sous le nom du *mouvement Brownien*. Pour son utilisation, le modèle est parfois simplifié. Ce modèle est sans mémoire car la vitesse et la direction courantes d'un nœud mobile sont indépendantes des vitesse et direction précédentes. Ceci peut produire des mouvements peu réalistes tels que des arrêts soudains et des retours à angle aigu (voir la figure 4.1). D'autres modèles, tels que le modèle de mobilité de Gauss-Markov représenté à la section 4.2.5, peuvent corriger ces anomalies.

4.2.2 Le modèle de mobilité de promenade aléatoire avec pause

Le modèle de mobilité de promenade aléatoire avec pause noté RWpMM (*Random Waypoint Mobility Model*) [15] inclut des temps de pause entre les changements de direction et/ou de vitesse [42]. Un nœud mobile commence en attendant dans un endroit pendant une certaine période de temps (c'est-à-dire, un temps de pause).

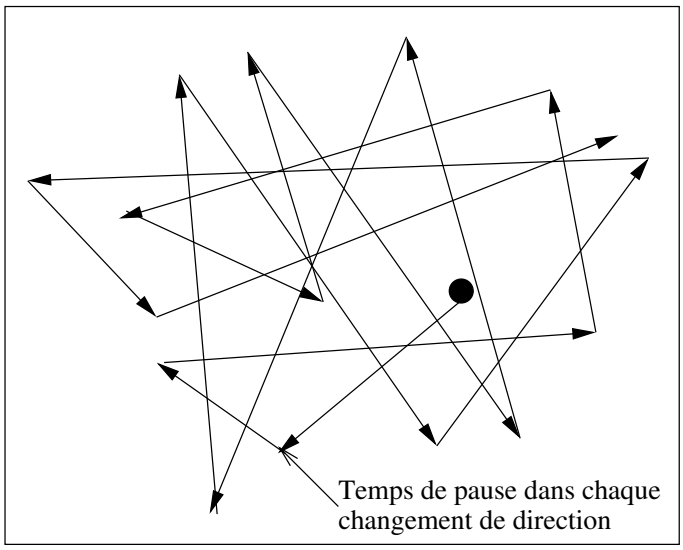


FIG. 4.2 – Modèle de mobilité de promenade aléatoire avec pause

Cette période de temps expirée, le nœud mobile choisit une destination aléatoire et une vitesse uniformément distribuée dans [*minspeed..maxspeed*].

Le nœud se déplace alors vers la nouvelle destination choisie avec la vitesse choisie. À l'arrivée, il fait une pause pendant une période de temps définie, avant de recommencer le processus.

La figure 4.2 illustre le déplacement d'un nœud mobile suivant le modèle RWpMM. Le modèle RWpMM ressemble au modèle RWMM si le temps de pause est nul et $minspeed = speedmin$ et $maxspeed = speedmax$. RWpMM est également un modèle largement répandu et souvent utilisé pour l'évaluation des performances des algorithmes et des protocoles de routage. Dans la majorité des tests de performance qui utilisent le modèle RWpMM, les nœuds mobiles sont d'abord distribués aléatoirement dans le secteur de simulation.

4.2.3 Le modèle de mobilité de direction aléatoire

Le modèle de mobilité de direction aléatoire noté RDMM (*Random Direction Mobility Model*) [15] a été créé pour surmonter le problème de *density waves* du modèle RWpMM. Une vague de densité (*density wave*) est le regroupement des nœuds dans un endroit du secteur de simulation. Dans le cas du modèle RWpMM, ceci se produit près du centre du secteur de simulation car la probabilité qu'un nœud mobile choisisse une nouvelle destination qui est située dans le centre du secteur de simulation, ou une destination qui exige le mouvement par le centre du secteur, est grande. Ainsi, les nœuds mobiles semblent converger, se disperser et converger encore.

Le modèle RDMM a été développé afin d'éviter ce type de comportement et favoriser un nombre semi-constant de voisins pendant toute la simulation. Dans ce modèle, les nœuds mobiles choisissent une direction aléatoire et ils se déplacent comme dans le modèle RWMM vers la frontière du secteur de simulation dans cette direction. Une fois que la frontière est atteinte, le nœud fait une pause pendant un temps prédéfini, choisit une autre direction dans un angle (entre 0 et 360 degrés) et continue le processus.

La figure 4.3 montre un exemple de déplacement d'un nœud mobile qui commence au centre du secteur de simulation, en utilisant le modèle RDMM. Les points dans la figure illustrent l'arrivée du nœud à la frontière où, après avoir fait une pause, et il choisit une nouvelle direction.

Il existe aussi une version modifiée du modèle RDMM dans laquelle les nœuds continuent à choisir des directions aléatoires mais elles ne sont plus forcées de se déplacer vers la frontière du secteur de simulation avant de s'arrêter et changer de direction. Pour éviter cela, le nœud choisit une direction aléatoire et un endroit (ou point) quelque part dans cette direction du dé-

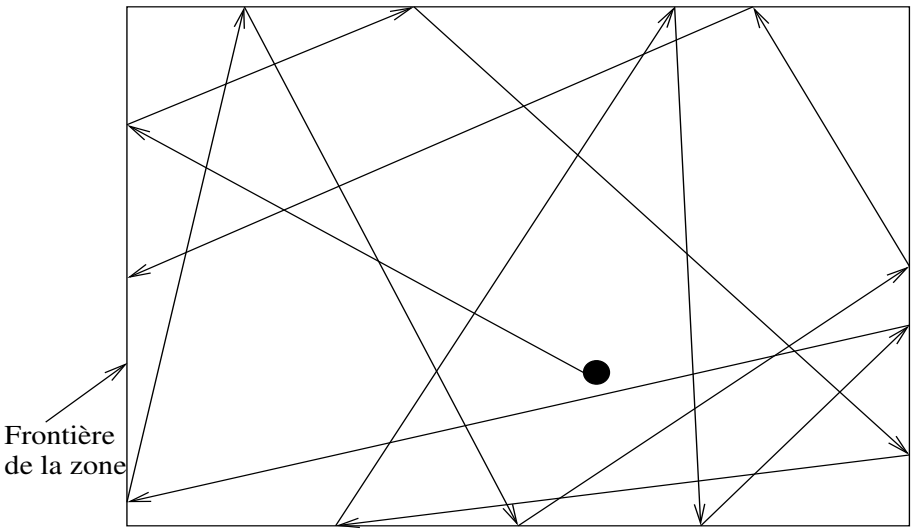


FIG. 4.3 – Modèle de mobilité de direction aléatoire

placement. Le nœud fait alors une pause à cet endroit avant de choisir une nouvelle direction aléatoire. Cette modification du modèle RDMM produit des modèles de mouvement qui pourraient être simulés par le modèle RWMM en y ajoutant des temps de pause.

4.2.4 Le modèle de mobilité dans une région de simulation illimitée

Pour le modèle de mobilité dans une région de simulation illimitée noté BSAMM (*Boundless Simulation Area Mobility Model*) [15], la direction et la vitesse courante du mouvement dépendent des direction et vitesse précédentes du nœud mobile. Un vecteur de vitesse $\overline{v} = (v, \theta)$ est utilisé pour décrire le déplacement de vitesse v et de direction θ. La position du nœud est représentée comme (x, y). Le vecteur de vitesse et la position sont mis à jour après chaque Δt unité de temps selon les formules suivantes :

$$
\begin{aligned}
v(t+\Delta t) &= min[max(v(t)+\Delta v, 0), V_{max}] \\
\theta(t+\Delta t) &= \theta(t)+\Delta\theta \\
x(t+\Delta t) &= x(t)+v(t)\cos\theta(t) \\
y(t+\Delta t) &= y(t)+v(t)\sin\theta(t)
\end{aligned}
$$

où V_{max} est la vitesse maximum définie dans la simulation, Δv est le changement de la vitesse qui est uniformément distribuée entre $[-A_{max} * \Delta t, A_{max} * \Delta t]$, A_{max} est l'accélération maximum d'un nœud mobile donné, $\Delta\theta$ est le changement de la direction qui est uniformément distribuée entre $[-\alpha * \Delta t, \alpha * \Delta t]$, et α est le changement angulaire maximum dans lequel le nœud se déplace.

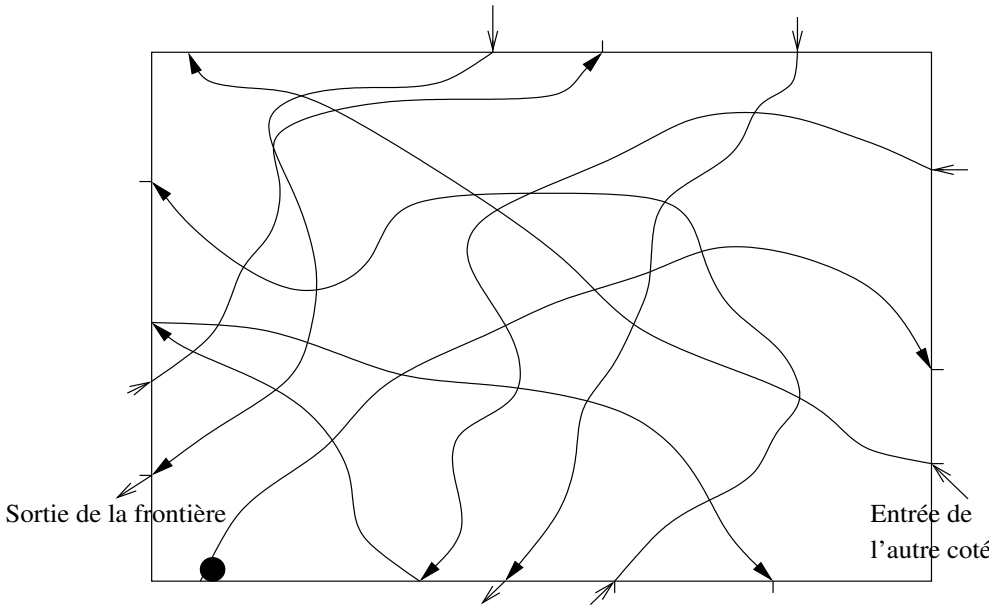


FIG. 4.4 – Modèle de mobilité dans une région de simulation illimité

Le modèle BSAMM est différent des modèles précédents dans la façon dont la frontière du secteur de simulation est prise en compte (voir la figure 4.4). Dans tous les modèles de mobilité mentionnés précédemment les nœuds « se reflètent » ou arrêtent de se déplacer une fois qu'ils atteignent la frontière. Dans le modèle BSAMM les nœuds qui atteignent la frontière continuent de se déplacer et réapparaissent du côté opposé du secteur de simulation d'où l'impression que le secteur de simulation est illimité. Ce modèle peut être utilisé pour simuler un réseau de grande taille de type WMAN ou WWAN.

4.2.5 Le modèle de Gauss-Markov

Le modèle de mobilité de Gauss-Markov [15] a été proposé pour la simulation d'un réseau PCS (*Personnal Communications Services*)[45], mais il a aussi été utilisé pour la simulation d'un protocole de routage ad hoc [65]. Ce modèle a été conçu pour s'adapter aux différents niveaux d'aspect aléatoire, par l'intermédiaire d'un paramètre d'accord (*tuning parameter*). Les mouvements générés par ce modèle évitent les arrêts soudains tout en étant proche de la réalité.

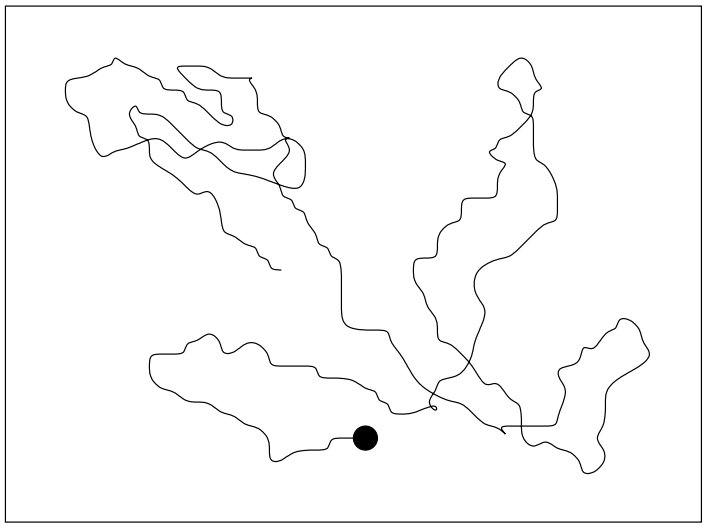


FIG. 4.5 – Le modèle de Gauss-Markov

Initialement, une vitesse et une direction sont assignées à chaque nœud mobile. À intervalle de temps fixe, un mouvement se produit qui met à jour la vitesse et la direction de chaque nœud. Plus précisément, la valeur de la vitesse et la direction au n-ième intervalle de temps est calculée en fonction des valeurs de la vitesse et de la direction du $(n-1)$-ème intervalle de temps et d'une variable aléatoire, en utilisant les équations suivantes :

$$s_n = \alpha s_{n-1} + (1-\alpha)\overline{s} + \sqrt{(1-\alpha^2)s_{x_{n-1}}}$$
$$d_n = \alpha d_{n-1} + (1-\alpha)\overline{d} + \sqrt{(1-\alpha^2)d_{x_{n-1}}}$$

où s_n et d_n sont les nouvelles vitesse et direction du nœud mobile à l'instant n ; α, $0 \leq \alpha \leq 1$, est le paramètre d'accord utilisé pour l'aspect aléatoire ; $\overline{s}$

et $\overline{d}$ sont des constantes représentant la valeur moyenne de la vitesse et de la direction quand $n \to \infty$; et $s_{x_{n-1}}$ $d_{x_{n-1}}$ sont des variables aléatoires d'une distribution de Gauss. À chaque intervalle de temps, la prochaine destination est calculée en utilisant la destination, la vitesse et la direction courantes du mouvement. Plus précisément, pendant un intervalle de temps n, la position d'un nœud mobile est donnée par les équations suivantes :

$$x_n = x_{n-1} + s_{n-1} \cos d_{n-1}$$
$$y_n = y_{n-1} + s_{n-1} \sin d_{n-1}$$

où (x_n, y_n) et (x_{n-1}, y_{n-1}) sont respectivement les coordonnées x et y de la position du nœud aux n-ième et $(n-1)$-ème intervalles, et s_{n-1}, d_{n-1} sont la vitesse et la direction du nœud mobile pendant le $(n-1)$-ème intervalle.

4.2.6 La version probabiliste du modèle de promenade aléatoire

La version probabiliste du modèle de promenade aléatoire, noté PRWMM (*Probabilistic Version of Random Walk Mobility Model*) [15], utilise une matrice de probabilités pour déterminer la position d'un nœud mobile à la prochaine itération, qui est représentée par trois différents états pour la position x et trois différents états pour la position y. L'état 0 représente la position courante (x ou y) pour un nœud mobile donné, l'état 1 représente la position précédente (x ou y) et l'état 2 représente la prochaine position du nœud s'il continue à se déplacer dans la même direction.

La matrice de probabilités utilisée est :

$$\mathbf{P} = \begin{pmatrix} P(0,0) & P(0,1) & P(0,2) \\ P(1,0) & P(1,1) & P(1,2) \\ P(2,0) & P(2,1) & P(2,2) \end{pmatrix}$$

où chaque élément $P(a,b)$ représente la probabilité qu'un nœud mobile passe de l'état a à l'état b. Les valeurs de cette matrice sont utilisées pour la mise à jour de l'ensemble des positions x et y du nœud. La figure 4.6 donne un exemple de trajectoire du déplacement d'un nœud mobile utilisant ce modèle. Ce modèle est peu réaliste par rapport aux modèles RWpMM ou RWMM car les mouvements produits sont probabilistes et pas aléatoires comme c'est plutôt le cas dans la pratique.

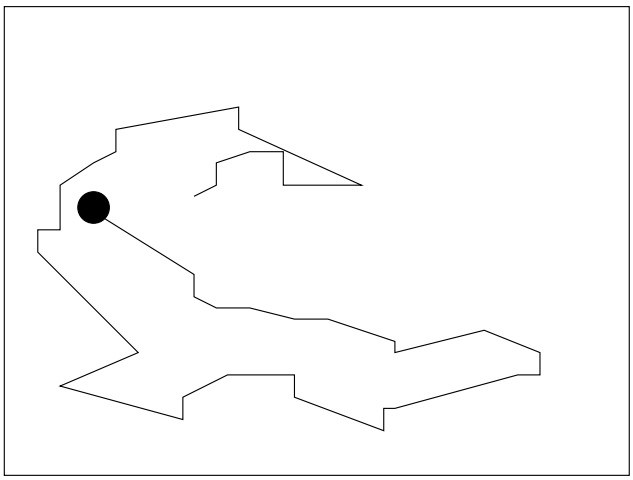


FIG. 4.6 – Version probabiliste du modèle de promenade aléatoire

4.2.7 Le modèle de mobilité des sections de ville

Dans le modèle de mobilité des sections de ville noté CSMM (*City section mobility model*) [15] le secteur de simulation est un réseau routier qui représente une section d'une ville où on veut établir un réseau ad hoc. Les rues et les limites de vitesse sont dépendantes de la ville que l'on simule. Par exemple, les rues peuvent former une grille dans le centre ville avec une route à grande vitesse près de la frontière du secteur de simulation pour représenter une boucle autour de la ville. Chaque nœud commence la simulation à un point défini sur une certaine rue et choisit alors aléatoirement une destination, également représentée par un point sur une rue quelconque (voir figure 4.7).

L'algorithme du mouvement, de la destination courante à la nouvelle destination, localise un chemin correspondant au temps de voyage le plus court entre les deux points. En outre, on peut considérer d'autres caractéristiques telles qu'une limite de vitesse et une distance minimum entre deux nœuds. Après l'arrivée à la destination, le nœud mobile fait une pause pendant une période de temps définie, puis il choisit aléatoirement une autre destination et répète le processus.

Ce modèle fournit des mouvements réalistes pour une partie d'une ville puisqu'il limite fortement le comportement des déplacements des nœuds mobiles. Dans la pratique, les nœuds n'ont pas la capacité de se déplacer libre-

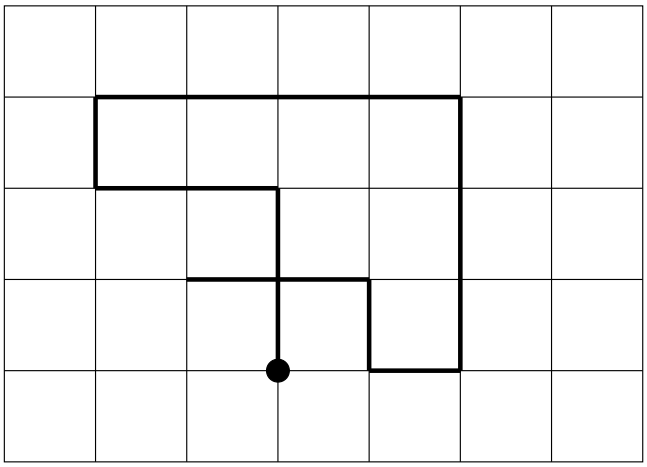


FIG. 4.7 – Modèle de mobilité des sections de ville

ment sans considérer les obstacles et les règlements du trafic. En outre, les gens tendent à déplacer dans des modèles semblables en conduisant à travers la ville ou en marchant à travers un campus. Le fait d'obliger tout nœud mobile à suivre des chemins prédéfinis augmentera la moyenne du nombre de sauts dans les simulations.

4.3 Modèles de mobilité de groupe

La section 4.2, présente les modèles de mobilité qui simulent les nœuds dont les comportements sont complètement indépendants. Cependant, dans un réseau ad hoc, il y a beaucoup de situations où il est nécessaire de modéliser le comportement des nœuds quand ils se déplacent ensemble. Par exemple, dans un scénario militaire, un groupe de soldats peut effectuer une mission dans une partie de terrain afin de détruire des mines antipersonnel, capturer les assaillants ennemis, ou travailler simplement ensemble d'une façon coopérative pour accomplir une tâche commune. Afin de modéliser de telles situations, un modèle de mobilité de groupe est nécessaire pour simuler cette caractéristique coopérative.

Cette section présente cinq modèles de mobilité de groupe. Quatre d'entre eux, dont le plus général est le modèle de la mobilité de groupe avec point de référence (RPGM), sont étroitement liés. En effet, les modèles de mobilité

de colonne, de communauté nomade et poursuite peuvent être mis en œuvre en tant que cas spéciaux du modèle RPGM.

4.3.1 Le modèle exponentiel aléatoire corrélé

Un des premiers modèles de mobilité de groupe proposé est le modèle exponentiel aléatoire corrélé, noté ECRMM (*Exponential Correlated Random Mobility Model*) [15]. Ce modèle utilise une fonction de mouvement pour créer les mouvements du nœud mobile. Étant donnée une position du nœud ou du groupe au temps t, le vecteur $\overrightarrow{b}(t)$ est utilisé pour définir la prochaine position du nœud ou du groupe au temps $t+1$, $\overrightarrow{b}(t+1)$:

$$\overrightarrow{b}(t+1) = \overrightarrow{b}(t)e^{-\frac{1}{\tau}} + (\sigma\sqrt{1-(e^{-\frac{1}{\tau}})^2})r$$

où τ ajuste le taux de changement de l'endroit précédent du nœud à son nouvel endroit. Un τ petit signifie un grand changement, tandis que r est une variable gaussienne aléatoire avec variance σ.

4.3.2 Modèle de mobilité de colonne

Le modèle de mobilité de colonne noté CMM (*Column Mobility Model*) [15] s'avère utile pour modéliser une recherche sur le terrain. Ce modèle représente un ensemble de nœuds mobiles qui se déplacent globalement en parallèle en suivant un axe donné (par exemple, une ligne de soldats marchant ensemble vers leur ennemi ou des opérations de recherche pour désactiver des mines antipersonnel). Une légère modification du modèle CMM permet aux nœuds mobiles de se suivre (par exemple, un groupe marchant en file indienne). Pour l'implantation de ce modèle, une première grille (*grid*) de référence (formant une colonne de nœuds mobiles) est définie [58].

Ensuite, chaque nœud (voir figure 4.8) est placé par rapport à son point de référence dans la grille de référence ; on permet alors au nœud mobile de se déplacer aléatoirement autour de son point de référence utilisant un des modèles de mobilité d'entité représentés dans la section 4.2 (dans la littérature, on propose d'utiliser le modèle de promenade aléatoire RWMM présenté dans la section 4.2.1). Le nouveau point de référence pour un nœud mobile est défini comme :

$$new\_référence\_point = old\_référence\_point + advance\_vector$$

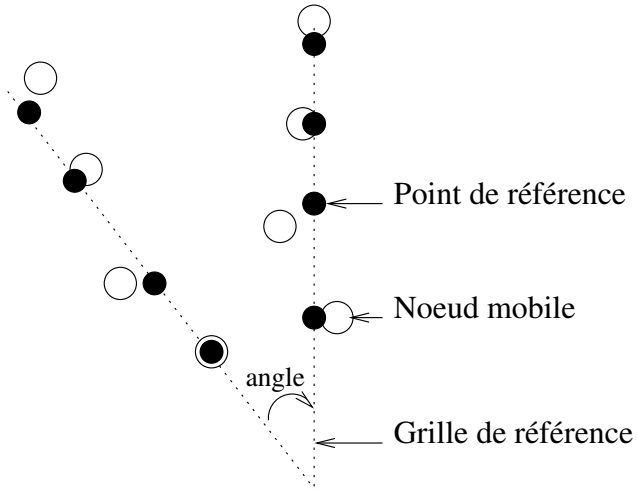


FIG. 4.8 – Modèle de mobilité de colonne

où *old_ référence_ point* est l'ancien point de référence et *advance_ vector* est un décalage (*offset*) prédéfini qui déplace la grille de référence. Le décalage prédéfini est calculé par l'intermédiaire d'une distance aléatoire et d'un angle aléatoire entre 0 et π puisque le mouvement se fait seulement dans l'axe de progression. Puisque le même décalage est utilisé pour chaque nœud mobile, la grille de référence est une ligne droite.

4.3.3 Le modèle de mobilité de communauté nomade (NCMM)

Comme les nomades qui se déplacent d'un endroit à un autre, le modèle de la communauté nomade NCMM (*Nomadic Community Mobility Model*) [15, 58] représente des groupes de nœud mobiles qui se déplacent ensemble d'un point à l'autre. Dans chaque communauté, les individus maintiennent leur propre espace personnel où ils se déplacent de manière aléatoire. De nombreuses applications existent pour ce type de scénario, par exemple, un groupe d'individus en train de visiter un musée.

Dans le modèle NCMM, chaque nœud utilise un des modèles de mobilité d'entité présentés dans la section 4.2, pour errer autour d'un point de référence donné. Quand le point de référence change, tous les nœuds mobiles dans le groupe se déplacent vers le nouveau secteur défini par le point de

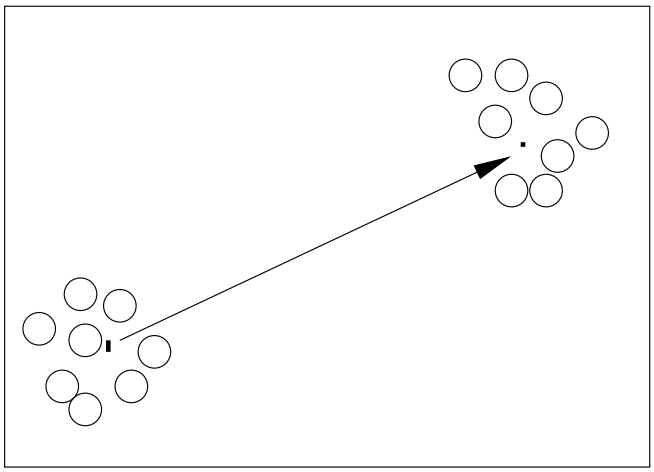


FIG. 4.9 – Modèle de mobilité de communauté nomade

référence et commencent à errer autour du nouveau point de référence (figure 4.9). Les paramètres pour le modèle de mobilité d'entité définissent à quelle distance un nœud mobile peut errer autour du point de référence.

Comparé au modèle de mobilité de colonne CMM où chaque nœud a un point de référence individuel, les nœuds dans le modèle de mobilité nomade NCMM partagent un point de référence commun.

4.3.4 Le modèle de mobilité de poursuite

Le modèle de mobilité de poursuite, noté PMM (*Pursue Mobility Model*) [58], essaie de représenter, comme son nom l'indique, des nœuds mobiles qui suivent une cible particulière.

Ce modèle consiste en une équation de mise à jour simple pour la nouvelle position de chaque nœud :

$$new\_position = old\_position + acceleration(target - old\_position) + random\_vector$$

où *acceleration(target-old_position)* est l'information sur le mouvement du nœud poursuivi, et *random_vector* est un décalage (*offset*) aléatoire pour chaque nœud. La valeur de *random_vector* est obtenue par un modèle de mobilité d'entité et l'aspect aléatoire des nœuds est limité afin de maintenir un chemin efficace du nœud poursuivi. La figure 4.10 illustre le déplacement

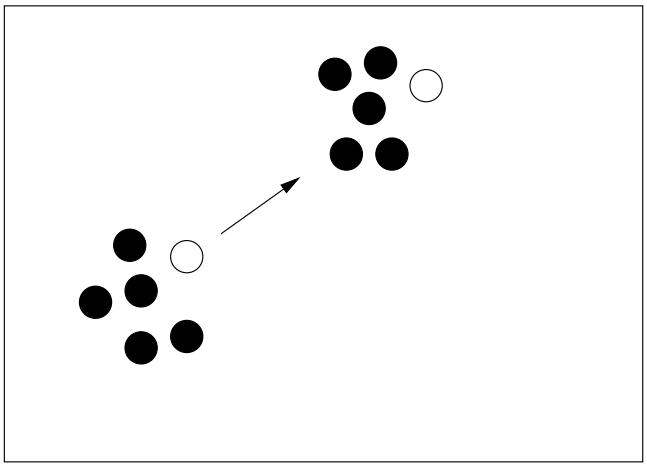


FIG. 4.10 – Le modèle de mobilité de poursuite

d'un groupe de nœuds (en noir dans la figure) qui poursuivent le nœud en blanc. Ce modèles est très réaliste et largement répandu dans la pratique.

4.3.5 Le modèle de mobilité d'un groupe avec point de référence

Le modèle de mobilité d'un groupe avec point de référence noté RPGM (*Reference Point Group Mobility Model*) [15, 58] représente le mouvement aléatoire d'un groupe de nœuds aussi bien que le mouvement aléatoire de chaque nœud individuellement dans le groupe [39]. Les mouvements du groupe sont basés sur le chemin parcouru par un centre du groupe qui est utilisé pour calculer le mouvement du groupe par l'intermédiaire d'un vecteur de mouvement $\overrightarrow{GM}$ qui peut être choisi aléatoirement ou être prédéfini.

Le mouvement du centre du groupe caractérise complètement le mouvement de son groupe (la direction et la vitesse) . Les nœuds individuels se déplacent aléatoirement par rapport à leurs propres points de référence prédéfinis, dont les mouvements dépendent du mouvement du groupe.

La figure 4.11 illustre le mouvement de 3 nœuds utilisant le modèle RPGM. Ainsi, à l'instant t, les 3 points noirs représentent les points de référence, $RP(t)$, pour les trois nœuds. Le modèle RPGM utilise le vecteur $\overrightarrow{GM}$ pour calculer les nouveaux points de référence, $RP(t+1)$, à l'instant

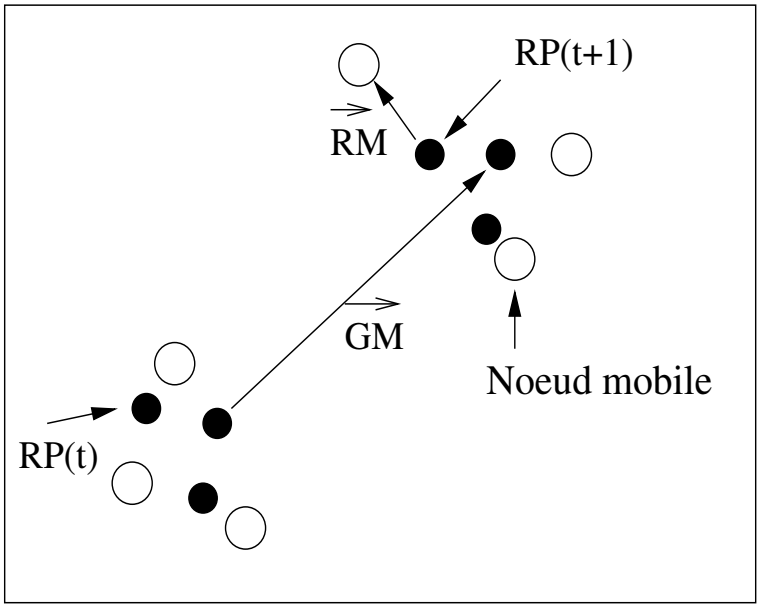


FIG. 4.11 – Mouvements de 3 nœuds utilisant le modèle RPGM

$t+1$. La nouvelle position de chaque nœud est alors calculée par la somme vectorielle du vecteur $\overrightarrow{GM}$ avec un vecteur aléatoire de mouvement, $\overrightarrow{RM}$. La longueur de $\overrightarrow{RM}$ est uniformément distribuée dans un disque qui a pour centre $RP(t+1)$, tandis que sa direction est uniformément distribuée dans $[0..2\pi]$.

Le modèle RPGM a été conçu pour faire face à des scénarios tels qu'une avalanche après laquelle une équipe de secours se composant d'humains et de chiens, travaillent coopérativement. Les guides humains (centres des groupes) tendent à définir un chemin général puisqu'ils connaissent habituellement l'endroit approximatif des victimes. Chacun des chiens crée son propre chemin aléatoire autour du secteur général choisi par leurs guides humains.

4.3.6 Le modèle de mobilité avec obstacles

Le modèle de mobilité avec obstacles [41] a été conçu pour modéliser le mouvement des nœuds mobiles dans les terrains qui ressemblent à des topographies réelles. Des objets modélisent les bâtiments et d'autres structures qui empêchent les mouvements des nœuds, ainsi que leur transmission sans

fil. En modélisant un tel terrain, un utilisateur peut définir les positions, les formes et les tailles de ces objets. Ce modèle peut manipuler des formes et des positions arbitraires pour les objets, permettant de modéliser beaucoup de terrains réels.

Le deuxième composant du modèle de mobilité est un graphe de mouvement qui représente les déplacements des nœuds. Ce graphe planaire est le diagramme de Voronoï des coins des obstacles (les arêtes sont des segments qui sont à une distance égale des deux coins d'un obstacle). Ainsi, ce modèle est basé sur l'intuition que les chemins tendent à se définir à mi-distance entre les bâtiments adjacents. Ce modèle permet également le mouvement par l'intérieur des bâtiments.

Le troisième composant du modèle est le choix des routes. Les nœuds utilisent les chemins les plus courts (en terme de longueur euclidienne) pour se déplacer entre deux points du graphe du mouvement, c'est-à-dire qu'ils suivent le chemin le plus court dans le diagramme de Voronoï. Le placement des objets et des chemins les reliant sont calculés au début de la simulation et ne changent pas pendant toute la simulation. Les nœuds sont distribués au hasard le long des chemins, ils choisissent une destination et puis se déplacent vers cette destination en suivant le chemin le plus court à partir de sa position courante.

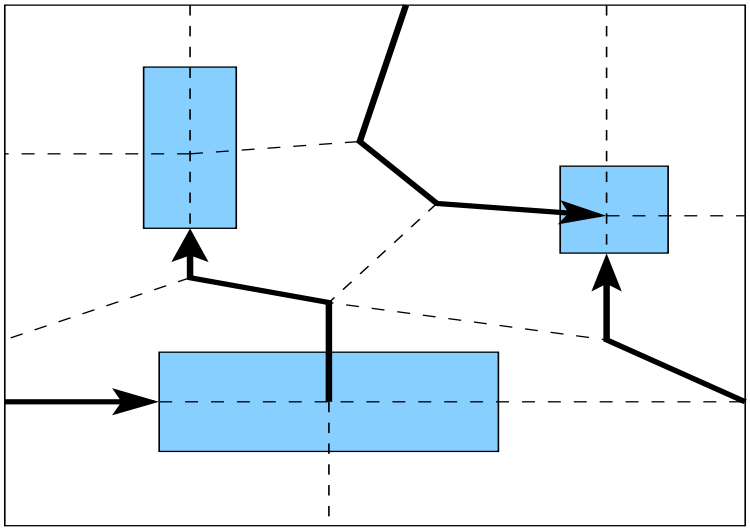


FIG. 4.12 – Mouvements avec obstacles utilisant le diagramme de Voronoï

Ainsi, chaque nœud calcule le chemin le plus court sur le graphe créé par les chemins et puis se déplace vers cette destination en utilisant le chemin calculé. Lorsqu'il atteint sa destination, le nœud fait une pause pour une certaine période de temps. Il choisit alors une nouvelle destination, calcule le chemin le plus court pour l'atteindre, et reprend le mouvement. Les nœuds peuvent se déplacer à l'intérieur des bâtiments car le plus court chemin entre deux endroits peut exiger le passage par l'intérieur d'un bâtiment.

4.4 Discussion sur les modèles de mobilité

La performance d'un protocole de réseau ad hoc peut changer de manière significative quand il est testé avec différents modèles de mobilité, mais aussi quand le même modèle de mobilité est utilisé avec différents paramètres. De plus, le choix d'un modèle exige un modèle de trafic de données qui influence aussi la performance du protocole. Par exemple, quand on simule un modèle de mobilité de groupe, l'évaluation du protocole devra prendre en compte l'aspect local d'une partie du trafic à l'intérieur du groupe. La performance d'un protocole de réseau ad hoc devrait être évaluée avec le modèle de mobilité qui est le plus proche du scénario réel prévu, ce qui peut faciliter le développement du protocole de réseau ad hoc.

Le modèle de mobilité de promenade aléatoire, RWMM, avec un petit paramètre d'entrée (distance ou temps) produit un mouvement Brownien et, en conséquence, évalue fondamentalement un réseau statique quand il est utilisé pour l'évaluation de la performance. Par contre, avec l'utilisation d'un grand paramètre d'entrée, le modèle RWMM ressemble au modèle de mobilité RWpMM si on y ajoute des temps de pause. La différence principale entre ces deux modèles est que les nœuds simulant le modèle RWpMM ont plus tendance à se grouper au centre du secteur de simulation, que les nœuds simulant le modèle RWMM. Le modèle RWpMM est utilisé dans beaucoup d'études de protocoles de réseau ad hoc. Il est flexible, et il semble créer des modèles de mobilité réalistes. Un inconvénient de ce modèle est la ligne droite du mouvement suivi par le nœud mobile qui se déplace vers la prochaine destination choisie.

Le modèle de mobilité de direction aléatoire, RDMM, est un modèle peu réaliste parce qu'il est peu probable que les dispositifs se disperseraient aléatoirement dans tout le secteur (un bâtiment ou une ville). En outre, il est peu probable qu'ils feront une pause seulement au bord de la frontière d'un

secteur. Le modèle de direction aléatoire modifié permet aux nœuds de faire une pause et de changer de direction avant d'atteindre la frontière du secteur de simulation. Cependant, cette version est identique au modèle de mobilité de promenade aléatoire, RWMM, en y ajoutant des temps de pause.

Le modèle de mobilité dans une région de simulation illimitée, BSAMM, fournit des modèles de mouvement auxquels on pourrait s'attendre dans la réalité. En outre, ce modèle est le seul qui permet aux nœuds mobiles de se disperser dans le secteur, en éliminant les effets de la frontière sur l'évaluation des performances. Cependant, les effets secondaires qui se produiraient en permettant aux nœuds mobiles de se déplacer autour d'un tore (*torus*) pourraient être considérables. Par exemple, un nœud mobile qui se déplace dans la même direction vers un nœud statique deviendra son voisin à plusieurs reprises.

Le modèle de mobilité Gauss-Markov fournit des modèles de mouvement auxquels on pourrait s'attendre dans la réalité si des paramètres appropriés sont choisis. En outre, la méthode utilisée pour forcer les nœuds à partir loin des bords du secteur de simulation (évitant ainsi les effets du bord de secteur) est intéressante.

Même si le modèle probabiliste de promenade aléatoire, PRWMM, fournit des mouvements auxquels on pourrait s'attendre dans la réalité, choisir des paramètres appropriés pour sa matrice de probabilité peut être difficile.

Le modèle de mobilité des sections de ville, CSMM, semble créer des mouvements réalistes pour une section d'une ville, puisqu'il limite sévèrement le comportement des déplacements des nœud mobiles. Ces nœuds n'ont pas la capacité d'errer librement sans se soucier des obstacles et d'autres règlements de trafic.

Concernant les cinq modèles synthétiques de mobilité de groupe pour les réseaux ad hoc, on pourrait dire que le modèle exponentiel aléatoire corrélé, ECRMM, semble décrire théoriquement tout autre modèle de mobilité. Cependant, le choix des valeurs appropriées pour les paramètres τ et σ (voir section 4.3.1) est (presque) impossible. Les modèles de colonne, de la communauté nomade et de poursuite sont des modèles utiles pour des scénarios réalistes spécifiques. Les modèles de mouvement qu'ils fournissent peuvent être obtenus en changeant les paramètres liés au modèle d'un groupe avec point de référence (RPGM). Enfin, un modèle de mobilité d'entité doit être conçu non seulement pour manipuler le mouvement d'un groupe de nœuds mobiles, mais aussi le mouvement des nœuds individuellement à l'intérieur du groupe.

La table 4.1 présente de manière compacte les principales caractéristiques, comme la vitesse, la direction et la nature (réaliste ou pas) des modèles de mobilité.

Ce que tous ces modèles de mobilité ont en commun est que les modèles qu'ils créent ne sont pas nécessairement comparables aux mouvements dans la réalité. En particulier, les gens sur des campus universitaires ou dans des centres commerciaux ne se déplacent généralement pas en directions aléatoires. Ils tendent à choisir une destination spécifique et à suivre un chemin bien défini pour atteindre cette destination. Le choix du chemin est influencé par les voies et les obstacles présents. Par exemple, sur un campus universitaire, les individus marchent généralement sur les chemins qui sont faits pour relier ensemble les bâtiments du campus, tandis que certains individus peuvent emprunter des chemins à travers les pelouses. En plus, les destinations ne sont typiquement pas aléatoires, mais sont des bâtiments, des bancs dans un parc, ou d'autres endroits spécifiques dans le campus.

Modéles de mobilité	Entité								Groupe				
Nom	RWMM	RWpMM	RDMM	BSAMM	Gauss-Markov	PRWMM	CSMM	Obstacles	ECRMM	CMM	NCMM	PMM	RPGMM
Vitesse aléatoire	oui	oui	oui	oui	non	oui	oui	oui	oui	oui	oui	oui	oui
Direction aléatoire	oui	oui	oui	oui	non	non	non	non	non	non	non	non	oui
Réaliste	oui	oui	peu	oui	peu	peu	oui	oui	non	oui	oui	oui	oui
Type d'application	Mouvement Brownien	WLAN	WLAN WMAN	WMAN WWAN	PCS	WLAN	Régulation du trafic	WLAN	-	Recherche	Visite touristique	Poursuite d'une cible	Situations de secours

TAB. 4.1 – Caractéristiques des modèles de mobilité

Chapitre 5

Routage et mobilité

Introduction

Le chapitre 3 a présenté le routage dans les réseaux ad hoc en décrivant les principaux protocoles de routage proposés. Beaucoup de caractéristiques sont déjà présentes ou peuvent être ajoutées pour faciliter le passage à l'échelle (*scalability*) de ces protocoles. Par exemple, R-DSDV (*Randomized-DSDV*) [9] ajoute un contrôle de congestion au protocole DSDV original, afin de limiter le surcoût du contrôle (*control overhead*). OLSR [18] réduit le nombre de paquets de contrôle en choisissant seulement une partie des nœuds voisins pour la diffusion de ses paquets (*packet broadcasting*). TBRPF [50] utilise un routage à états de liens, maintient et partage un arbre fournissant des chemins vers tous les nœuds accessibles, aussi utilisé pour les mises à jour par diffusion. Toutes ces approches fournissent des améliorations pour le passage à l'échelle des protocoles de routage eux-mêmes, mais des problèmes de performances demeurent, comme par exemple des routes avec beaucoup de sauts ou un nombre important de messages de contrôle.

Le chapitre 4 a présenté les modèles de mobilité utilisés pour l'évaluation des protocoles de routage. Cette évaluation doit être faite avec le modèle le plus proche du scénario réel prévu. Toutefois, il n'y a pas en général de relation directe entre les travaux sur le routage et ceux sur la mobilité. Le routage ne se sert pas de la mobilité des nœuds (il la subit) et la mobilité n'est pas contrainte par le routage. Pourtant, dans un réseau ad hoc, la mobilité et le routage sont très liés. Par exemple, les nœuds s'éloignent souvent en créant des partitions du réseau, rendant les algorithmes de routage inopérants. La

communication par sauts devient alors impossible puisque les nœuds voisins sont trop éloignés. La mobilité est considérée comme un facteur qui pose des problèmes aux protocoles de routage.

En fait, la recherche sur les algorithmes de routage dans les réseaux ad hoc s'est principalement concentrée sur les algorithmes dédiés à des réseaux totalement connectés dans lesquels la mobilité des nœuds n'est pas utilisée. Pourtant, si on peut contrôler la mobilité des nœuds, elle peut devenir un facteur positif pour améliorer les performances des protocoles de routage. Par exemple, au lieu d'attendre que le réseau devienne connecté, un nœud peut changer sa trajectoire en fonction de sa connaissance sur les trajectoires de ses voisins pour reconnecter le réseau.

Dans cette thèse, nous nous sommes intéressés à des techniques qui mettent la mobilité au service du routage. Ainsi notre étude s'intéresse à une architecture de contrôle de la mobilité des nœuds pour améliorer les performances des protocoles de routage d'un réseau ad hoc.

Ce chapitre présente quelques travaux proches de notre travail ainsi qu'une introduction de notre architecture du contrôle de la mobilité. Comme nous allons le voir, l'idée de contrôler les mouvements des nœuds afin d'améliorer les performances est partagée par plusieurs groupes de recherche. Ce chapitre sert aussi de lien logique entre, d'une part les chapitres 3 et 4, sur le routage et la mobilité dans les réseaux ad hoc et d'autre part, notre travail sur les améliorations que le contrôle de la mobilité peut amener au routage. Dans un réseau mobile ad hoc, la mobilité est d'une importance essentielle et nous pensons qu'elle peut s'avérer très utile afin d'améliorer les performances des protocoles de routage.

5.1 Changement des trajectoires

Un sujet principal dans l'utilisation de la mobilité est : « comment la contrôler efficacement ». Concevoir des algorithmes de contrôle de la mobilité pour améliorer la communication est un défi, car chaque solution proposée doit aborder les problématiques suivantes :

- d'abord, la nature du contrôle effectif de mobilité sera dépendante des applications. Les nœuds vont se déplacer différemment sous différents modèles de trafic, par exemple si l'on considère les cas suivants : une seule paire de source-destination (dite *single flow*), ou plusieurs paire

source-destination (dite *multiflows*), ou plusieurs sources et une destination (dite *concast*). Y a-t-il un seul schéma couvrant tous ces scénarios ?
- en ce qui concerne le passage à l'échelle et la robustesse, il ne doit pas y avoir une entité centrale qui calcule les mouvements de tous les nœuds, c'est-à-dire que le schéma doit être distribué.
- ce schéma distribué doit être capable d'organiser les nœuds afin d'optimiser les performances tout en satisfaisant en même temps d'autres contraintes. Par exemple, si l'objectif d'un schéma de contrôle de mobilité est d'optimiser le flux de l'information sur l'énergie des nœuds, il peut être important que le schéma essaie aussi d'assurer la connectivité du réseau.

Mais, même si la mobilité a un potentiel pour améliorer les performances du réseau, il peut y avoir des scénarios où le contrôle de la mobilité sera moins utile à cause de divers facteurs atténuants comme les limitations du matériel ou des modèles de trafic.

5.1.1 Approche de Li et Rus

Dans leur travail [44], Li et Rus proposent un schéma de contrôle des mouvements qui garantit une transmission des messages dans un temps minimum. Leur idée consiste à forcer le changement de trajectoire des nœuds dans un réseau ad hoc afin de transmettre des messages. Ils supposent un scénario où un ensemble de nœuds se déplacent selon des trajectoires prédéterminées. La vitesse maximale des nœuds est assez grande par rapport à la distance entre les nœuds. Dans ces situations, le temps pour un nœud d'approcher d'autres nœuds est très court et n'affecte pas l'estimation du déplacement des autres nœuds. Autrement dit, le temps pendant lequel un nœud dévie de sa trajectoire originale est négligeable. Li et Rus ont proposé deux algorithmes qui minimisent les modifications des trajectoires en supposant respectivement que :

- les mouvements de tous les nœuds dans le système sont connus,
- les mouvements des nœuds dans le système ne sont pas connus.

Le premier algorithme suppose que chaque nœud connaît les mouvements et les positions de tous les autres nœuds dans le réseau. Cette restriction est

très forte mais peut avoir des applications pratiques comme le mouvement dans les rues d'une ville (voir section 4.2.7 dans le chapitre 4). Chaque nœud a une tâche à accomplir qui peut inclure le mouvement et le traitement de l'information qu'il doit envoyer aux autres. L'algorithme pour un nœud h_i du système est présenté dans la figure 5.1.

- accomplir différentes tâches si nécessaire, en attendant de recevoir des messages ;
- générer un message quand on veut communiquer ;
if *un message m_j est reçu* ***then***
  if *la destination de m_j est h_i* ***then***
    traiter m_j
  else
    // le nœud h_i n'est pas la destination du message
    if *la destination de m_j est h_k* ***then***
      - calculer le chemin optimal (h_i,h_k), étant donné une table de routage (host,path-to-reach-host) ;
      - envoyer le message au prochain nœud en changeant la trajectoire si nécessaire pour se trouver dans la zone de transmission du prochain nœud, suivi par un retour à la trajectoire originale ;
    end if
  end if
end if

FIG. 5.1 – L'algorithme du comportement d'un nœud h_i

Deux nœuds peuvent communiquer s'il se trouvent dans la zone de transmission l'un de l'autre, c'est-à-dire à une distance $\leq r$. Mais si un nœud n_1 veut envoyer un message au nœud n_4 (voir la figure 5.2) qui est hors de sa zone de transmission, le nœud n_1 peut se déplacer et transmettre le message directement au nœud n_4. En utilisant des nœuds intermédiaires, le temps de transmission des messages peut être inférieur à celui obtenu en forçant n_1 à se déplacer vers n_4. C'est ici que la fonction du calcul du chemin optimal intervient. Ainsi, le nœud n_1 peut s'approcher d'un autre nœud qui sert de relais et lui transmettre le message. Ce nœud relais s'occupe ensuite de l'acheminement du message jusqu'au nœud n_4. Par exemple dans la figure 5.2, le nœud n_1 se déplace vers le nœud n_2, n_2 se déplace vers n_3, ensuite n_3 se déplace et transmet le message au nœud n_4.

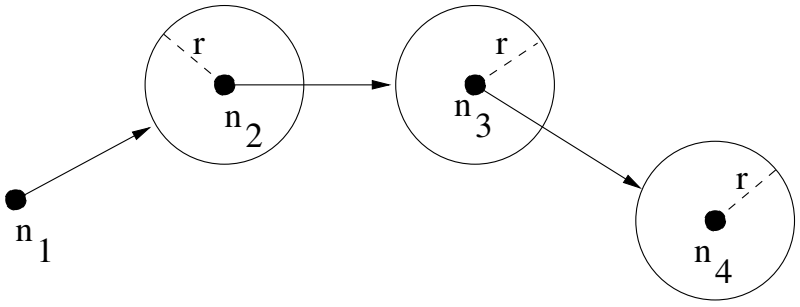


FIG. 5.2 – Transmission de messages

Le problème qui se pose dans ce type de scénario concerne surtout deux questions principales : comment peut-on trouver un nœud qui a modifié sa trajectoire pour acheminer un message et qu'est-ce qui se passe si un nœud doit transmettre plusieurs messages dans différentes directions ? Ainsi, cet algorithme devient dépendant des applications et des compromis peuvent être faits pour adapter ce schéma à une application particulière.

Le deuxième algorithme ne suppose pas une connaissance des mouvements de tous les nœuds. L'algorithme de la figure 5.3 montre la mise à jour des positions des nœuds quand ils ne connaissent pas leurs trajectoires *a priori*. Chaque nœud informe les autres nœuds sur sa position courante. Le pseudo-code présente les mises à jour que reçoit un nœud h_i sur la position du nœud h_0. Pour que ce schéma fonctionne, il faut savoir :

- quand un nœud doit envoyer l'information de mise à jour de sa position ?
- à qui doit-il envoyer cette information ?
- comment faut-il envoyer cette information ?

La modélisation du problème est faite par un graphe dont les sommets représentent les nœuds et le poids des arêtes la distance physique entre les nœuds. Le deuxième algorithme utilise un arbre recouvrant minimal (*minimum spanning tree*) qui contient les arêtes les plus courtes du graphe (en terme de distance) qui assurent une connectivité complète du graphe. Chaque nœud met à jour sa position en informant tous les autres nœuds liés à lui (ses voisins) dans l'arbre recouvrant minimal (cf. première ligne de la boucle *for* dans l'algorithme de la figure 5.3).

Notation :
t_i *: le dernier instant où* h_i *a reçu la mise à jour de la position de* h_0
t *: le temps courant*
(x_{t_i}, y_{t_i}) *: la position de* h_0 *au moment* t_i
(x_t, y_t) *: la position de* h_0 *au moment* t
for *tous les nœuds* $h_1, h_2, ..., h_k$ *qui sont adjacents de* h_0 *dans l'arbre recouvrant minimal* ***do***
  calculer la distance optimale r_i *entre* h_0 *et* h_i
  if $|(x_t, y_t) - (x_{t_i} + v_x(t - t_i), y_{t_i} + v_y(t - t_i))| \geq r$ ***then***
    se déplacer à h_i *pour mettre à jour sa position* $(t_i = t$ *et* $(x_{t_i}, y_{t_i}) = (x_t, y_t))$
  end if
  if *il y a échange de messages entre* h_0 *et* h_i ***then***
    mettre à jour ti et (x_{t_i}, y_{t_i}) *dans le temps et position courants*
  end if
end for

FIG. 5.3 – L'algorithme du nœud h_i

Le protocole force tous les nœuds mobiles à dévier légèrement de leurs routes prédéfinies et déterministes pendant une courte période de temps, afin d'acheminer les messages. Chaque nœud se déplace dans une région et il sait qui, des autres nœuds, maintiennent la trace de ses positions. La mise à jour des positions se fait quand le nœud dévie hors de sa région. Dans ce cas, il informe uniquement ses voisins dans l'arbre recouvrant minimal.

Ce travail introduit un schéma de déploiement d'un réseau ad hoc, mais il ne s'intéresse pas à tous les problèmes de la communication et en particulier au routage. Les algorithmes consistent surtout à minimiser les déviations des nœuds faites pour acheminer un message. Cette approche peut être utile dans le cas d'un réseau où la plupart des nœuds sont connectés et que certains nœuds sont dispersés, pas très loin les uns des autres, mais aussi dans le cas où la distance entre deux nœuds est légèrement supérieure à leur zone de transmission. Ainsi, les nœuds font une petite déviation de leur trajectoire pour acheminer des messages.

5.1.2 Approche de Gerla et al.

Dans [64], Gerla et al. introduisent un mécanisme pour la prédiction de la mobilité. Le but de leur schéma est de fournir un service de maintien de la connexion entre les nœuds avant que les routes n'expirent. Plus précisément, si les paramètres du mouvement de deux nœuds voisins sont connus, tels que la vitesse, la direction et la distance de propagation radio, alors on peut déterminer le temps pendant lequel ces deux nœuds seront connectés. Si on suppose que deux nœuds i et j sont dans la zone de transmission radio r l'un de l'autre, et que (x_i, y_i), v_i, θ_i, (x_j, y_j), v_j, θ_j, $(0 \leq \theta_i, \theta_j < 2\pi)$ sont respectivement les coordonnées, la vitesse et la direction de mouvement pour les nœuds i et j, alors la durée D_t pendant laquelle les deux nœuds seront connectés est donnée par la formule suivante :

$$D_t = \frac{-(ab+cd) + \sqrt{(a^2+c^2)r^2 - (ad-bc)^2}}{a^2+c^2}$$

où

$$\begin{aligned} a &= v_i \cos\theta_i - v_j \cos\theta_j \\ b &= x_i - x_j \\ c &= v_i \sin\theta_i - v_j \sin\theta_j \\ d &= y_i - y_j \end{aligned}$$

Si $v_i = v_j$ et $\theta_i = \theta_j$, alors D_t devient ∞. En fait, D_t est le temps d'expiration du lien entre les deux nœuds. Le schéma utilise l'information GPS sur les positions des nœuds. Cette information est stockée (*piggybacked*) dans des paquets de données durant une connexion et utilisée pour estimer le temps d'expiration du lien entre deux nœuds voisins. Cette prédiction de la mobilité est utilisée pour anticiper les changements de la topologie en reconfigurant les routes avant leur expiration.

5.1.3 Approche de Goldenberg at al.

Dans [33], Goldenberg et al. proposent un schéma de contrôle de mobilité pour améliorer les performances dans un réseau ad hoc. Leur système opère dans un espace physique où les nœuds essaient d'atteindre une configuration pour permettre la communication en minimisant l'usage de l'énergie. Les

deux algorithmes présentés, synchrone (figure 5.4) et asynchrone (figure 5.6), supposent que chaque nœud connaît sa position, par exemple, en utilisant le système GPS. Ainsi, les nœuds qui servent de relais se déplacent vers de nouvelles coordonnées pour assurer la communication en essayant de remplir la contrainte de Stojmenovic et Lin [63] : « tout au long d'une route multi-sauts, les routes droites sont plus efficaces pour minimiser l'usage de l'énergie ».

Algorithme du nœud i

x_i *: position courante du nœud i*
x_{i-1} *et* x_{i+1} *: positions des nœuds* $i-1$ *et* $i+1$
$g \in (0,1]$ *: dumping factor*
repeat
  envoyer x_i *aux voisins* $i-1$ *et* $i+1$
  recevoir x_{i-1} *et* x_{i+1}
  set $x'_i = (x_{i-1} + x_{i+1})/2$
  se déplacer à $x_i + g(x'_i - x_i)$
until *convergence*

FIG. 5.4 – Algorithme synchrone de Goldenberg at al.

Un nœud calcule la distance moyenne entre ses deux voisins, et il se déplace (et ne saute pas) vers ce point. Le mouvement est ralenti ou atténué (*damped*). Cette atténuation est utilisée pour limiter les oscillations (aller retour autour d'un point) qui peuvent se produire et qui augmentent la distance parcourue par les nœuds.

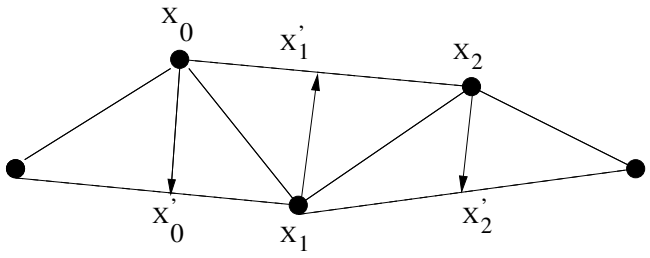


FIG. 5.5 – Déplacement des nœuds relais

Un exemple de déplacement des nœuds relais selon l'algorithme synchrone de la figure 5.4 est présenté dans la figure 5.5. Cette figure montre qu'en exécutant l'algorithme de la figure 5.4, chaque nœud intermédiaire essaie de se positionner dans la ligne droite entre son nœud prédécesseur et son nœud suivant dans le but de remplir la contrainte de Stojmenovic et Lin.

Algorithme du nœud i

subroutine *listen()*
en recevant x_{i-1} *faire* $L = True$
en recevant x_{i+1} *faire* $R = True$
en recevant $moving_{i-1}$ *faire* $L = False$
en recevant $moving_{i+1}$ *faire* $R = False$

L *et* R *: variables internes booléennes*
$moving_i$ *: message signalant que le nœud* i *commence à se déplacer*
x_i *: position courante du nœud* i
x_{i-1} *et* x_{i+1} *: positions des nœuds* $i-1$ *et* $i+1$
$g \in (0, 1]$ *: dumping factor*
repeat
  envoyer x_i *aux voisins* $i-1$ *et* $i+1$
  repeat
    listen()
  until $(L\&\&R) == True$
  envoyer $moving_i$ *aux voisins* $i-1$ *et* $i+1$
  set L=False, R=False
  set $x'_i = (x_{i-1} + x_{i+1})/2$
  se déplacer vers $x_i + g(x'_i - x_i)$
  repeat
    listen()
  until *arriver en temps borné*
until *convergence*

FIG. 5.6 – Algorithme asynchrone de Goldenberg at al.

Les algorithmes définissent les opérations des nœuds intermédiaires qui servent de relais et assurent la connectivité entre deux nœuds voisins. Cela fait que pendant le déplacement, la fonctionnalité du chemin n'est pas com-

promise et que chaque nœud maintient toujours le contact avec ces deux voisins afin de calculer sa nouvelle destination. Ces algorithmes prennent en compte des réseaux denses avec un fort degré de connectivité.

L'algorithme asynchrone de la figure 5.6 ne suppose pas une information globale sur les nœuds du réseau, mais exige de chaque nœud qu'il rejoigne sa position dans un temps borné. Un souci potentiel de cet algorithme, à cause de sa nature asynchrone, est le blocage (*deadlock*), mais les auteurs [33] prouvent que si la transmission des messages est fiable, alors l'algorithme est sans blocage.

5.1.4 Approche de Lin et al.

Dans [46], Lin et al. étudient le problème de rendez-vous des agents mobiles. Il s'agit du comportement collectif d'un groupe de $n > 1$ agents mobiles qui se déplacent dans un plan. Chaque agent mobile est capable de connaître les positions des autres agents mobiles qui se trouvent dans sa « zone de détection » (*sensing region*)[1]. Le problème de rendez-vous essaie de définir une stratégie « locale » de contrôle pour chaque agent qui, sans une communication active entre les agents, les oblige par la suite à se rendre dans un endroit non spécifié. Leur intérêt est exclusivement la conception des stratégies qui dictent quand et où les agents doivent se déplacer. Ainsi, ils ne traitent pas la façon dont les manœuvres sont effectuées réellement ou la façon dont les collisions sont évitées.

Pour résoudre le problème, deux types de solutions sont présentées. La première repose sur un schéma synchrone, c'est-à-dire sur la synchronisation mutuelle de chaque agent mobile avec une horloge globale. La deuxième solution est un schéma asynchrone qui repose sur des stratégies « locales » qui sont indépendantes les unes des autres, c'est-à-dire sans référence à une horloge globale.

Pour se rendre à un rendez-vous avec les autres agents, chaque agent exécute une séquence de manœuvres *stop-and-go*. Chacune de ces manœuvres a lieu dans un intervalle de temps qui est divisé en deux sous-intervalles consécutifs. Le premier appelé période de détection (*sensing period*) est un intervalle de longueur fixe τ_D pendant lequel l'agent est immobile. Le deuxième

[1] C'est la zone où un agent mobile peut détecter la présence des autres agents. Par exemple pour les agents mobiles équipés de cartes 802.11b la zone de détection (*carrier sensing range*) est un disque de rayon d'environ 550 m et la zone de transmission est un disque de 250 m de rayon, en l'absence d'obstacle

appelé période de manœuvre (*maneuvering period*), est un intervalle de longueur variable pendant lequel l'agent se déplace de sa position courante vers sa prochaine position. Les positions successives pour chaque agent sont *choisies* de telle sorte qu'elles soient au maximum à r_M unités l'une de l'autre, où r_M est une distance positive prédéfinie inférieure au rayon de sa zone de détection r. En plus, il est supposé que pour chaque agent i on choisit une valeur $\tau_{M_i} > 0$, appelée temps de manœuvre, qui est assez grande pour faire en sorte que le déplacement de l'agent i entre deux positions successives se fait au maximum en τ_{M_i} secondes.

Dans le cas synchrone, les temps de manœuvre pour tous les agents sont les mêmes, τ_M. Tous les agents se synchronisent dans le sens que tous sont immobiles pendant les périodes de détection et tous se déplacent seulement pendant les périodes de manœuvre, lesquelles ont une durée fixe τ_M.

Dans le cas asynchrone, pour chaque agent i, l'axe du temps peut être divisé dans une séquence d'intervalles de temps $[0, t_{i1}), [t_{i1}, t_{i2}), ..., [t_{i(k_i-1)}), t_{ik_i}), ...$ chacun de longueur $\tau_D + \tau_{M_i}$ où $\tau_D > \tau_{M_i}$. Chaque intervalle $[t_{i(k_i-1)}), t_{ik_i})$ est constitué d'une période de détection de longueur fixe τ_D pendant laquelle l'agent est immobile, suivie d'une période de manœuvre d'une durée maximum τ_{M_i} pendant laquelle l'agent i se déplace. Même si tous les agents utilisent le même temps τ_D, ils opèrent de manière asynchrone dans le sens où les temps $t_{i1}, t_{i2}, ..., i \in 1, 2, ..., n$ ne sont pas corrélés. Ainsi, la stratégie de chaque agent peut être implémentée indépendamment des autres sans l'aide d'une horloge globale.

Ce travail ne s'occupe pas de la manière dont les agents mobiles se déplacent. Il ne se situe pas dans le domaine des réseaux ad hoc mais il est intéressant du point de vue de l'utilisation de la mobilité. Le rendez-vous des agents mobiles n'a pas pour but d'assurer la communication entre ces agents.

5.1.5 Autres approches sur le contrôle de mobilité

Même si la mobilité a été largement étudiée, les études réalisées portent généralement sur le mouvement aléatoire des nœuds mobiles, e.g. [41], et non sur le mouvement contrôlé. Par exemple dans [34], Grossglauser et Tse ont montré que le mouvement aléatoire des nœuds peut améliorer le débit du réseau. Dans [16], Chakraborty et al. ont étudié des algorithmes qui essayent de prédire les mouvements des nœuds afin de réduire la consommation de l'énergie.

Le mouvement des nœuds présente une opportunité : si les nœuds se

déplacent d'une manière « appropriée » les algorithmes peuvent en tirer profit et avoir de meilleures performances. Dans [37], Hatzis et al. définissent la notion du *protocole de contrôle des mouvements (compulsory protocol)* selon laquelle un ensemble de nœuds se déplace d'une manière spécifique. Un tel protocole force les nœuds à se déplacer selon un schéma afin de satisfaire les objectifs (e.g., se rencontrer plus souvent, se disperser dans une zone géographique, etc.). Ils présentent un protocole pour la sélection d'un leader unique d'un ensemble de nœuds. Commençant par une configuration où tous les nœuds sont dans le même état, le protocole exige des nœuds certains mouvements aléatoires afin d'assurer son exécution correcte et aboutit à une configuration avec un leader qui connaît la taille du réseau.

Tous ces travaux, ainsi que le nôtre, peuvent se situer dans la classe des protocoles de contrôle des mouvements définie par Hatzis et al. dans [37]. Ils montrent que le contrôle de mobilité est un domaine de recherche actif. Ces dernières années, des progrès ont été réalisés dans l'étude des systèmes mobiles distribués qu'ils soient naturels ou artificiels. Mais ces études ne sont pas focalisées sur la communication entre les composants de ces systèmes et aucun ne considère le routage.

5.2 Notre vue sur la topologie - introduction

Dans la section 2.1 du chapitre 2 nous avons présenté une classification des réseaux sans fil. Une telle classification est évidemment valable pour les réseaux ad hoc si l'on prend comme critère l'étendue géographique ou la topologie et l'infrastructure. Dans ces topologies, la connectivité des nœuds est plutôt la règle et si les nœuds se déplacent hors de la portée de ses interfaces de communication, alors la communication devient impossible. Dans cette thèse nous considérons un autre cas, celui des réseaux de grande taille où la topologie du réseau est très large par rapport à la zone de transmission des nœuds mobiles. Les nœuds sont dispersés et le réseau n'est pas dense comme par exemple dans [32]. Dans ce contexte nous montrons comment les mouvements peuvent être utilisés pour acheminer efficacement les messages avec la coopération des nœuds intermédiaires.

Dans les travaux présentés dans les sections précédentes de ce chapitre, le contrôle de mobilité est fait surtout pour assurer la connectivité du réseau. Nous avons plutôt étudié ce problème sous un autre angle, celui des améliorations que ce contrôle de la mobilité peut amener au routage. Les travaux

précédents supposent que les nœuds connaissent leurs coordonnées exactes et parfois celles de leurs voisins en utilisant des techniques de localisation. Cette connaissance n'est pas nécessaire pour l'exécution de notre algorithme.

Dans un premier temps, nous définissons le contexte de l'environnement d'un réseau de grande taille où notre algorithme est susceptible de s'exécuter. Ensuite, nous définissons un schéma de contrôle de mouvements des nœuds du réseau. Les nœuds se déplacent selon ce chéma afin de mettre en place un mécanisme de rendez-vous ce qui ne les empêche pas d'accomplir d'autres tâches autre que communiquer. Le mécanisme de rendez-vous assure aussi une connectivité locale du réseau ce qui améliore le routage. Pour prouver cette amélioration nous avons testé différentes métriques comme le nombre de sauts qu'un message fait pour atteindre sa destination ou le délais de bout en bout de la transmission des messages. Les résultats obtenus sont satisfaisants. Notre solution assure une communication entre n'importe quels nœuds du réseau dans un temps borné. Nous avons aussi prouvé la faisabilité du schéma de contrôle des mouvements par l'implantation de l'algorithme dans des robots.

Notre solution pourra être utilisée dans le cas des topologies mentionnées dans la section 2.1 du chapitre 2. L'algorithme est aussi valable si la topologie du réseau est dense, mais dans ce cas, il est moins utile.

Notre solution pourra être utilisée aussi dans un terrain en présence d'obstacles. Dans cette thèse nous ne nous occupons pas du problème de navigation dans une surface $2D$. Cela est un vaste domaine de recherche dans la robotique [47] et n'est pas discuté dans cette thèse.

5.3 Discussion

Avec le progrès rapide de la technologie, les nœuds mobiles augmenteront leurs capacités de déplacements. De plus en plus d'agents mobiles seront disponibles avec des ressources d'énergie (batteries) assurant une autonomie plus large. Par exemple, beaucoup de robots mobiles sont équipés de différents capteurs. Il existe aussi des robots insectes qui se déplacent de plusieurs manières, volent ou glissent sur l'eau. La communication sera alors essentielle pour le bon fonctionnement de ces réseaux mobiles. On peut imaginer plusieurs cas où la mobilité peut être utilisée pour l'amélioration des performances des réseaux. Un tel scénario peut être un déploiement d'agents mobiles ou de robots avec de puissants capteurs dont le but est d'intercepter,

de rassembler des informations et puis de transmettre des données concernant une cible ennemie. Si ces nœuds sont capables de se déplacer en assurant par exemple une communication fiable vers leurs base, alors l'utilité de ces agents sera considérablement accrue. On peut aussi imaginer des réseaux mobiles deployés géographiquement qui s'adaptent au déplacements des nœuds pour mieux servir les demandes de communication et/ou de confidentialité de cette communication.

Il existe aussi des situations où la source d'énergie est renouvelable pour les nœuds mobiles mais séparée d'une source d'énergie non-renouvelable pour les communications. De telles situations pourraient émerger dans des systèmes hybrides bio-électroniques comme par exemple des humains portant des radios qui fonctionnent avec des batteries non-rechargeables. On peut imaginer aussi un système où des insectes équipés avec de capteurs et d'émetteurs radio, et dont le mouvement est commandé par une interface neuro-électronique.

En général, les déploiements à long terme qui demandent habituellement des modèles de communication persistants sont les premiers candidats à l'application de la mobilité afin d'améliorer les performances du réseau.

Chapitre 6

Conception de l'algorithme

Introduction

La gestion de la mobilité dans un réseau ad hoc doit faire face à plusieurs défis comme le profil très varié de la mobilité des nœuds. Ainsi, l'amélioration des performances de routage et la connectivité des nœuds sont très influencées par la nature du modèle de mobilité des nœuds.

Les réseaux ad hoc sont de nature dynamique dans le sens où les connexions locales sont temporaires et peuvent changer pendant que les nœuds se déplacent. La manière dont les nœuds se déplacent peut changer d'un nœud à l'autre, puisque certains nœuds peuvent s'arrêter pour exécuter des tâches dépendantes de leurs zones de déplacement (e.g. prendre des mesures, recharger les batteries ou sonder l'environnement). Dans un tel environnement, établir une communication entre les nœuds par le biais d'un protocole distribué pourra être une tâche non triviale.

La manière la plus commune d'établir la communication est de former un chemin de nœuds intermédiaires où il existe un lien entre deux nœuds s'ils se trouvent dans la zone de transmission l'un de l'autre. Cette approche est utilisée dans des réseaux ad hoc qui couvrent un petit espace ou des réseaux qui sont denses.

Dans les réseaux ad hoc de grande taille (*wide area ad hoc networks*) ou dans les VANET (*Vehicular Ad hoc NETworks*) [2, 62], il est difficile d'établir des chemins efficaces. Et même si un tel chemin est établi, si des liens de communications tombent le long de ce chemin à cause du déplacement d'une partie des nœuds qui le composent, alors le chemin ne sera plus valide.

Une approche différente pour résoudre le problème précédent est de tirer profit des mouvements des nœuds mobiles en les faisant communiquer quand ils se rencontrent. Dans ce cas, si les nœuds se rencontrent souvent et même s'ils sont déployés dans une zone assez large, une communication peut avoir lieu.

Une solution pour alléger ces problèmes est de forcer les nœuds mobiles à se déplacer selon un schéma spécifique afin d'accomplir les demandes du protocole. Notre approche consiste notamment à forcer les nœuds à se déplacer vers des points de rencontre avec d'autres nœuds qui se déplacent dans les zones adjacentes.

Ce chapitre est composé de deux parties. La première partie (section 6.1) présente l'approche intuitive de notre solution ainsi que l'environnement du réseau étudié dans cette thèse. La deuxième partie (section 6.2) présente les notations utilisées dans ce manuscrit. Ensuite, elle définit le concept de base de notre solution (sous-sections 6.2.2 et 6.2.3), ainsi que le modéle des nœuds qui forment le réseau. Enfin, un résumé conclut le chapitre.

6.1 Une approche intuitive de la solution

Cette section fait un historique de notre travail. Elle commence par décrire l'architecture mise en place pour la communication d'un ensemble de robots. Ensuite, elle présente l'extension de cette architecture aux réseaux ad hoc et décrit la topologie que nous considérons pour appliquer notre idée sur le contrôle des mouvements des nœuds mobiles dans un réseau ad hoc de grande taille.

6.1.1 Architecture de communication entre robots

L'idée de se servir du mouvement pour améliorer le routage dans un réseau ad hoc a débutée lors de notre travail sur un schéma de communication pour un ensemble de robots [10, 12]. Dans ce travail, nous avons étudié les mouvements d'un ensemble de robots afin d'assurer l'acheminement des messages d'une source à une destination. Comme source ou destination, nous considérions une base fixe, par exemple un PC, ou un robot.

Nous nous sommes intéressés à la communication en mode ad hoc d'un ensemble de robots déployés sur un terrain. Outre les cas présentés dans la section 5.3 du chapitre précédent, on peut imaginer l'application d'une

telle architecture dans une situation militaire où un ensemble de robots est déployé dans un terrain miné, ou contaminé, et où l'on veut établir une communication en mode ad hoc entre ces robots pour acheminer des ordres ou rapatrier des informations. Un tel robot peut être équipé avec différents capteurs qui servent à sonder le terrain et à recueillir des informations ou des données. Chaque robot est autonome et se déplace dans une zone limitée. Dans cette zone, le robot peut effectuer diverses tâches et peut communiquer avec d'autres robots quand ils se trouvent dans les zones de transmission les uns des autres. Chaque robot fonctionne en mode ad hoc ; il peut donc être à la fois source, destination ou routeur pour acheminer les messages vers d'autres robots. Ainsi, les messages sont acheminés d'un robot à l'autre jusqu'à ce qu'ils arrivent à destination.

Dans ce contexte, nous avons supposé que la zone de transmission des robots est très petite comparée aux zones où ils se déplacent. Les robots sont plutôt déconnectés et ils acheminent des messages pendant leur déplacement. Ainsi, d'un point de vue abstrait, les robots ont des liens de communication avec une durée de vie limitée. La communication est complètement asynchrone et elle est possible seulement s'il existe une séquence valide dans le temps, de liens actifs qui forment un chemin de la source à la destination. La communication a lieu dans quelques zones « spéciales » qui sont des zones d'intersection entre les zones de transmission des robots. Dans notre modèle [10, 13, 11], les robots se donnent rendez-vous dans plusieurs points du plan pour pouvoir acheminer des messages, toujours en mode ad hoc. Tandis que dans [46] tous les agents mobiles sans une communication active se donnent rendez-vous dans un point non spécifié du plan.

6.1.2 Des robots aux nœuds d'un réseau ad hoc

L'idée d'utiliser la mobilité pour optimiser le routage dans un réseau ad hoc prend de plus en plus d'ampleur parmi les équipes de recherche. En fait, notre travail sur les robots peut être considéré comme un travail plus général sur des nœuds mobiles constituant un réseau ad hoc. Ainsi, nous nous intéressons à ceux-ci dans la suite de cette thèse.

Une des caractéristiques des réseaux ad hoc est la forte mobilité des nœuds. Cette mobilité est, dans la plupart des cas, imprévisible. Évidemment, cette mobilité accrue dans les réseaux ad hoc influence les performances des réseaux, notamment en ce qui concerne l'énergie des nœuds mobiles, la connectivité des nœuds dans le réseau et le routage. Notre étude de la mobi-

lité concerne justement l'utilisation du contrôle de la mobilité pour assurer de meilleures performances du réseau. Plus précisément, notre idée consiste à forcer chaque nœud à choisir une trajectoire au lieu d'« errer » dans l'environnement où s'étend la topologie du réseau. Tout au long de cette trajectoire, chaque nœud peut communiquer avec d'autres nœuds dans certains endroits spécifiques où les nœuds peuvent se rencontrer ou se croiser. Tous les nœuds suivent un ordonnancement préalablement choisi qu'ils essayent de respecter. En ce qui concerne le routage, la communication aura lieu dans les endroits où les nœuds se rencontrent. Ainsi, notre travail s'inscrit dans le cadre des protocoles de contrôle des mouvements (*compulsory protocols*). Dans ce type de protocoles, on demande à un sous-ensemble de nœuds mobiles de se déplacer d'une manière spécifique. Le mouvement est considéré comme une opportunité. Si les nœuds se déplaçaient d'une manière utile, les algorithmes pourraient tirer profit du mouvement en améliorant les performances du réseau plus efficacement que dans les réseaux statiques.

6.1.3 Contexte de l'environnement et topologie du réseau

Nous considérons des réseaux ad hoc de grande taille où les nœuds participants dans ces réseaux sont plutôt déconnectés. Plus précisément, la topologie du réseau est un plan où un ensemble de nœuds mobiles est déployé et où l'on veut établir une communication en mode ad hoc entre ces nœuds. Les mouvements des nœuds peuvent couvrir ou non toute la topologie du réseau et ils se déplacent de manière indépendante l'un de l'autre.

D'abord, nous supposons que la topologie du réseau est très large et qu'elle est divisée en zones. Dans chaque zone se déplace un seul nœud. La zone de transmission des nœuds est beaucoup plus petite comparée à la taille de la topologie du réseau. Ainsi, chaque nœud peut communiquer uniquement avec les nœuds qui se déplacent dans les zones adjacentes. Les points de communication sont prédéfinis. Cela peut se faire en utilisant par exemple des GPS (*Global Positioning System*) ou différentes méthodes de localisation [6, 59, 60]. Nous supposons aussi qu'entre deux nœuds il existe toujours un chemin qui les relie. Comme les nœuds sont la plupart du temps déconnectés, ce chemin n'est pas une suite de liens de communications, mais un chemin réel (physique) dans la topologie. Autrement dit, il existe toujours une suite de zones qui peut relier deux nœuds se déplaçant dans deux zones distinctes.

Exemple

La figure 6.1 illustre la topologie d'un tel réseau. Parmi d'autres informations, elle indique que 4 nœuds sont attachés aux zones A, B, C et D où ils se déplacent et/ou accomplissent diverses tâches. Les nœuds des zones A et B ont un point de rencontre en commun et les nœuds des zones A et C peuvent communiquer *via* les nœuds des zones B et D.

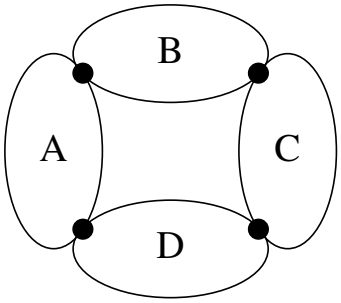


FIG. 6.1 – Exemple de réseau

En réalité, ces points de rencontres sont plutôt des zones de rencontre, c'est-à-dire que les nœuds peuvent se rapprocher jusqu'à ce qu'ils se trouvent dans leurs zones de transmission afin de pouvoir échanger des messages. Pour la modélisation du problème, ces zones de rencontres seront présentées comme des points de rencontre. En effet, ces zones étant beaucoup plus petites que les zones où se déplacent les nœuds, cette modélisation par des points n'affecte pas le problème de communication en général. Il va falloir maintenant trouver un ordonnancement des mouvements des nœuds pour assurer une communication en mode ad hoc entre les nœuds du réseau.

6.2 Concepts et notation

Cette section commence par la présentation de quelques notations sur le comportement du système et définit les zones et les points de rendez-vous qui sont à la base de sa modélisation. Ensuite, elle introduit le modèle des nœuds et une hypothèse sur leur mouvement. Ces concepts et notations seront utilisés dans la suite de la thèse quand on introduira la modéliation du système, les algorithmes d'ordonnancement ainsi que leurs améliorations.

6.2.1 Notation

Pour se familiariser aux termes utilisés dans la suite de cette thèse, cette sous-section définit quelques uns d'entre eux.

Definition 1 *Nous appelons point de rendez-vous le point de rencontre entre deux zones (nœuds) du plan.*

En fait, ces points de rendez-vous sont des points d'intersection entre les zones du plan. Il faut noter que dans un point de rendez-vous les nœuds peuvent accomplir plusieurs tâches, la principale d'entre elles sera de communiquer. D'autres activités, comme par exemple le rechargement des batteries ou le sondage de l'environnement par différents capteurs, peuvent également avoir lieu.

Malgré ce qu'on a supposé pour le système dans la section 6.1.3, il est toujours possible de trouver un ordonnancement dans lequel deux nœuds, qui « théoriquement » peuvent se rencontrer, ne se croisent jamais. Cela veut dire qu'il n'est pas toujours possible d'assurer une communication entre deux nœuds dans un temps borné. Ainsi, il est possible de trouver un ordonnancement des mouvements des nœuds selon lequel les nœuds ne se rencontrent jamais. Dans ce cas, nous dirons que le système est bloqué (*frozen*). Le but final de notre solution sera d'assurer que l'ordonnancement des mouvements des nœuds n'aboutira pas à un système bloqué. Enfin, dans la suite de cette thèse, les termes « réseau » et « système » sont utilisés de façon indifférente.

6.2.2 Définiton des zones et des points de rendez-vous

La localisation est essentielle pour beaucoup d'applications dans les réseaux ad hoc. Dans tous les systèmes de localisation comme *GPS* ou [56] et [61], une erreur qui dépend de l'environnement doit être prise en compte. Le *GPS* est relativement précis mais il exige la visibilité de ses satellites et ainsi il est inefficace à l'intérieur des bâtiments. En plus, il est coûteux. Les systèmes basés sur l'infrastructure sont pour la plupart destinés à fonctionner à l'intérieur des bâtiments et ils ont une marge d'erreur de quelques mètres. Les systèmes de localisation ad hoc peuvent parfois faire des erreurs de localisation importantes dues aux facteurs environnementaux affectant ainsi les mesures des positions.

Dans notre solution, la communication entre deux nœuds aura lieu dans des zones de rendez-vous. Ainsi, elle exige des nœuds de connaître seulement leur position approximative. Tandis que dans les autres travaux les nœuds doivent connaître leurs positions exactes, dans notre architecture, il suffit qu'ils connaissent les positions des points de rendez-vous.

Pour la division du plan en zones, nous nous sommes basés sur la notion classique de la géométrie algorithmique des diagrammes de Voronoï [70]. Un diagramme de Voronoï résulte de la division d'un plan avec n points de référence (les nœuds dans notre cas) en n polygones convexes tels que chaque polygone contient exactement un point de référence et chaque point dans un polygone donné est plus proche de son point de référence central que de tout autre point de référence. Le temps de calcul du pire cas du diagramme de Voronoï est de l'ordre de $O(n \log n)$ [70].

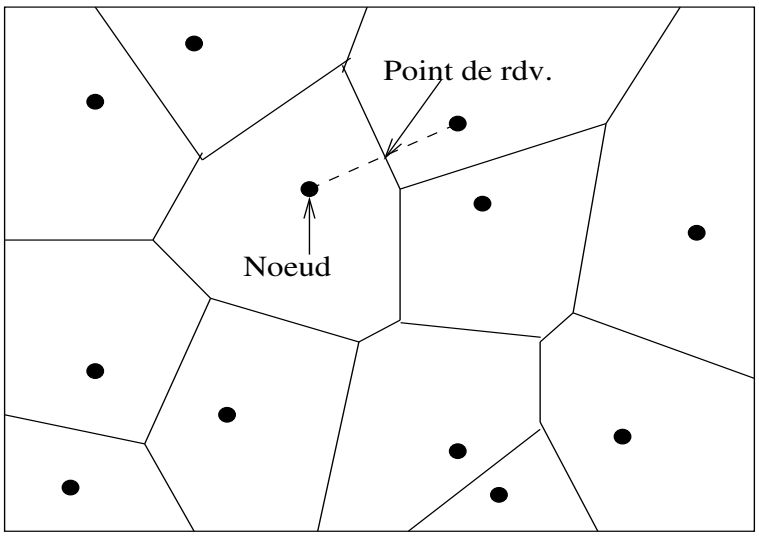


FIG. 6.2 – Exemple de diagramme de Voronoï

Ainsi, si on considère comme points dans le plan les nœuds du réseau, alors les polygones convexes seront les zones où vont se déplacer ces nœuds. Un exemple de la division d'un plan avec un ensemble de nœuds en diagramme de Voronoï est représenté dans la figure 6.2.

Après la division du plan en zones, les points de rendez-vous peuvent être définis sur les segments de la structure du diagramme de Voronoï comme le

point d'intersection entre le segment qui relie deux points de référence de deux zones voisines avec un segment de la structure du diagramme.

Enfin, il faut noter que dans les exemples, entre deux zones voisines peut ne pas avoir toujours un point de rendez-vous. Cela peut être dû par exemple, à la présence d'obstacles dans la frontière entre deux zones.

6.2.3 Contraintes

Notre objectif sera de forcer les nœuds à se déplacer selon un schéma spécifique afin d'assurer que la communication entre deux nœuds du réseau aura lieu dans un temps borné. Dans la section 6.1.3, nous avons décrit la topologie du réseau où l'on veut appliquer notre idée de contrôle des mouvements. Mais, en l'état actuel, il est possible de trouver un ordonnancement des mouvements des nœuds dans lequel ils ne se croisent pas et tel que le système soit bloqué. Cela implique qu'il ne sera pas possible d'assurer que le délai de bout en bout de la transmission des messages sera borné.

Ainsi, il est donc nécessaire d'ajouter une condition sur les déplacements des nœuds. Compte tenu des ressources limitées des nœuds en terme d'énergie, nous avons essayé de trouver une contrainte qui puisse être respectée avec un coût minimum. Un moyen simple consiste à forcer les nœuds à s'attendre mutuellement :

le premier nœud qui arrive sur le point de rendez-vous attend son homologue ou voisin, avant de continuer son déplacement.

Cette contrainte est très forte et il est possible de trouver un ordonnancement des mouvements où les nœuds sont arrêtés en attente de leurs voisins. Par exemple, dans le réseau présenté dans la figure 6.3, si le nœud A attend dans le point de rendez-vous 1, le nœud B attend dans le point 3, C dans 2 et enfin D dans 4, alors le système pourra être bloqué.

Pour éviter le blocage (*deadlock*) dans le système, nous allons introduire un algorithme qui ordonnance les mouvements des nœuds et qui assure que deux nœuds, ayant un point de rendez-vous en commun, se rencontreront dans un temps borné. Cette condition garantit qu'un message qui suit un chemin de points de rendez-vous atteindra sa destination dans un temps borné.

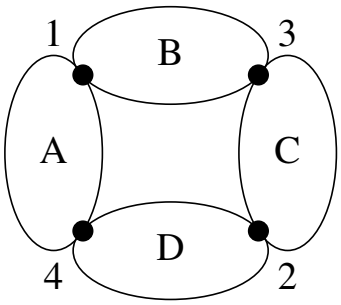


FIG. 6.3 – Exemple de réseau

6.2.4 Modèle des nœuds du réseau

Pour mieux comprendre les algorithmes qui seront présentés dans les chapitres suivants, il faut définir le modèle des nœuds qui forment le réseau ad hoc défini dans les sections précédentes et pour lequel notre solution est conçue. Ce modèle concerne :

La communication

Une communication a lieu dans les points de rendez-vous. Chaque nœud peut être une source de messages en les générant et en les transmettant, mais il fonctionne aussi comme un routeur en acheminant des messages. La communication entre les nœuds se fait par messages et non par flots. Quand un nœud termine l'acheminement d'un message, il ne s'occupe plus de son acheminement jusqu'à la destination. Ainsi, quand une route vers une destination, impliquant certains nœuds intermédiaires, est définie, ces nœuds ne sont pas obligés d'attendre avant et après l'acheminement que le message arrive à destination. Ces nœuds intermédiaires peuvent être impliqués dans l'acheminement d'autres messages. De ce point de vue, ce modèle de routage est asynchrone. Il n'est pas un modèle temps réel et il ne peut pas être appliqué, par exemple, pour la téléphonie. Quand les nœuds se rencontrent dans les points de rendez-vous, la communication a lieu au plus tôt.

La consommation d'énergie

L'énergie est un paramètre important lorsqu'on conçoit un réseau ad hoc. Nous considérons des réseaux ad hoc où les nœuds mobiles ont une source

épuisable d'énergie (batteries) [1]. Dans notre modèle, les nœuds consomment de l'énergie quand ils se déplacent entre les points de rendez-vous, mais aussi quand ils communiquent. Pour améliorer les performances de notre solution, nous essayons de réduire la consommation de l'énergie en réduisant les déplacements et les communications inutiles.

Le déplacement

Chaque nœud se déplace dans sa zone entre les points de rendez-vous. Ce déplacement constitue une tâche avec une durée qui inclut le temps de déplacement mais aussi la durée d'autres processus qu'un nœud peut exécuter comme, par exemple, sonder l'environnement ou rassembler des informations.

Dans un point de rendez-vous, un nœud s'arrête et attend pour rencontrer son voisin s'il est le premier à arriver. Cette tâche est représentée par la primitive *wait()* dans les algorithmes. Si à l'arrivée d'un nœud dans un point de rendez-vous, il y trouve son voisin, alors après une éventuelle communication, les deux nœuds déclenchent le mouvement vers leurs points de rendez-vous suivants respectifs. Cette tâche est représentée par la primitive *notify()* dans les algorithmes.

Dans le système, certaines tâches comme le déplacement et la communication peuvent avoir lieu en parallèle.

Les pannes

Un nœud mobile peut subir des défaillances. Dans cette thèse nous considérons les défaillances par arrêt, par exemple, quand la source d'énergie est épuisée. Quand un nœud tombe en panne, il ne peut plus parcourir ses points de rendez-vous. Un nœud peut se rétablir après une panne, par exemple par une intervention extérieure (rechargement de batteries).

6.3 Résumé

Ce chapitre introduit les problèmes et les motivations qui nous ont amené à l'idée d'utiliser la mobilité pour améliorer les performances d'un réseau ad hoc. Nous considérons des réseaux ad hoc de grande taille où les distances

[1] Il existe des nœuds, que nous ne les considérons pas, fonctionnant à l'aide de panneaux solaires.

entre les nœuds sont beaucoup plus grandes que leur zone de transmission. Dans ce type de réseaux les nœuds sont plutôt déconnectés et établir une communication entre eux représente un défi. Nous avons supposé que chaque nœud se déplace dans sa propre zone du plan. Ainsi, nous avons essayé de tirer profit de la mobilité des nœuds en les forçant à se déplacer vers des zones de rendez-vous entre leurs zones de déplacement, où ils peuvent communiquer. Ces zones peuvent être définies par le diagramme de Voronoï qui divise un plan avec n points (les nœuds du réseau) en n polygones convexes. Ce chapitre présente aussi le modèle des nœuds qui forment le réseau que nous considérons en définissant les conditions de la consommations d'énergie, leur communication, leur déplacement et leurs tâches.

Chapitre 7

Modélisation de la topologie

Introduction

Dans un environnement très dynamique comme les réseaux ad hoc, où les problèmes de communication sont fréquents à cause du haut degré de mobilité ou de manque d'énergie, la robustesse est très importante. Ainsi, la capacité d'adaptation à la dynamique du réseau afin d'atteindre un « comportement » normal et des performances raisonnables, joue-t-il un rôle majeur dans notre modélisation.

Ce chapitre présente la modélisation du problème. Nous essayons de combiner la mobilité et la robustesse de notre solution afin de la rendre adaptable. La base de notre schéma est le contrôle de la mobilité. Toutefois, nous essayons de ne pas beaucoup contraindre les mouvements des nœuds afin de ne pas nuire la dynamique du réseau. Ainsi, nous essayons d'assurer un temps borné de la transmission des messages en mettant en œuvre un mécanisme de rendez-vous quand il s'agit de communiquer. Nous évaluons dans ce cas le délais de la transmission des messages et la longueur des routes. De plus, notre mécanisme doit être économe en énergie (*energy-efficient*). Minimiser les déplacements, une des raisons principales de la consommation d'énergie, peut rendre le mécanisme plus efficace.

Ce chapitre commence par présenter la modélisation de la topologie du réseau et introduit un algorithme d'ordonnancement des mouvements dans la section 7.3. Une série de propriétés sur cette modélisation est présentée dans la section 7.4. La section 7.5 généralise notre algorithme et la section 7.6 définit un schéma de routage dans le réseau. Enfin, un résumé conclut ce

chapitre.

7.1 Modélisation du problème

Nous avons considéré comme topologie du réseau un plan $2D$, où un ensemble de nœuds mobiles est déployé. Selon ce que nous avons supposé dans la section 6.1.3, la topologie du réseau, donc le plan, sera divisée en zones. Dans la pratique, cette division pourra être dépendante des ressources des nœuds ainsi que de leurs caractéristiques. Pour l'instant, les obstacles qui peuvent être présents dans l'environnement du réseau ne sont pas considérés.

En divisant le plan en zones, il peut arriver qu'une zone ait plus d'un point d'intersection avec d'autres zones. Autrement dit, un nœud peut avoir plusieurs points de rendez-vous. Il s'agit donc maintenant de trouver un ordonnancement des mouvements des nœuds. Notre idée consiste à forcer chaque nœud à choisir une trajectoire afin de traverser tous ses points de rendez-vous selon un ordre approprié. Cet ordre doit aussi assurer que le système ne sera jamais bloqué.

Afin d'introduire un ordonnancement pour les mouvements des nœuds, nous avons modélisé le système par un graphe non-orienté. Ce graphe sera à la base de notre algorithme d'ordonnancement et de la modélisation du système. Il servira aussi à produire les automates qui sont la prochaine étape de la modélisation et qui, de leur coté, serviront entre autres pour introduire une série de propriétés du système. Ce graphe, est défini comme suit :

Definition 2 *Nous appelons graphe des points de rendez-vous le graphe* $\mathcal{G}_{rdv} = (V, E)$*, où* V *est l'ensemble des sommets représentant les nœuds mobiles, et où il existe une arête* $uv \in E$ *si et seulement si les nœuds* u *et* v *ont un point de rendez-vous en commun.*

Ainsi, dans le graphe $\mathcal{G}_{rdv}$, chaque sommet représente un nœud et chaque arête un point de rendez-vous. Ce graphe va nous servir à produire l'ordonnancement des mouvements des nœuds. En plus, puisque les communications auront lieu dans les points de rendez-vous, ce graphe peut être considéré comme le graphe des communications. Il pourra alors être utilisé pour étudier et optimiser les communications dans le réseau. En mettant des poids sur ses arêtes, différentes optimisations peuvent être étudiées selon les métriques que l'on veut considérer comme la longueur des routes et le délais de la transmission des messages.

Exemple

La figure 7.1 représente un plan divisé en zones. Chaque zone est représentée par une lettre de a à g.

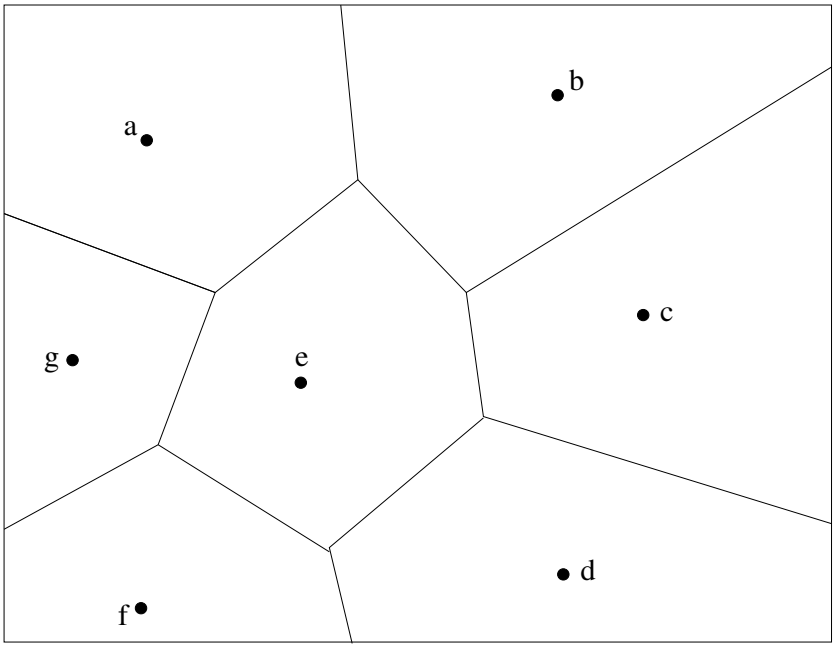


FIG. 7.1 – Plan divisé en zones

Chaque nœud, que l'on va identifier avec les lettres représentant sa zone, se déplace uniquement dans celle-ci. Par la définition 2, on peut générer le graphe des points de rendez-vous. Un point dans chaque zone représentera un sommet du graphe. Ce sommet représente le nœud qui se déplace dans sa zone et s'il a un point de rendez-vous avec un autre nœud, alors il y aura une arête dans le graphe.

Dans la figure 7.1 on suppose qu'entre les zones a et b il y a un point de rendez-vous, donc il y aura une arête entre les zones a et b. La même logique est suivie pour les autres zones entre lesquelles il est supposé d'exister des points de rendez-vous. Le graphe des points de rendez-vous (en gras) ainsi obtenu est représenté dans la figure 7.2.

Tout au long de ce chapitre nous allons nous référer à cet exemple (figure 7.2) pour expliquer au fur et à mesure les différents notions introduites.

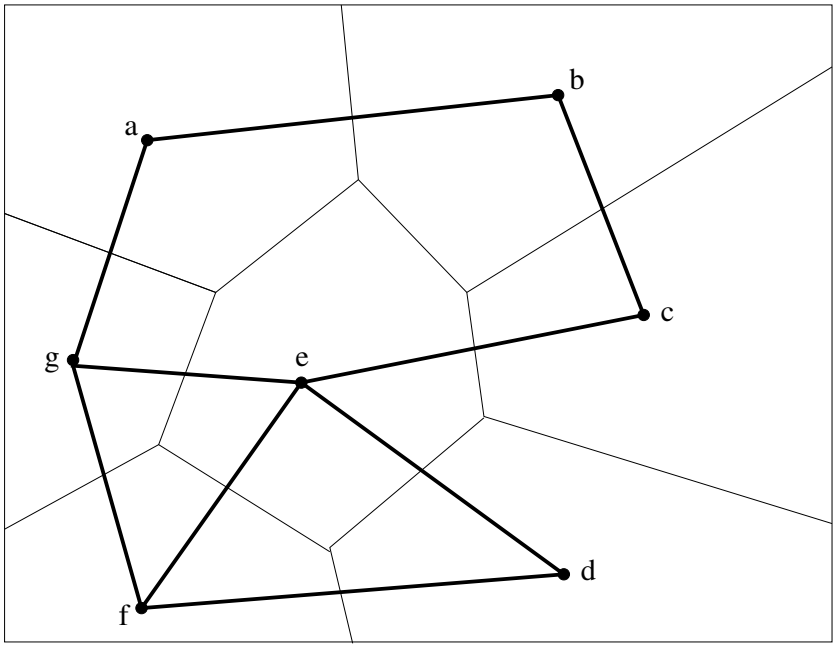


FIG. 7.2 – Plan divisé en zones et son graphe

Pour l'instant nous ne considérons que des rencontres en couple dans les points de rendez-vous mais, comme nous allons le montrer plus tard, les rencontres à plus de deux nœuds dans un même point de rendez-vous peuvent être considérées comme valables.

7.2 Ordonnancement des mouvements

Après avoir modélisé le système par le graphe des points de rendez-vous $\mathcal{G}_{rdv}$, revenons à notre idée initiale, c'est-à-dire essayer d'introduire un schéma de contrôle des mouvements. Autrement dit, il s'agit notamment de forcer les nœuds à se déplacer selon un ordre prédéfini afin de satisfaire les demandes du protocole de communication.

Afin d'introduire un ordonnancement sur les mouvements des nœuds mobiles, nous avons d'abord choisi une numérotation des points de rendez-vous, c'est-à-dire des arêtes du graphe des points de rendez-vous $\mathcal{G}_{rdv}$. Cette numérotation est complétement **aléatoire** et elle est définie comme suit :

Definition 3 *Étant donné un graphe de points de rendez-vous* $\mathcal{G}_{rdv} = (V, E)$ *tel que* $|V| = M$ *et* $|E| = N$*, une numérotation* σ *est une bijection de* E *à* $[1..N]$.

Pour l'exemple du graphe de la figure 7.2, une numérotation de ses arêtes est présentée dans la figure 7.3.

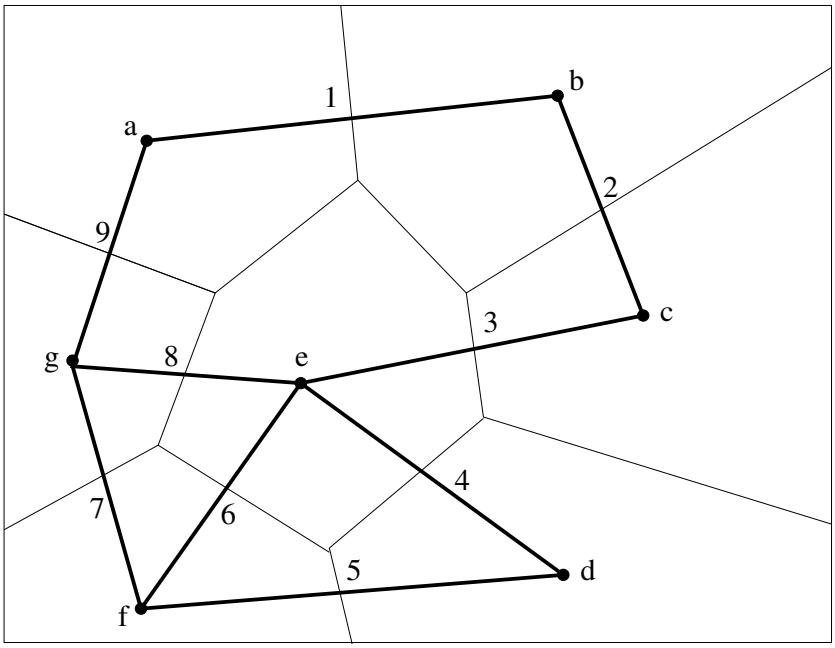


FIG. 7.3 – Graphe numéroté

L'ordonnancement des mouvements des nœuds consiste alors à les forcer à parcourir leurs points de rendez-vous dans l'ordre de la numérotation en respectant la contrainte introduite dans la section 6.2.2 selon laquelle dans un point de rendez-vous, chaque nœud attend son voisin avant de continuer son déplacement.

Étant donné un graphe $\mathcal{G}_{rdv} = (V, E)$ et une numérotation σ, nous associons à chaque nœud n_k ($k \in [1..M]$), un automate $\mathcal{A}_k$ qui représentera son ordonnancement. Un tel automate représente le parcours par le nœud de tous ses points de rendez-vous. Nous avons fait le choix que le nœud parcourt ses points de rendez-vous en **ordre croissant**. Chaque automate $\mathcal{A}_k$, que l'on appelle automate local, est défini comme un ensemble d'états Q_k, de

transitions $\mathcal{T}_k$ et d'un état initial i_k. Les états de l'automate sont les points de rendez-vous et les transitions entre ces états représentent les mouvements entre les points de rendez-vous. L'automate local pour un nœud n_k est défini formellement comme suit :

Definition 4 (Automate local) *On appelle automate local du nœud n_k, l'automate $\mathcal{A}_k = \{Q_k, \mathcal{T}_k, i_k\}$ tel que :*

- *il existe un état q dans l'automate pour chaque point de rendez-vous e du nœud n_k. Pour chacun d'entre eux on note $label(q) = \sigma(e)$;*
- *il existe une transition $t \in \mathcal{T}_k$ de l'état de label $\sigma(e_i)$ à l'état de label $\sigma(e_j)$*
  - *si $\sigma(e_i) < \sigma(e_j)$ et qu'il n'existe pas d'état $q \in Q_k$ tel que $\sigma(e_i) < label(q) < \sigma(e_j)$, ou*
  - *si $\sigma(e_i) = max\{label(q), q \in Q_k\}$ et $\sigma(e_j) = min\{label(q), q \in Q_k\}$;*
- *l'état initial de l'automate est $i_k = min\{label(q), q \in Q_k\}$.*

La figure 7.4 représente les automates locaux, obtenus à partir de la numérotation du graphe de la figure 7.3, pour chaque nœud.

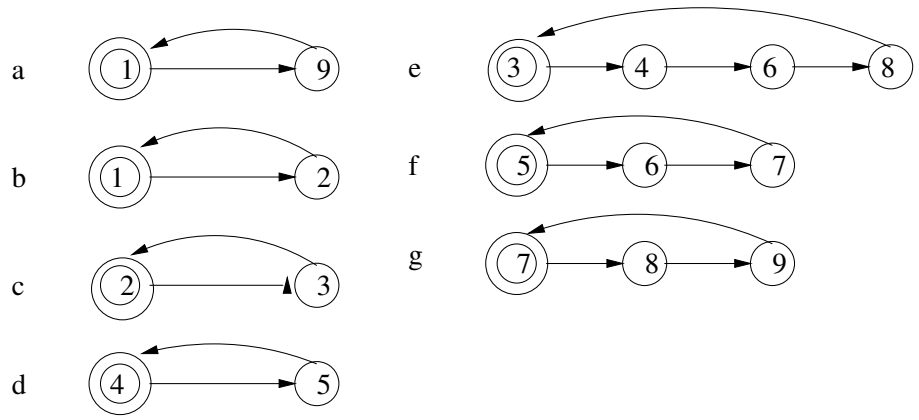


FIG. 7.4 – Les automates locaux

Les transitions entre les états dans ces automates locaux suivent un ordre croissant depuis leurs états initiaux respectifs pour chaque nœud. Il faut noter l'équivalence état/point de rendez-vous et transition/mouvement entre deux points de rendez-vous. L'ensemble des automates locaux représente l'ensemble des états locaux du système. En effet, ils indiquent où se trouve un

nœud pendant le parcours de ses points de rendez-vous. Ils serviront aussi à définir un autre automate (voir section 7.4) qui représentera l'état global du système à chaque instant.

Il faut noter aussi qu'au début, chaque nœud est initialisé dans son point de rendez-vous ayant le plus petit numéro. Ainsi, on a dès le début au moins deux nœuds dans le même point de rendez-vous, celui numéroté avec 1. Cela assure le déclenchement des mouvements car, dès le départ il y aura ces deux nœuds qui vont se déplacer vers leurs points de rendez-vous suivants dans leurs ordres respectifs. Pour l'exemple de la figure 7.4, si on considère les nœuds a et b, alors :
Pour le nœud a : l'automate local $\mathcal{A}_a$ est l'ensemble d'états $Q_a = \{1, 9\}$, et de transitions $\mathcal{T}_a = \{1 \rightarrow 9, 9 \rightarrow 1\}$, donc deux états et deux transitions.
Pour le nœud b : l'automate local $\mathcal{A}_b$ est l'ensemble d'états $Q_b = \{1, 2\}$, et de transitions $\mathcal{T}_b = \{1 \rightarrow 2, 2 \rightarrow 1\}$, donc deux états et deux transitions.
Au départ les nœuds a et b sont dans l'état de même label 1, ce qui signifie qu'ils sont dans le même point de rendez-vous. Le nœud a va alors se diriger vers le point de rendez-vous 9, tandis que b va se diriger vers le point de rendez-vous 2.

7.3 Algorithme d'ordonnancement

L'architecture de contrôle de mobilité que l'on veut établir est basée sur la modélisation du système par le graphe des points de rendez-vous, sa numérotation aléatoire et le mécanisme des rendez-vous. Par ailleurs, les automates locaux avec les équivalences état/point de rendez-vous et transition/mouvement, représentent totalement l'état de chaque nœud.

Cette architecture va être mise en place à partir d'un algorithme que nous avons conçu et qui définit la manière dont les nœuds vont se déplacer. La stratégie de mouvement des nœuds sera complètement distribuée car leurs décisions ne sont pas synchronisées par une horloge globale.

Cet algorithme est totalement asynchrone par les mouvements des nœuds car plusieurs nœuds peuvent se déplacer en parallèle. En plus, pendant le routage d'un message, les nœuds intermédiaires ne sont pas obligés de s'occuper seulement du routage de ce message particulier, mais ils peuvent accomplir d'autres tâches parmi lesquelles participer éventuellement à une autre transmission de message. De ce point de vue, l'algorithme est également asynchrone par la communication entre les nœuds parce qu'en attendant qu'un

message arrive à sa destination, d'autres communications peuvent avoir lieu impliquant des nœuds qui viennent d'acheminer un message.

```
initialiser le nœud n_k dans son point de rdv avec le plus petit numéro ;
for chaque point de rendez-vous de n_k dans l'ordre et en boucle do
  if un nœud est en attente dans ce point de rendez-vous then
    échanger des messages s'il y en a ;
    notify() ; // déclenche le mouvement vers le point de rdv suivant
    if si un message m est reçu then
      if la destination de m est n_k then
        traiter m ;
      else
        conserver m pour routage ;
      end if
    end if
    continuer le mouvement vers le point de rendez-vous suivant ;
  else
    wait() ; // reste en attente de l'autre nœud dans ce point de rdv
  end if
end for
```

FIG. 7.5 – L'algorithme d'ordonnancement du nœud n_k

Les nœuds peuvent choisir de ne pas se déplacer tout le temps, mais seulement quand il s'agit de transmettre un message qui nécessite leur participation dans le routage. Les mouvements des nœuds seront les mêmes pour les différents modèles de trafic dont on parle dans la section 5.1. Ainsi, que le modèle de trafic soit : une seule paire de source-destination (*single flow*), ou plusieurs paires source-destination (*multiflows*), ou plusieurs sources et une destination (dite *concast*), le schéma de mouvement reste le même. L'algorithme d'ordonnancements des mouvements pour un nœud n_k est présenté dans la figure 7.5.

Grâce au mécanisme de points de rendez-vous, les nœuds peuvent utiliser une petite zone de transmission (*small transmission range*). La réduction de la zone de transmission a pour conséquence la conservation de l'énergie et de la bande passante (*bandwidth*) en maintenant en même temps la connectivité du réseau.

7.4 Propriétés du réseau

L'algorithme de la figure 7.5 introduit un ordonnancement pour les mouvements des nœuds mobiles. Cependant, il faudra prouver que cet ordonnancement assure qu'il n'y aura pas de blocage (*deadlock*) dans le réseau. Si cela était le cas, alors les messages pourraient ne pas être acheminés dans un temps borné d'une source à une destination. Pour prouver l'absence de blocage nous nous sommes servis des automates locaux décrits dans la section 7.1. Ces automates, qui représentent les états locaux du système, sont un moyen puissant pour introduire et prouver les propriétés du réseau.

Pour cela, nous introduisons d'abord la notion d'automate produit. Étant donné un ensemble d'automates locaux $\mathcal{A}_k = \{Q_k, \mathcal{T}_k, i_k\}$ $(k \in [1..M])$ leur automate produit est construit comme suit :

- *ses états sont des M-uplet des états des automates locaux* $(q_1, ..., q_M) \in Q_1 \times \ldots \times Q_M$
- *l'état initial est l'état composé de tous les états initiaux des automates locaux* $(i_1, ..., i_M)$
- *il existe une transition entre deux états* $\mathcal{Q} = (q_1, \ldots, q_M)$ *et* $\mathcal{Q}' = (q'_1, \ldots, q'_M)$ *de l'automate produit si et seulement si l'état* $\mathcal{Q}$ *et l'état* $\mathcal{Q}'$ *diffèrent seulement par les états de deux automates locaux* $\mathcal{A}_i$ *et* $\mathcal{A}_j$ *tel que dans* $\mathcal{Q}$, $label(q_i) = label(q_j)$ *et tel que dans les automates* $\mathcal{A}_i$ *et* $\mathcal{A}_j$ *il existe, respectivement, une transition de l'état* q_i *à* q'_i *et de l'état* q_j *à* q'_j. *Cette transition a pour label* $\{label(q_i)\}$.

Cet automate modélise l'évolution de l'état global du système à chaque instant. En effet, en analysant chacun de ses états, on est capable de savoir où se trouve chaque nœud à chaque instant. Une transition dans cet automate représente en effet un changement de position d'au moins deux nœuds dans le réseau.

Dans un premier temps, afin de minimiser les transitions, nous avons choisi de représenter les transitions où seulement deux automates locaux changent d'état. Autrement dit, dans le système, un seul évènement (rencontre dans un point de rendez-vous) est pris en compte à la fois. Pour les automates locaux de la figure 7.4, leur automate produit est représenté dans la figure 7.6.

Dans cet automate sont présentés seulement les états accessibles depuis l'état initial $(1, 1, 2, 4, 3, 5, 7)$. En effet, l'automate produit complet a 576

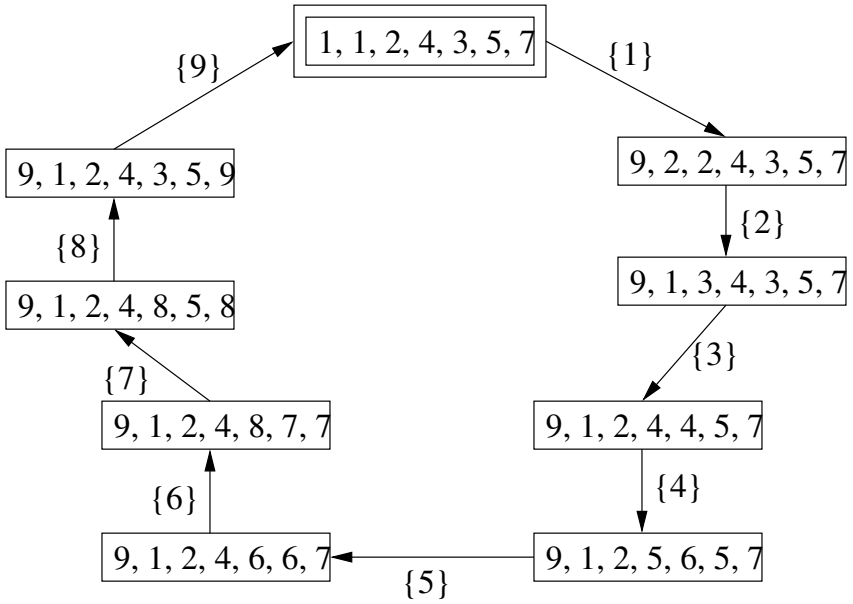


FIG. 7.6 – Automate produit

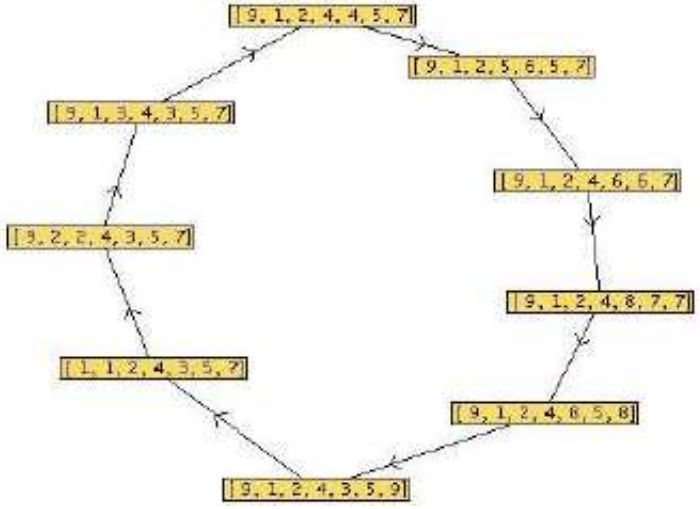


FIG. 7.7 – Automate produit

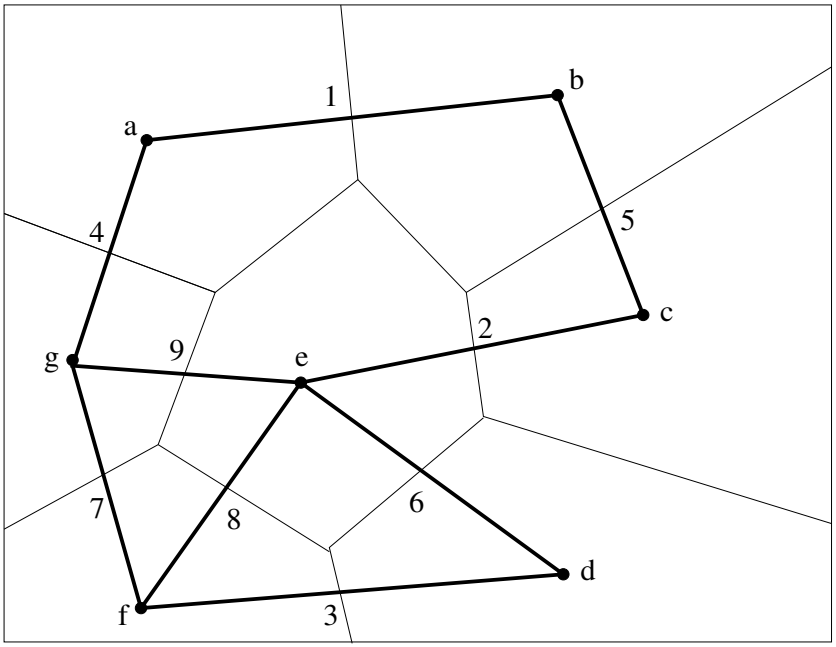


FIG. 7.8 – Autre numérotation du graphe des points de rendez-vous

états mais, dans la figure 7.6, ne sont présentés que les 9 états accessibles depuis cet état initial.

Pour nos tests, nous avons conçu une plate-forme de simulation de réseau ad hoc et qui sera présentée dans le chapitre 9. Cet outil dispose d'une interface graphique qui permet de représenter les automates produits comme illustré dans la figure 7.7. Cet automate est le même avec celui de la figure 7.6, mais il n'affiche pas les labels des transitions.

Cet automate produit est généré à partir de la numérotation des points de rendez-vous, qui comme nous l'avons noté est aléatoire. L'automate de la figure 7.6 est relativement simple, mais, pour une autre numérotation des points de rendez-vous (ou des arêtes du graphe des points de rendez-vous) on peut obtenir un automate produit tout à fait différent de celui de la figure 7.7. Ainsi, une autre numérotation comme celle représenté dans la figure 7.8, aboutit à un automate produit ayant 39 états accessibles depuis son état initial.

L'automate produit pour cette nouvelle numérotation, généré par la plate-forme de test est représenté dans la figure 7.9.

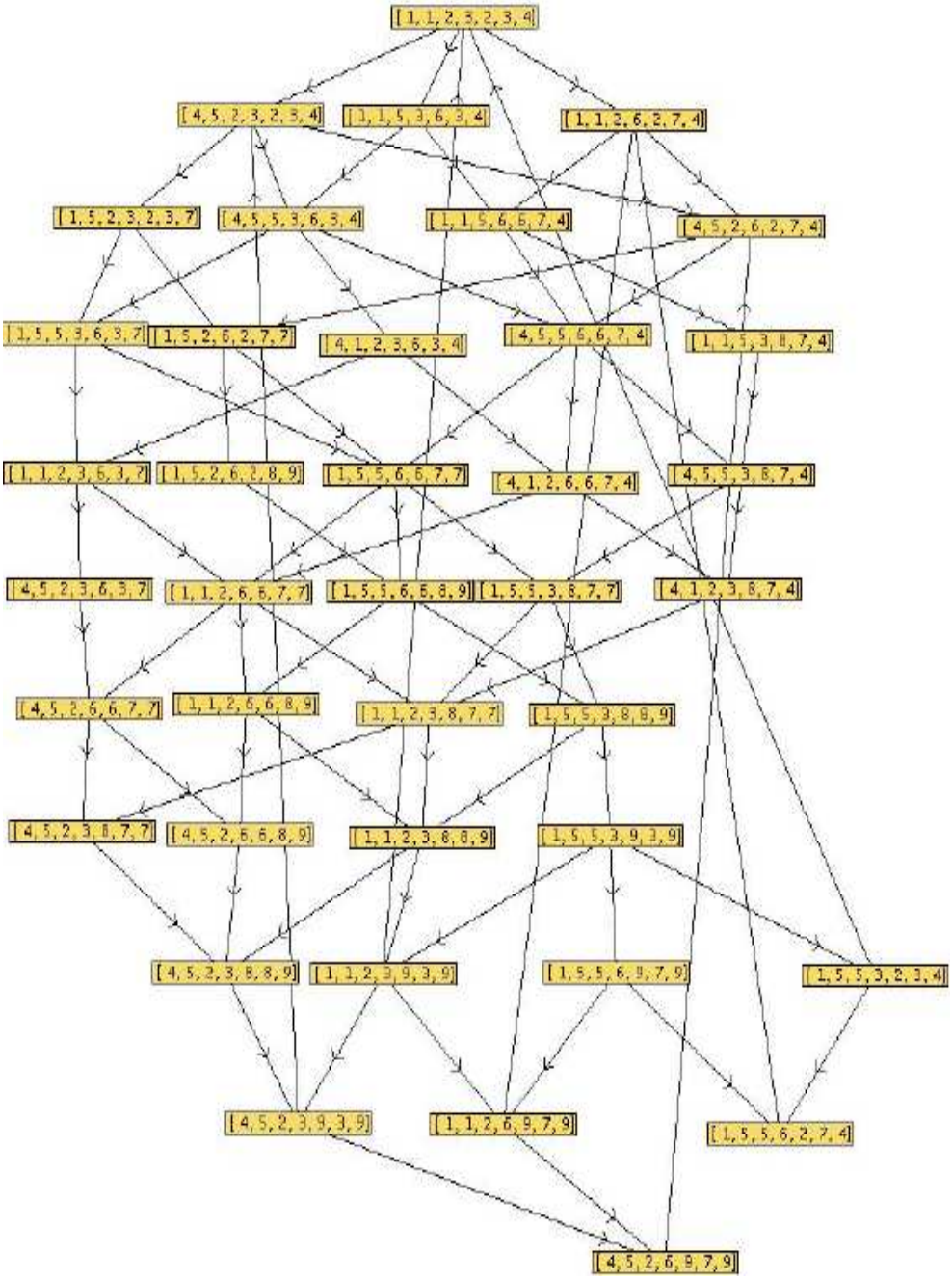


FIG. 7.9 – L'automate produit pour l'autre numérotation

Dans cet automate, l'état initial $(1, 1, 2, 3, 2, 3, 4)$ en haut de la figure, a trois successeurs : $(4, 5, 2, 3, 2, 3, 4)$, $(1, 1, 5, 3, 6, 3, 4)$ et $(1, 1, 2, 6, 2, 7, 4)$ et deux prédécesseurs $(1, 1, 2, 3, 9, 3, 9)$ et $(1, 5, 5, 3, 2, 3, 4)$.

Dans ces automates produits, chaque état a un successeur ce qui prouve que le système n'est jamais bloqué. En plus, quelque soit le chemin (suite d'états) suivi dans ces automates, on peut constater plus ou moins simplement que les nœuds traversent tous leurs points de rendez-vous après un nombre borné de transitions.

Le fait de ne considérer qu'un évènement à la fois n'aura aucun effet sur les propriétés du système. Dans ce cas, l'automate produit aura moins d'états. En effet, en réalité les temps de parcours entre deux points de rendez-vous sont très différents pour les différents nœuds. Néanmoins, l'automate produit étant une abstraction du système, il n'associe pas le temps absolu aux transitions. Ainsi, il est toujours possible de considérer deux transitions sur des points de rendez-vous distincts.

En étudiant l'automate produit nous avons pu nous rendre compte de plusieurs propriétés du système. La proposition de *Non-Blocage global* décrit le système en général tandis que celle de *Non-Blocage local* décrit le comportement des nœuds en particulier.

Proposition 1 (Non-Blocage global) *Quelle que soit la numérotation choisie pour les points de rendez-vous, si l'ordonnancement des déplacements des nœuds est construit en respectant l'algorithme d'ordonnancement de la figure 7.5, le système n'est jamais bloqué.*

Nous allons nous référer (*en italique*) à l'exemple de la figure 7.8 avec son automate produit de la figure 7.9 pour illustrer la preuve de cette propriété.

Preuve Afin de prouver que le système n'est jamais bloqué, il suffit de prouver que chaque état de l'automate produit, atteint à partir de l'état initial, a un successeur.

D'abord, l'état initial de l'automate produit a toujours un successeur. En effet, par la construction de l'automate produit, dans cet état tous les automates locaux sont dans leurs états initiaux et deux d'entre eux sont dans un état de label 1. Ainsi, il existe au moins une transition à partir de l'état initial de l'automate produit de label $\{1\}$.

Maintenant, il faut prouver que, quel que soit l'état de l'automate produit atteint à partir de l'état initial, cet état a un successeur.

Pour ce faire, il suffit de prouver que chaque état atteint à partir de l'état initial est un état initial pour une autre numérotation qui produit le « même » automate (modulo les états initiaux et les labels). Ainsi, cet état aura au moins un successeur. Par récurrence, on prouve que chaque état atteint à partir de l'état initial est un état initial pour une autre numérotation et donc il a au moins un successeur.

Supposons que, dans l'état initial de l'automate produit, on a choisi une transition $\{p\}$ qui implique les automates locaux $\mathcal{A}_i$ et $\mathcal{A}_j$. Ces deux automates locaux sont, par définition, dans des états ayant le même label, qui est d'ailleurs le plus petit label p de leurs états. Dans tous les cas, la transition $\{1\}$ peut être choisie, mais il peut en exister d'autres possibles.

Par exemple, pour la numérotation de la figure 7.8, les automates locaux sont présentés dans la figure 7.10. Ces automates génèrent l'automate produit de la figure 7.9 avec comme état initial $(1, 1, 2, 3, 2, 3, 4)$. *Alors,* $p = 1$, $p = 2$ *ou* $p = 3$.

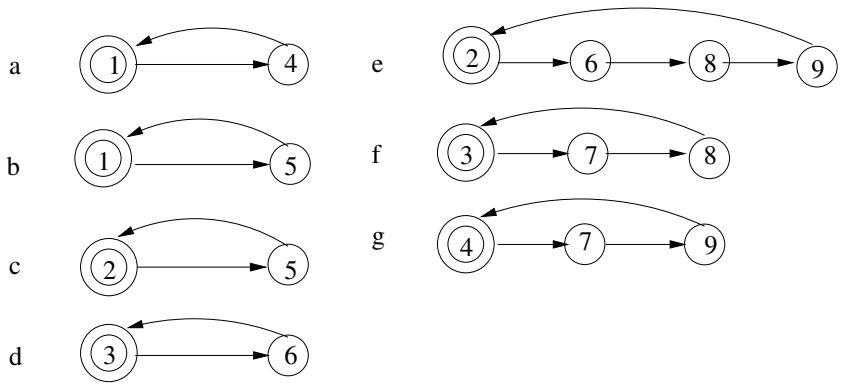


FIG. 7.10 – Automates locaux

Soit N le nombre total des points de rendez-vous et supposons que la renumérotation suivante soit appliquée à tous les points de rendez-vous :

- si l'état q a le label k, $k > p$, alors son nouveau label est $k - 1$;
- l'état de label p prend le label N ;
- sinon le label de l'état ne change pas

Si l'on considère les automates locaux $\mathcal{A}_d$ *et* $\mathcal{A}_f$, *et* $p = 3$, *alors l'état de label* 3 *prend le label* 9. *Les états de label inférieur à* 3 *ne changent pas et*

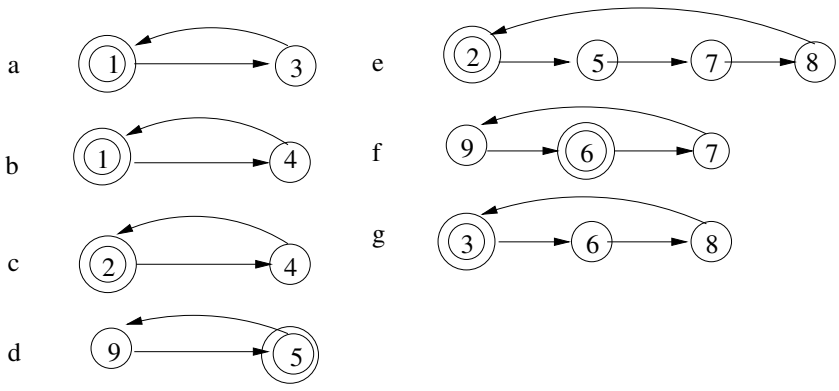


FIG. 7.11 – Automates locaux après la numérotation

ceux de label $k > 3$ prendront le label $k - 1$.

D'abord, considérons les automates locaux différents de $\mathcal{A}_i$ et $\mathcal{A}_j$ après la nouvelle numérotation. Sachant que les automates locaux $\mathcal{A}_i$ et $\mathcal{A}_j$ étaient les seuls ayant un état avec le label p [1] et que la nouvelle numérotation conserve l'ordre des labels des autres états, ces automates sont les mêmes que ceux (automates) qui auraient été obtenus avec cette nouvelle numérotation.

Ainsi, pour notre exemple, les automates locaux $\mathcal{A}_a$, $\mathcal{A}_b$, $\mathcal{A}_c$, $\mathcal{A}_e$ et $\mathcal{A}_g$ ne changent pas (seuls quelques labels changent mais les automates sont les mêmes). Pour notre exemple, les « nouveaux » automates locaux sont représentés dans la figure 7.11.

Maintenant, comparons les automates $\mathcal{A}_i$ et $\mathcal{A}_j$ après l'application de la nouvelle numérotation avec ceux de la numérotation initiale. L'ordre de parcours des états est respecté d'une manière « circulaire », mais l'état initial a changé. En effet, le nouvel état initial dans chacun de ces automates après renumérotation est le successeur de p puisqu'il a le label le plus petit.

Pour notre exemple, comparons les automates locaux $\mathcal{A}_d$ et $\mathcal{A}_f$ de la figure 7.10 avec les automates $\mathcal{A}_d$ et $\mathcal{A}_f$ de la figure 7.11. L'ordre de parcours est respecté sauf que, pour $\mathcal{A}_d$ l'état initial prend le label 5 et pour l'automate $\mathcal{A}_f$, l'état initial prend le label 6.

L'automate produit ayant subi la renumérotation est le même (aux labels

[1] On a supposé que dans un point de rendez-vous, il n'y a que deux nœuds qui se rencontrent. Alors, grâce à l'équivalence état/point de rendez-vous, il n'y a que $\mathcal{A}_i$ et $\mathcal{A}_j$ ayant le label p

près) que l'automate produit que l'on peut obtenir à partir des automates locaux initiaux en leur faisant subir la même renumérotation. La seule chose qui change est l'état initial de l'automate produit.

L'état initial de l'automate produit ayant subi la renumérotation est la renumérotation du successeur, par la transition $\{p\}$, de l'état initial de l'automate produit original. En effet, seuls les états initiaux des automates locaux $\mathcal{A}_i$ et $\mathcal{A}_j$ ont changé (ce sont désormais les successeurs de p) et, si on suit la transition $\{p\}$ dans l'automate produit initial (ou dans l'automate produit des automates locaux de la figure 7.10), tous les automates, différents de $\mathcal{A}_i$ et $\mathcal{A}_j$, restent dans leurs états initiaux et, $\mathcal{A}_i$ et $\mathcal{A}_j$, avancent dans leurs états avec le plus petit label dans la nouvelle numérotation. Ces états sont, par construction, leurs nouveaux états initiaux. Ainsi, après la transition $\{p\}$ tous les automates sont dans l'état initial défini par la nouvelle numérotation.

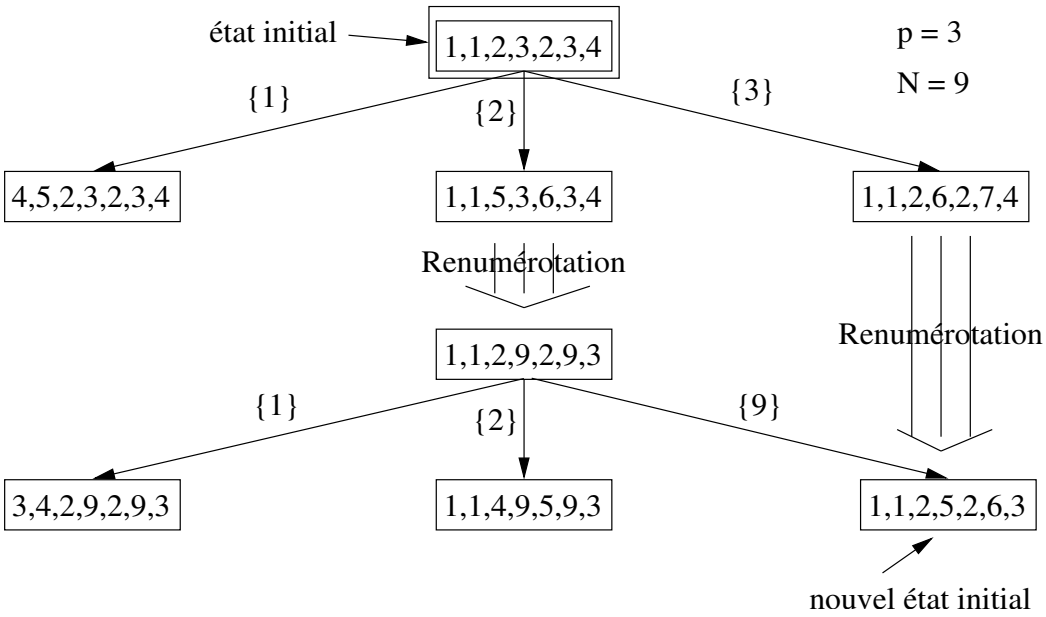


FIG. 7.12 – Automates produits

Pour notre exemple, dans la figure 7.12 en haut est représenté une partie de l'automate produit avec son état initial et ses 3 successeurs. Cet automate produit est généré à partir des automates locaux de la figure 7.10.
La partie en bas de la figure 7.12, représente une partie de l'automate produit qui peut être généré :

- *soit par la renumérotation (avec $p = 3$ et $N = 9$) de l'automate produit d'en haut ;*
- *soit par les automates locaux de la figure 7.11, lesquels sont obtenus en appliquant la renumérotation sur les automates locaux de la figure 7.10.*

Dans le nouvel automate, ainsi obtenu, l'état initial change. Il peut être obtenu après une renumérotation de la transition par $\{p\}$ de l'état initial de l'automate d'en haut.

□

La propriété que l'on vient de prouver est une propriété globale. Elle assure qu'il y a toujours au moins un nœud qui se déplace dans le réseau. En revanche elle ne prouve pas que tous les nœuds se déplacent au bout d'un temps borné. Cette propriété du réseau où l'on applique l'ordonnancement des mouvements selon l'algorithme de la figure 7.5 est garantie par la proposition suivante :

Proposition 2 (Non-Blocage local) *Quelle que soit la suite de transitions dans l'automate produit, tous les automates locaux changent d'état au bout d'un nombre borné de transitions dans l'automate produit.*

Preuve D'abord, comme l'automate produit est fini et qu'il ne présente pas de blocage, chaque suite de transitions dans cet automate amène toujours dans un état déjà visité après un nombre fini (borné) de transitions. Dans ce cycle (non vide), les états de tous les automates locaux sont égaux au début et à la fin du cycle. Ainsi, soit l'automate local reste dans le même état, soit il a parcouru tous ses états au moins une fois.

Maintenant, prouvons que si l'automate local a plus d'un état, il changera d'état et ainsi il parcourra tous ses états. Supposons le contraire, c'est-à-dire qu'il n'a pas changé d'état. Alors tous les autres automates avec lesquels il a des points de rendez-vous en commun ne changent pas d'état. En effet, sinon ils parcourraient tous leurs états et en particulier l'état en question ce qui est contradictoire avec la supposition. Comme tous les nœuds sont reliés par un chemin (ou suite) de points de rendez-vous (il n'y a pas de nœud isolé), alors par transition aucun nœud ne se déplacera, ce qui est impossible dans un cycle non vide puisque au moins deux automates doivent changer d'état. □

De cette propriété on peut déduire que quel que soit l'état courant du réseau atteint à partir de l'état initial, chaque nœud atteindra chacun de

ses points de rendez-vous après un nombre borné de transitions dans l'automate produit. Ainsi, chaque chemin (suite) de points de rendez-vous sera parcouru(e) dans un temps borné. Cela veut dire que les mouvements des nœuds, modélisés par les transitions dans les automates locaux, peuvent aider à assurer une communication de bout en bout dans un temps borné.

Remarque : Une autre approche pour la preuve des propriétés 1 et 2 est donnée par El Haddad et Haddad [36] utilisant des réseaux de Petri. Le travail de El Haddad et Haddad est basé sur notre premier algorithme de la figure 7.5 et essaye d'introduire un schéma auto-stabilisant ordonnançant les visites des points de rendez-vous.

Les propriétés 1 et 2 permettent d'établir le corollaire suivant :

Corollaire 1 *Chaque message, transmis par un nœud source vers un nœud destination du réseau, peut atteindre la destination dans un temps borné s'il existe un chemin de points de rendez-vous qui les connecte.*[2]

À partir des démonstrations précédentes, on peut aussi déduire une propriété sur les cycles dans l'automate produit. Mais d'abord nous donnons la définition suivante sur ce que nous appelons les *cycles élémentaires* :

Definition 5 *Nous appelons cycle élémentaire, une suite d'états de l'automate produit qui à partir d'un état initial, aboutit après un nombre fini de transitions dans le même état sans jamais passer deux fois par le même état.*

Proposition 3 *La longueur des cycles élémentaires dans l'automate produit est un multiple du nombre de points de rendez-vous du réseau.*

Preuve La preuve de la proposition 3 est basée sur le fait que si, dans un cycle, un nœud ne se déplace pas, alors tous les autres nœuds ne se déplaceront pas par transitivité. De la même manière, si dans un cycle un nœud parcourt tous ses points de rendez-vous n fois, alors les nœuds avec lesquels il partage des points de rendez-vous doivent aussi parcourir n fois leurs points de rendez-vous. Grâce à la connectivité du réseau on obtient que dans un cycle, tous les nœuds parcourent tous leurs points de rendez-vous le même nombre de fois, ce qui prouve la proposition. □

[2]Ceci n'est vrai que dans la mesure où, lorsqu'un nœud n_i est au point de rendez-vous avec un nœud n_j, il émet tous les messages qui doivent passer par n_j.

7.5 Généralisation de l'algorithme

Jusqu'à maintenant, nous avons considéré que dans un point de rendez-vous il n'y avait que deux nœuds qui se rencontraient. Pourtant en réalité plus de 2 nœuds peuvent se rencontrer dans un point de rendez-vous.

Il est facile de généraliser l'algorithme de la figure 7.5 s'il y a plus de deux nœuds qui se rencontrent dans un point de rendez-vous. La figure 7.13 représente un tel réseau, où dans le point de rendez-vous 3, il y a trois nœuds qui se rencontrent, les nœuds b, c et d et dans le point de rendez-vous 5 les nœuds c, e et f.

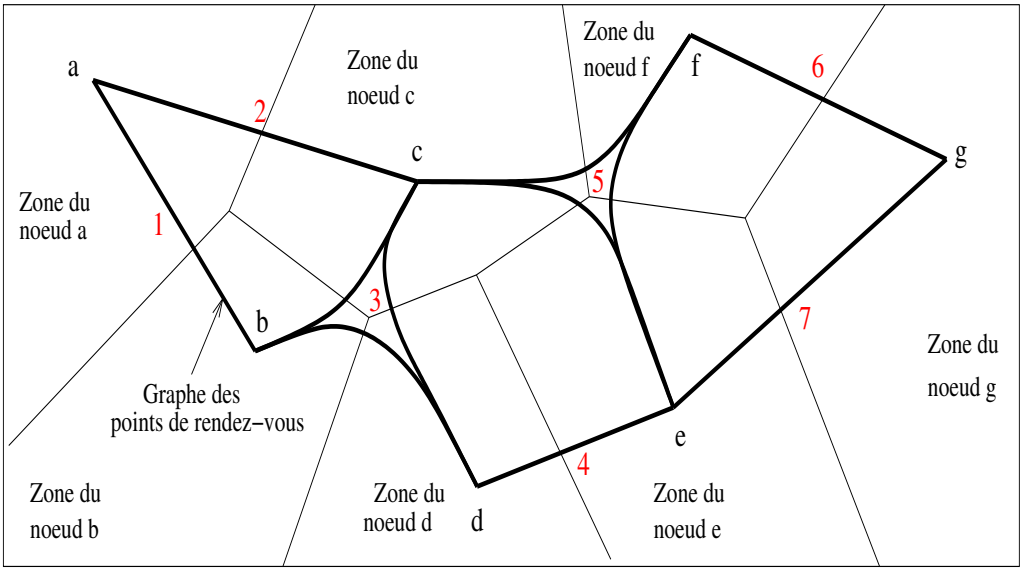


FIG. 7.13 – Le graphe des points de rendez-vous

Dans ce cas, les propriétés et leurs preuves restent valables. En effet, pour la preuve de la proposition 1, il suffit de remplacer les deux automates $\mathcal{A}_i$ et $\mathcal{A}_j$ par l'ensemble des automates locaux $\mathcal{A}_{i_1}$, $\mathcal{A}_{i_2}$,...,$\mathcal{A}_{i_k}$ qui se trouvent dans le même point de rendez-vous pour que celle-ci reste valable.

Pour le cas où plus de deux nœuds se rencontrent dans un point de rendez-vous l'algorithme d'ordonnancement pour un nœud n_k est présenté dans la figure 7.14. Selon cet algorithme, si par exemple, trois nœuds se rencontrent dans un point de rendez-vous, alors il faut que les trois nœuds soient présents pour qu'il pussent communiquer. Pour vérifier cela ils échangent des messages

de contrôle. Une fois que tous les nœuds sont présents, il peuvent échanger les messages concernant le routage.

```
initialiser le nœud n_k dans son point de rendez-vous avec le plus petit
numéro ;
for chaque point de rendez-vous de n_k dans l'ordre et en boucle do
  while les autres nœuds ne sont pas arrivés dans ce point de rdv do
    échanger des messages de contrôle jusqu'à ce que tous les nœuds ar-
    rivent ;
    wait() ; // reste en attente des autres nœuds dans ce point de rdv
  end while
  échanger des messages s'il y en a ;
  notify() ; // déclenche le mouvement vers le point de rdv suivant
  if si un message m est reçu then
    if la destination de m est n_k then
      traiter m ;
    else
      conserver m pour routage ;
    end if
  end if
  continuer le mouvement vers le point de rendez-vous suivant ;
end for
```

FIG. 7.14 – L'algorithme d'ordonnancement généralisé du nœud n_k

Bien que le nombre de nœuds qui se rencontrent dans un point de rendez-vous ne change rien en ce qui concerne les propriétés, nous allons considérer dans la suite qu'il n'y a que deux nœuds (un seul couple) qui se rencontrent dans un point de rendez-vous.

7.6 Le routage dans le réseau

Les sections précédentes de ce chapitre présentent l'ordonnancement des mouvements des nœuds dans un réseau ad hoc. Cet ordonnancement assure que les nœuds se rencontrent et qu'il est possible d'acheminer un message d'un nœud quelconque du réseau à un autre dans un temps borné. Pour

établir une communication dans le réseau, il faut mettre en place un schéma de routage. Pour cela, il faut d'abord définir la manière dont sont construites les tables de routage. Grâce à celles-ci, chaque nœud acheminera de façon « appropriée » (en respectant l'algorithme 7.5) les messages dans le réseau.

Afin d'être aussi précis que possible, la table de routage de chaque nœud doit contenir, pour chaque point de rendez-vous, une entrée pour chaque nœud destination. En effet, le chemin le plus court vers une destination dépend de l'endroit où se trouve le nœud quand il prend la décision du routage d'un message. Ainsi, une entrée de la table de routage représente le point de rendez-vous dans lequel le nœud doit transmettre le message. Ce point de rendez-vous fait partie de la route que le message doit suivre pour atteindre la destination.

Pour calculer les tables de routage nous avons considéré un graphe que l'on appelle *graphe des déplacements*. Ce graphe est construit à partir de l'ordonnancement des mouvements. Il contient des arcs qui représentent les mouvements entre les points de rendez-vous et d'autres arcs qui représentent les temps d'attente[3] dans un point de rendez-vous. En plus, chaque arc de mouvement est pondéré pour prendre en compte les tâches accomplies par chaque nœud durant ces mouvements.

Puisque tous les arcs sont connus au démarrage, il est possible d'utiliser l'algorithme de Dijkstra pour trouver le chemin le plus court (en nombre de sauts où en temps). Les algorithmes du plus court chemin trouvent des routes entre des paires de points de rendez-vous de nœuds distincts. Ensuite, pour générer les tables de routage, pour chaque point de rendez-vous initial et chaque nœud destination, on maintient seulement le chemin le plus court vers un point de rendez-vous du nœud destination. En effet, on s'intéresse au chemin le plus court vers un nœud et non vers un point de rendez-vous de ce nœud.

En ce qui concerne le graphe des déplacements, sa pondération est arbitraire. Elle pourra être modifiée pour favoriser certaines routes aux dépens d'autres. Par exemple, si on veut favoriser les déplacements par rapport aux communications, on pourra mettre des poids importants sur les arcs correspondants aux temps d'attente dans les points de rendez-vous et des poids faibles aux arcs qui correspondent aux mouvements. Si par contre on veut minimiser l'énergie consommée par les nœuds mobiles, alors on fera le contraire.

Pour le graphe des points de rendez-vous de la figure 7.15 son graphe des

[3]Les temps de communications sont negligés par rapport aux temps d'attente.

déplacements est représenté dans la figure 7.16. Comme on peut voir dans cette figure, dans les points de rendez-vous nous avons ajouté des arcs pour prendre en compte les attentes dans les points de rendez-vous.

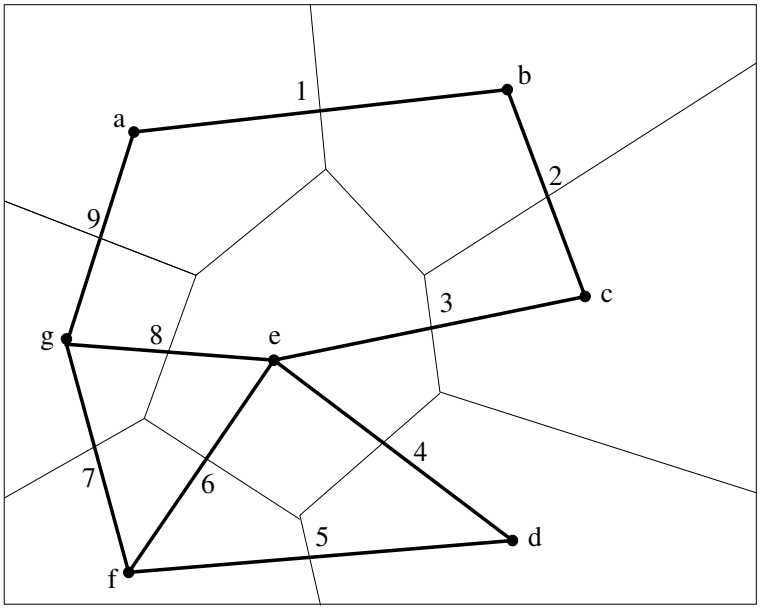


FIG. 7.15 – Graphe numéroté

Par exemple, si toutes les arêtes du graphe de la figure 7.16 ont le même poids, une partie de la table de routage générée pour le nœud de la zone a afin d'atteindre le nœud de la zone e est présentée dans la table 7.1.

Selon cette table, si le nœud a se trouve dans son point de rendez-vous 1 et veut envoyer un message au nœud e, alors il peut le faire *via* le point de rendez-vous 1 ou le point de rendez-vous 9 [4].

La table de routage complète pour le nœud de la zone a est présentée dans la table 7.2. Cette table contient seulement la première entrée de la table 7.1 parce que c'est l'un des plus courts chemins du point de rendez-vous 1 du nœud a vers le nœud e. Cette table contient une seule entrée pour chaque couple *(point de rendez-vous, destination)*, celle dont la longueur est optimale.

[4]Les distances sont calculées dans le graphe orienté de la figure 7.16

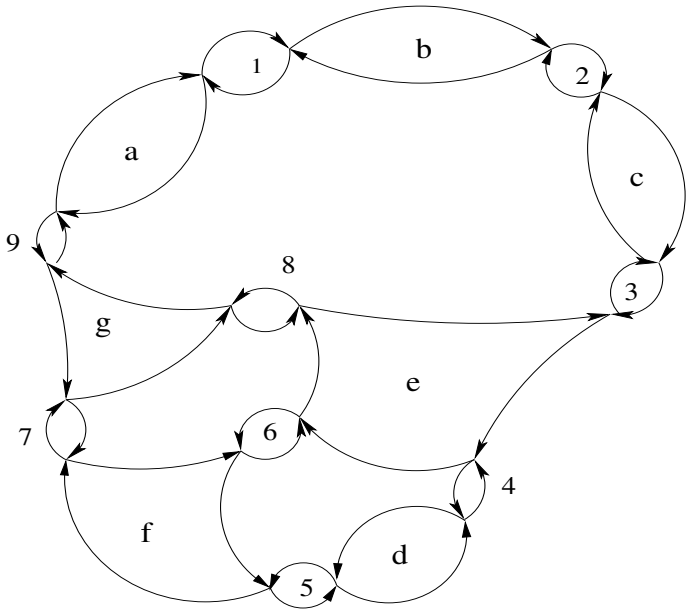


FIG. 7.16 – Le graphe des déplacements

Source	Dest.	Rdv	Longueur du chemin (en nb. de sauts+nb. déplacements)
a.1	e.3	a.1	5
a.1	e.4	a.1	6
a.1	e.6	a.9	6
a.1	e.8	a.9	5

TAB. 7.1 – Partie de la table de routage du nœud a

Source	Dest.	Rdv	Longueur du chemin (en nb. de sauts+nb. déplacements)
a.1	b	a.1	1
a.1	c	a.1	3
a.1	d	a.9	7
a.1	e	a.1	5
a.1	f	a.9	4
a.1	g	a.9	2
a.9	b	a.1	2
a.9	c	a.1	4
a.9	d	a.9	5
a.9	e	a.9	4
a.9	f	a.9	3
a.9	g	a.9	1

TAB. 7.2 – Table de routage du nœud a

Le graphe des déplacements est un graphe orienté où le poids de chaque arc pourra signifier soit la longueur soit le temps de parcours. Mais d'un point de vue général, le poids des arcs peut être un ensemble de valeurs chacune représentant les différentes tâches qu'un nœud exécute en se déplacant dans sa zone. Intuitivement, un nœud mobile tentera de choisir le chemin le plus court vers la destination. En utilisant ce graphe on peut obtenir le chemin le plus court entre la position courante du nœud et la destination du message. Comme métrique d'optimisation, on peut choisir une des valeurs de l'ensemble qui servent comme poids pour les arcs.

7.7 Résumé

Ce chapitre a présenté la modélisation faite pour étudier notre architecture. Nous avons utilisé un graphe que l'on a appelé *graphe de points de rendez-vous*. Dans ce graphe, chaque sommet représente un nœud et chaque arête un point de rendez-vous. En effet, ce point de rendez-vous est un lien de communication entre deux nœuds. Un premier algorithme introduit un ordonnancement des mouvements des nœuds. Nous avons prouvé que cet or-

donnancement basé sur le mécanisme de rendez-vous assure le non-blocage du système. L'utilisation des points de rendez-vous peut permettre la réduction de la zone de transmission des nœuds. Ainsi, on peut réduire aussi la consommation de l'énergie et de la bande passante. Ce chapitre, présente aussi la façon dont on construit les tables de routage pour chaque nœud. Les tables de routage sont calculées à partir du *graphe des déplacements*. Une entrée de la table de routage donne le chemin le plus court vers un point de rendez-vous du nœud destination. Ces tables de routage évitent la redondance des chemins.

Chapitre 8

Améliorations de l'algorithme

Introduction

Nous sommes maintenant capables de construire un ordonnancement des mouvements des nœuds d'un réseau ad hoc basé sur la division du plan en zones. Cet ordonnancement est basé sur une numérotation aléatoire des points de rendez-vous.

Pourtant, nous avons constaté que toutes les numérotations possibles ne produisent pas des ordonnancements équivalents. Certains ordonnancements sont meilleurs que d'autres pour réduire les déplacements, d'autres pour réduire la longueur des routes ou pour augmenter le degré du parallélisme (plusieurs tâches en parallèle) dans le réseau.

Par ailleurs, notre architecture doit faire face aux défaillances du système. L'algorithme doit donc prendre en compte les pannes qui peuvent arriver, ce qui n'est pas le cas pour l'instant.

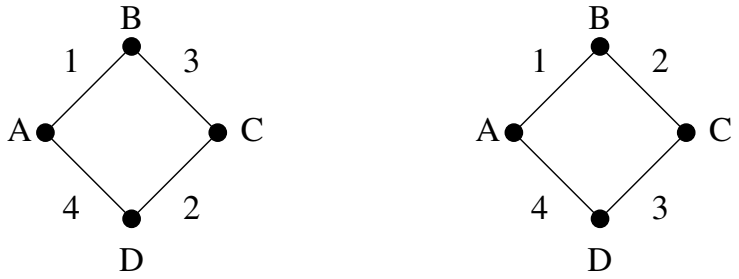
Ce chapitre est composé de quatre parties. La première partie (section 8.1) présente les améliorations qui sont faites pour augmenter le degré du parallélisme dans le réseau. La deuxième partie (section 8.2) présente une amélioration de la numérotation des points de rendez-vous afin de réduire les déplacements inutiles en réduisant ainsi la consommation de l'énergie. La troisième partie (section 8.3) présente les améliorations faites pour rendre l'algorithme initial tolérant aux pannes. La section 8.4 présente une comparaison avec un algorithme auto-stabilisant. Enfin, la section 8.5 fait un récapitulatif des propriétés de notre algorithme et un résume conclut ce chapitre.

8.1 Amélioration du parallélisme

Cette section présente les amélioration du parallélisme. Elle définit d'abord la notion de parallélisme dans le réseau et présente l'algorithme de Vizing utilisé pour coloré les arêtes du graphe des points de rendez-vous. Elle introduit la notion de l'automate produit complet et une séries de propriétés sur le parrallélisme.

8.1.1 Améliorations des performances utilisant la coloration

Les ordonnancements qui permettent à deux (ou plus) couples de nœuds de se rencontrer et de se déplacer en même temps (c'est-à-dire en parallèle) vers leurs points de rendez-vous respectifs, semblent plus adaptés pour optimiser (dans ce cas minimiser) la « latence » globale des communications par rapport aux ordonnancements qui forcent les nœuds à se déplacer séquentiellement (les uns après les autres). Prenons l'exemple de la figure 6.3 et son graphe de points de rendez-vous numéroté, représenté ici dans la figure 8.1(a). Si on utilise une autre numérotation, comme dans la figure 8.1(b), alors l'automate produit présenté dans la figure 8.2(b) sera différent.

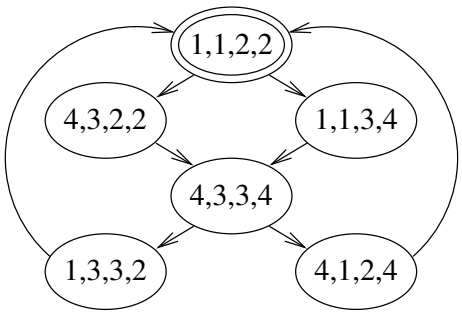


(a) Numérotation initiale

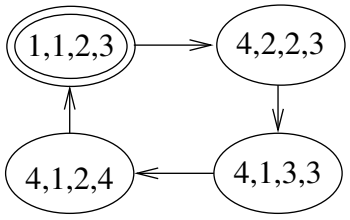
(b) Autre numérotation

FIG. 8.1 – Exemple de graphes de points de rendez-vous

L'automate de la figure 8.2(b) implique des mouvements séquentiels. En effet, dans cet automate, pour chaque état il n'y a qu'une pair de nœuds dans le même point de rendez-vous, tandis que les autres nœuds sont en attente dans leurs points de rendez-vous.



(a) Automate produit pour la numérotation de la figure 8.1(a)



(b) Automate produit pour la numérotation de la figure 8.1(b)

FIG. 8.2 – Exemple d'automates

Pour réduire la latence des communications, l'idée est donc d'introduire du parallélisme dans le système. Ce parallélisme correspond au mouvement de plus d'un couple de nœuds en parallèle. Pour définir le parallélisme dans le réseau nous introduisons la définition suivante :

Definition 6 *Il y aura parallélisme dans le réseau si et seulement si au moins deux couples de nœuds peuvent se rencontrer en même temps (un point de rendez-vous pour chaque couple).*

Pour introduire le parallélisme dans le réseau, nous avons utilisé une heuristique basée sur la coloration des arêtes du graphe. Cette heuristique consiste à colorer les arêtes du *graphe des points de rendez-vous* avec un nombre minimum de couleurs, à condition que deux arêtes avec la même couleur ne partagent pas le même sommet. L'heuristique propose que : **les arêtes[1] ayant la même couleur, correspondent à des rencontres qui se produisent de manière simultanée.**

Plus précisément, la coloration du graphe des points de rendez-vous est définie comme suit :

Definition 7 (Coloration) *Si $G = (V, E)$ est le graphe des points de rendez-vous, alors une coloration des arêtes de G est une fonction $\Phi : E \rightarrow [1..\phi]$, telle que si $\Phi(e_1) = \Phi(e_2)$ alors e_1 et e_2 n'ont pas de sommet en commun.*

[1]Une arête équivaut à un point de rendez-vous.

Definition 8 (Nombre chromatique) *Si $G = (V, E)$ est le graphe des points de rendez-vous coloré en respectant la définition 7, alors le nombre chromatique de G est la valeur minimum acceptée pour ϕ.*

Le problème de la coloration des arêtes d'un graphe est un problème NP-complet [38]. Cependant, la démonstration constructive du théorème de Vizing [68] permet de colorer le graphe en temps polynomial avec au plus le nombre minimum de couleur plus une.

8.1.2 Théorème et algorithme de Vizing

Cette section présente le théorème de Vizing et l'idée de sa preuve faite en utilisant un algorithme de Edsger W. Dijkstra et Josyula R. Rao [28]. Le théorème de Vizing sur la coloration des arêtes d'un graphe non-orienté est le suivant :

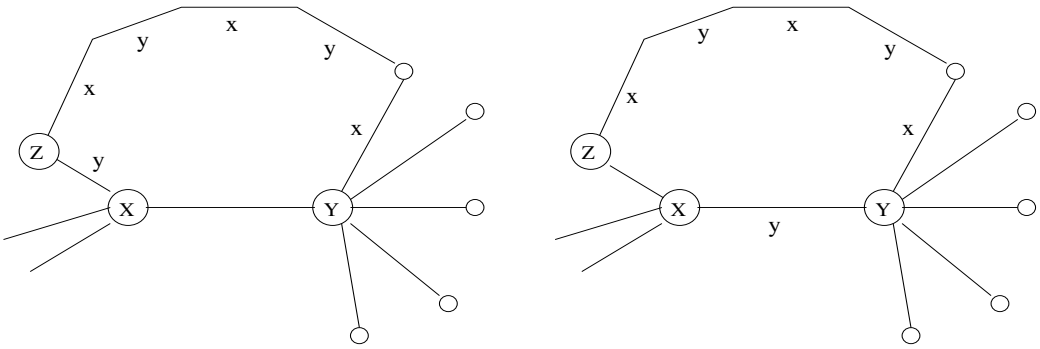
Theoreme 1 (Théorème de Vizing) *Soit n le nombre maximal des arêtes d'un sommet d'un graphe non-orienté, sans auto-cycles et sans arêtes multiples entre deux sommets. Alors $n+1$ couleurs suffisent pour faire une coloration des arêtes, telle que les arêtes adjacentes aient des couleurs différentes.*

Une preuve de ce théorème peut être faite en utilisant un algorithme de coloration. Cet algorithme commence par les sommets du graphe sans arêtes et ajoute les arêtes une par une. À chaque étape, l'algorithme construit une coloration pour le graphe avec la nouvelle arête qui vient d'être ajoutée à partir de la coloration précédente. Si cela est nécessaire, pendant une étape, il est possible de modifier les couleurs de certaines arêtes pour préserver une coloration valide.

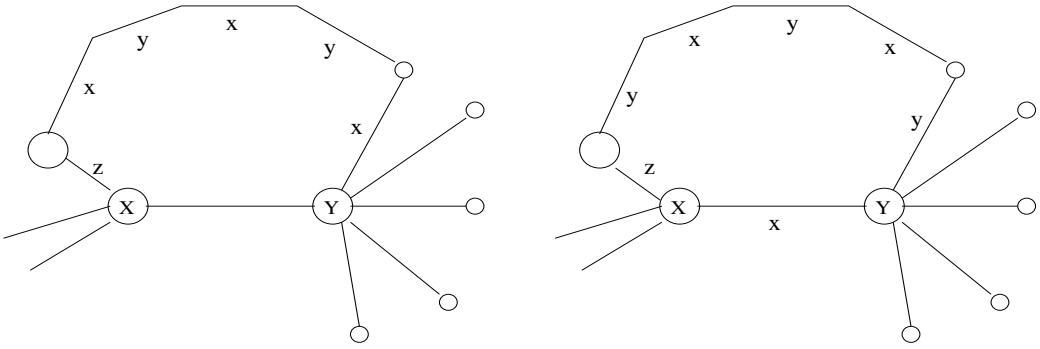
Une étape commence avec un graphe coloré où on ajoute une arête non colorée. Le fait que le nombre de couleurs disponibles dépasse le nombre maximum d'arêtes d'un sommet (on utilise $n+1$ couleurs), assure que chaque sommet a toujours une couleur dite « libre », c'est-à-dire une couleur qui n'a pas encore été attribuée aux arêtes déjà colorées de ce sommet.

L'existence des couleurs « libres » permet de donner à une arête XY non colorée, une couleur x telle qu'il y aura au plus un conflit de couleurs, par exemple dans le sommet X de cette arête. Si cela est le cas, il est possible de résoudre le conflit en décolorant l'autre arête de couleur x sortant du sommet X.

Pour résoudre les conflits de couleurs dans les sommets, l'algorithme considère, entre deux sommets X et Y, des chemins commençant par Y, différents de l'arête XY, dont la coloration satisfait la grammaire $(xy)*$. Selon Dijkstra et Rao [28], ces conflits peuvent être repoussés jusqu'à ce qu'ils disparaissent en inversant les couleurs dans le chemin « alterné » $x-y$ commençant par Y ou en supprimant la couleur de l'arête qui cause conflit et en relançant l'algorithme.



(a) Décoloration



(b) Coloration en inversant les couleurs

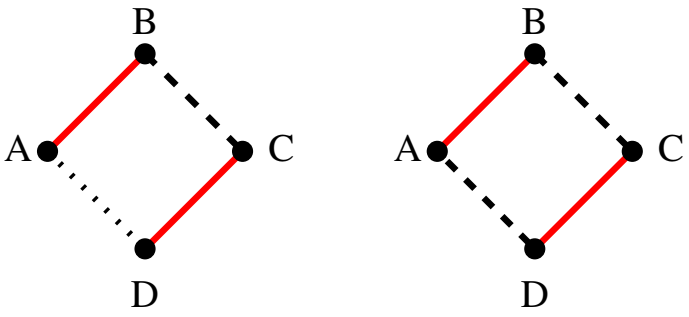
FIG. 8.3 – Coloration selon l'algorithme de Vizing

La figure 8.3 présente deux cas de coloration selon l'algorithme de Vizing. Dans la figure 8.3(a), la couleur y est donnée à l'arête XY. Pour résoudre le conflit de couleur du sommet X, on décolore l'arête XZ qui avait été

colorée auparavant avec la couleur y. Pour le deuxième cas, représenté dans la figure 8.3(b), la couleur x est donnée à l'arête XY et les couleurs sont inversées sur le chemin $(xy)*$.

8.1.3 Coloration et propriétés

La coloration de la figure 8.4(a) est obtenue en appliquant l'algorithme de Vizing sur le graphe de la figure 8.1(a). Trois couleurs ont été utilisées pour la coloration. Les arêtes présentées en gras $(A, B), (C, D)$, dans la figure 8.4(a) ont la même couleur, tandis que les deux autres arêtes, $(A, D), (B, C)$ ont deux autres couleurs différentes. Cette coloration n'est pas la meilleure possible car, ce graphe peut être coloré avec deux couleurs. Ainsi, les arêtes $(A, B), (C, D)$, peuvent avoir la même couleur et les arêtes $(A, D), (B, C)$ une deuxième couleur comme il est montré dans la figure 8.4(b). Ainsi, son nombre chromatique est 2.



(a) Exemple de coloration du graphe

(b) Graphe coloré avec 2 couleurs

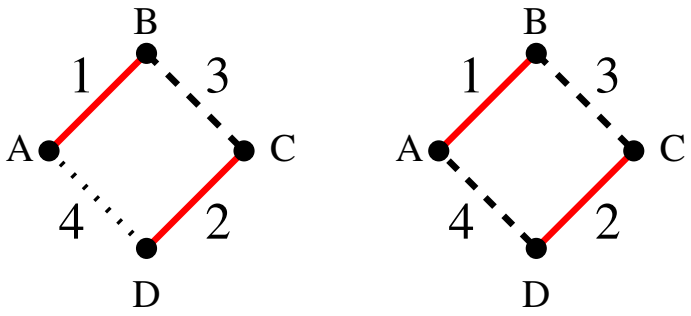
FIG. 8.4 – Colorations du graphe

Alors, la numérotation produite par la coloration est définie comme suit :

Definition 9 *Étant donnée une coloration des arêtes du graphe de points de rendez-vous, la numérotation de ces arêtes se fait de manière à ce que celles qui ont la même couleur aient des numéros consécutifs.*

Le choix de l'ordre des couleurs est aléatoire. Pour le graphe coloré de la figure 8.4(a) on numérote d'abord les arêtes en gras, qui ont la même couleur,

avec 1 et 2. Ensuite les arêtes en pointillés gras avec 3 et puis celles avec pointillés 4. Le résultat est le graphe de la figure 8.5(a). La même technique peut être appliquée pour numéroter le graphe coloré avec deux couleurs de la figure 8.5(b). Le résultat est le graphe de la figure 8.5(b).



(a) Le graphe de la figure 8.4(a) coloré

(b) Graphe coloré et numéroté

FIG. 8.5 – Colorations du graphe

La coloration du graphe a produit une numérotation équivalente à celle de la figure 8.1(a) qui est la meilleure numérotation du graphe même si la coloration n'est pas la meilleure possible. Cette nouvelle sorte de numérotation influencera directement le nouvel ordonnancement des mouvements des nœuds. Cela veut dire que la communication dans le réseau en appliquant l'algorithme de la figure 7.5 basé sur la nouvelle numérotation, va changer.

Maintenant il reste à prouver que si le graphe est coloré avec un nombre minimum de couleurs, alors la numérotation obtenue va produire un « meilleur ordonnancement » relativement au parallélisme.

Pour mesurer le parallélisme dans le réseau, on utilise *l'automate produit complet*. Cet automate est défini comme suit :

Definition 10 (Automate produit complet) *On appelle automate produit complet, un automate produit contenant toutes les transitions valides d'un état à un autre.*

Comparé à l'automate produit de la section 7.4, ce nouvel automate a, en plus, toutes les transitions valides d'un état à un autre. Par exemple

l'automate produit complet pour la numérotation de la figure 8.1(a) (ou des figures 8.5(a) et 8.5(b)) est celui représenté dans la figure 8.6.

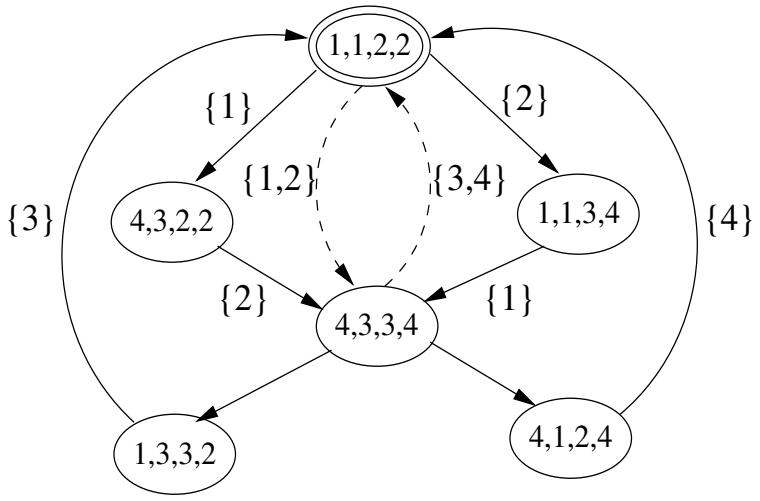


FIG. 8.6 – Automate produit complet

Dans cet automate il y a deux transitions (en lignes pointillés) où les nœuds peuvent se rencontrer et se déplacer en parallèle. Afin de faciliter la notation pour cet automate, les transitions auront comme label, le label du point de rendez-vous où les nœuds se rencontrent. Par exemple, pour l'automate de la figure 8.6, la transition de l'état $(1, 1, 2, 2)$ à l'état $(4, 3, 3, 4)$ aura pour label $\{1, 2\}$.

Il s'agit maintenant de trouver le meilleur ordonnancement possible qui prend en compte le parallélisme. Pour cela, nous commençons par donner notre définition d'un meilleur ordonnancement :

Definition 11 *On appelle meilleur ordonnancement pour le parallélisme, un ordonnancement tel que son automate produit complet contient un cycle dont la longueur est minimale vis à vis de tous les ordonnancements possibles.*

Intuitivement, si la longueur du cycle est minimale, alors chaque nœud parcourt tous ses points de rendez-vous après un nombre minimum de sauts. Cela minimise les communications et la consommation de l'énergie. La proposition suivante définit la longueur du cycle minimal par rapport à la coloration des arêtes du graphe des points de rendez-vous. Rappelons ici qu'une numérotation est représentée par une fonction σ comme dans le chapitre 7.

Proposition 4 *Étant donné un graphe G, la longueur du cycle minimal d'un « meilleur ordonnancement » est égale au nombre chromatique de G.*

Preuve Pour prouver cette proposition, il faut dans un premier temps prouver qu'il existe dans l'automate produit complet un cycle d'une longueur égale au nombre chromatique du graphe. Ensuite, il faut prouver que ce cycle est minimal.

D'abord, supposons que le nombre chromatique de G est ϕ_{min}. Alors il existe une coloration des arêtes $\Phi_{min} : E \rightarrow [1..\phi_{min}]$. Si k_i est le nombre des arêtes de couleur i, alors on note $e^i_1, \ldots, e^i_{k_i}$ les arêtes de couleur i. À partir de cette coloration on définit la suite (chemin) suivante de transitions dans l'automate produit complet :
$\{e^1_1, \ldots, e^1_{k_1}\}, \ldots, \{e^{\phi_{min}}_1, \ldots, e^{\phi_{min}}_{k_{\phi_{min}}}\}$ telle que $\forall e \in E$ si $\Phi_{min}(e) = i$ [2] alors $\exists j \in [1..k_i], e = e^i_j$. Cela veut dire qu'il existe une suite de transitions dans l'automate produit complet où les arêtes dans chaque transition $\{e^i_1, \ldots, e^i_{k_i}\}$, ont la même couleur i $(0 < i \leq \phi_{min})$. Dans une telle transition les rencontres entre les nœuds peuvent avoir lieu en même temps car deux couples ne font pas intervenir les mêmes nœuds. Cette suite est un cycle parce qu'elle contient tous les points de rendez-vous [3]. En effet, dans cette suite se trouvent toutes les couleurs de 1 à ϕ_{min}, donc toutes les arêtes du graphe G.
Ce cycle est un cycle valide dans l'automate produit complet pour la numérotation σ car $\sigma(e_1) < \sigma(e_2)$ si $\Phi_{min}(e_1) \leq \Phi_{min}(e_2)$ (par la définition 9). Il est un cycle car, par définition, il est passé par tous ses points de rendez-vous, son état final est égal à son état initial et chaque arête de G n'y apparaît qu'une fois. Il est valide puisqu'il respecte l'ordre des transitions des automates locaux. Il est évident de construire une telle numérotation σ grâce à la définition 9.

Pour le graphe de la figure 8.5(b) son automate produit complet est celui de la figure 8.6. Si dans cet automate, on considère le cycle $(1,1,2,2)$, $(4,3,3,4)$, *les transitions qui se produisent dans ce cycle sont :* $\{1,2\}, \{3,4\}$. *Dans la figure 8.5(b), les arêtes* 1 *et* 2 *soit* $(A,B), (C,D)$ *ont la même couleur. Les arêtes* 3 *et* 4, *soit* $(A,D), (B,C)$ *ont aussi la même couleur, différente de la couleur des arêtes* $(A,B), (C,D)$. *Ces transitions forment un cycle valide de l'automate avec longueur* 2 *qui est le nombre chromatique du graphe de la figure 8.5(b).*

[2] $\Phi_{min}(e)$ donne la couleur de l'arête e.
[3] Un point de rendez-vous équivaut à une arête.

Il faut maintenant prouver que le cycle défini dans le paragraphe précédent (d'une longueur égale au nombre chromatique) est un cycle minimal. Pour cela, supposons le contraire, c'est-à-dire qu'il existe un ordonnancement (numérotation) pour lequel il existe dans l'automate produit complet un cycle, dont la longueur ϕ est inférieure au nombre chromatique. Soit $\{e_1^1, ..e_{k_1}^1\}$, ... , $\{e_1^\phi, ..e_{k_\phi}^\phi\}$ ce cycle. Il est facile de montrer, à partir de la définition de l'automate produit complet, que chaque arête de G apparaît une et une seule fois dans ce cycle. Et cela parce que ce cycle est minimal et parce qu'il existe nécessairement un cycle $\{e_1\}, \ldots, \{e_N\}$ tel que $N = |E|$ et $\sigma(e_i) = i$. Alors, $\Phi(e_j^i) = i$ est une coloration valide des arêtes utilisant ϕ. Il y aurait donc une coloration avec un nombre chromatique plus petit que ϕ_{min} ce qui n'est pas possible et contredit l'hypothèse.
Ainsi, la longueur du cycle minimal est égale au nombre chromatique de G.
□

Cette propriété montre que si on fait une coloration des arêtes du graphe des points de rendez-vous et ensuite une numérotation selon la définition 9, alors on obtient un cycle minimal dans l'automate produit complet. Dans ce cycle minimal, dont la longueur est égale au nombre chromatique du graphe, un maximum de tâches peut avoir lieu en parallèle. Ce résultat montre donc que la numérotation obtenue avec l'algorithme de coloration peut apporter des améliorations à l'exécution de l'algorithme d'ordonnancement.

8.2 Réduction de la consommation de l'énergie

Pendant les tests, nous nous sommes aperçus, que certaines numérotations des points de rendez-vous peuvent diminuer les déplacements des nœuds. Cette diminution peut entraîner une réduction de la consommation de l'énergie. Ainsi, cette section présente une méthode pour numéroter les points de rendez-vous des nœuds.

8.2.1 Amélioration de la numérotation

Comme il est souligné dans la section 7.2 du chapitre 7, la numérotation des points de rendez-vous est complètement aléatoire. En fait, cette numérotation affecte les déplacements des nœuds, qui sont un facteur important de la consommation de l'énergie.

Considérons l'exemple du réseau avec son diagramme de Voronoï de la figure 6.2 de la section 6.2.2, représenté ici dans la figure 8.7. Une numérotation aléatoire de son graphe de points de rendez-vous peut être celle représentée dans la figure 8.8 (en gras le graphe des points de rendez-vous).

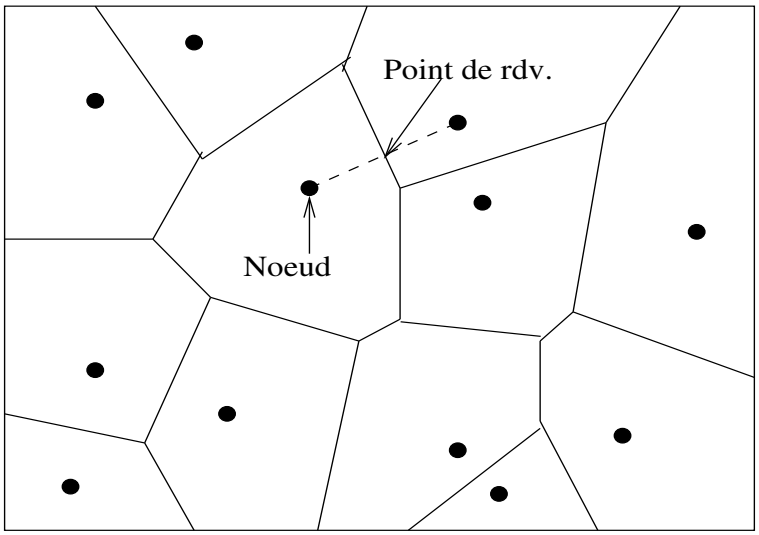


FIG. 8.7 – Exemple de diagramme de Voronoï

Si on considère le nœud n_7, alors la trajectoire du parcours de ses points de rendez-vous est représentée en flèches grises dans la figure 8.8 ou agrandi dans la figure 8.9. Ce nœud va être initialisé dans son plus petit point de rendez-vous 4, et va parcourir ses points de rendez-vous dans l'ordre, $4 \rightarrow 5 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 4$, respectant l'algorithme des points de rendez-vous.

Dans la pratique une telle trajectoire n'est pas très favorable car elle nécessite des déplacements supplémentaires. Ainsi, une trajectoire qui suit le sens des aiguilles d'une montre, ou le contraire, serait plus convenable car elle pourrait diminuer la distance de parcours. On peut donc, sans changer la nature de l'algorithme des points de rendez-vous, choisir une numérotation qui suit le sens des aiguilles d'une montre.

Pour cela, on choisit au hasard un nœud et on numérote ses points de rendez-vous dans le sens des aiguilles d'une montre. Soit n_k ce nœud et $p_1, p_2, ..., p_i$ une série croissante d'entiers positifs, la numérotation de ses points de rendez-vous. Ensuite, on continue en numérotant les points du

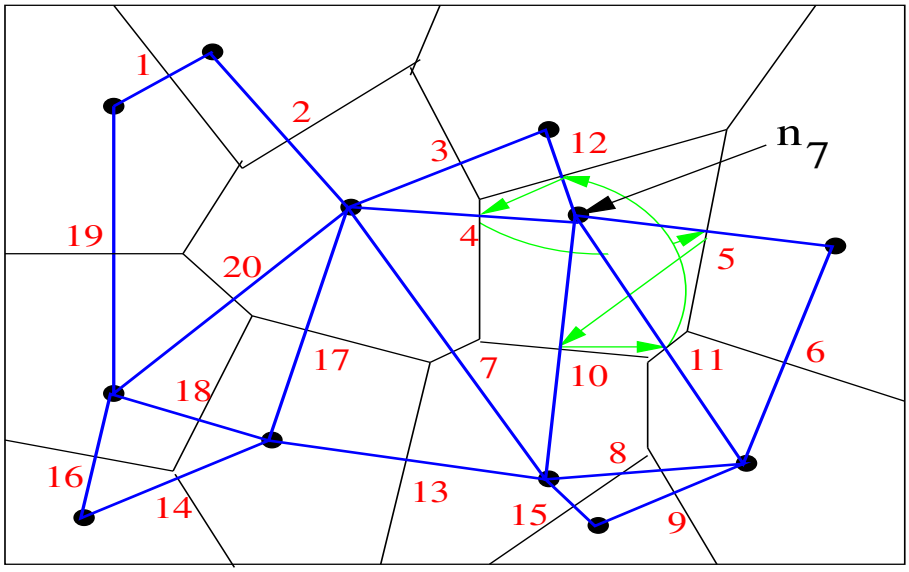


FIG. 8.8 – Exemple de diagramme de Voronoï avec le graphe des points de rendez-vous

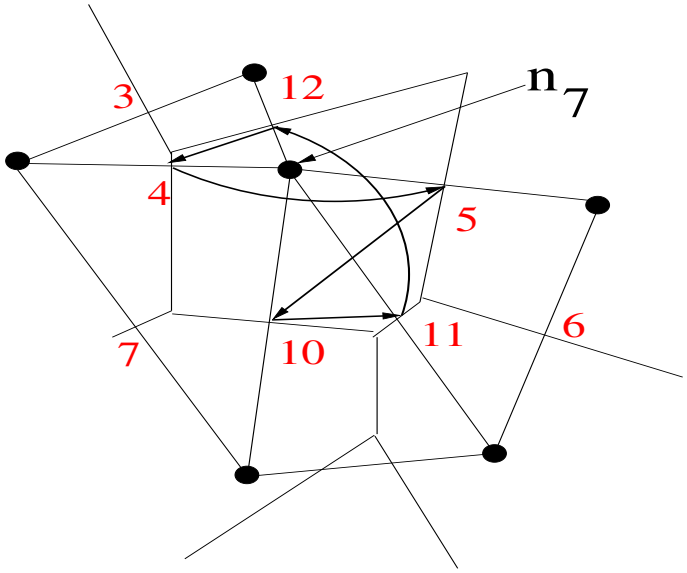


FIG. 8.9 – Une partie de l'exemple précédent agrandi

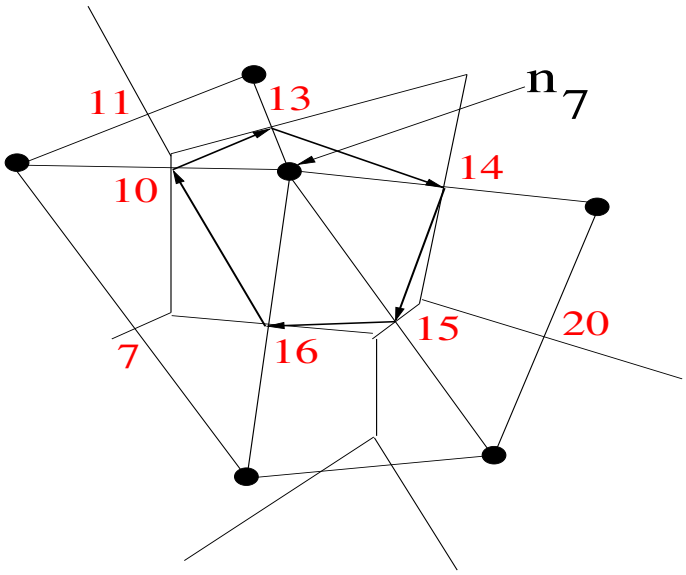


FIG. 8.10 – Une partie du diagramme de Voronoï avec le graphe des points de rendez-vous

nœud qui partage le point de rendez-vous p_i avec le nœud n_k. Soit n_l ce nœud et $p_i, p_{i+1}, ..., p_{i+j}$ la numérotation de ses points de rendez-vous. La numérotation continue en choisissant comme prochain nœud celui qui partage le point de rendez-vous p_{i+j} avec n_l dans le sens ou le sens inverse des aiguilles d'une montre (en effet le sens de la numérotation dépend du fait que d'autre points de n_l peuvent déjà être numérotés).

Si tous les points de ce nœud sont déjà numérotés, alors on choisit le nœud qui partage le point de rendez-vous p_{i+j-1} avec n_l, et ainsi de suite. À la fin de ce procédé certains nœuds auront leurs points de rendez-vous numérotés dans le sens des aiguilles d'une montre, certains dans le sens inverse et certains dans un ordre quelconque. La version du graphe de la figure 8.8, numéroté selon cette nouvelle méthode est représentée dans la figure 8.10.

Dans ce cas, le parcours des points de rendez-vous du nœud n_7 est fait de façon « circulaire » ($10 \rightarrow 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 10$) ce qui peut réduire les déplacements.

Il faut noter que cette méthode de numérotation n'est pas compatible avec l'heuristique de la coloration. Le choix de l'heuristique à utiliser peut être faite selon l'étendue de la topologie du réseau. Si la taille de la topologie

est considérable, l'heuristique de la numérotation peut être choisie pour limiter les déplacements. Cependant, si beaucoup de messages circulent dans le réseau, alors l'heuristique de la coloration peut être plus adaptée pour limiter le délai de bout en bout de la transmission des messages.

8.3 Tolérance aux défaillances

Selon le modèle des nœuds que nous avons introduit dans la section 6.2.4, un nœud mobile peut subir des défaillances. Dans cette thèse, nous considérons les défaillances par arrêt, par exemple quand la source d'énergie d'un nœud est épuisée. Cette section présente notre approche pour faire face à de tel sorte de pannes dans le réseau. Nous introduisons un nouvel algorithme tolérant aux pannes ainsi que la notion de l'automate produit complet temporisé. Cet automate, construit à partir de l'automate produit complet, est utilisé pour prouver une propriété sur l'état local (dans quel point de rendez-vous se trouve le nœud à un instant donné) de chaque nœud dans le réseau.

8.3.1 La gestion des pannes

Notre stratégie de parcours des points de rendez-vous présentée dans la figure 7.5 n'est pas tolérante aux pannes. En effet, si un nœud tombe en panne, tous les autres nœuds resteront bloqués puisqu'ils seront en attente récursivement les uns des autres. Comme tous les nœuds sont bloqués, un nœud ne peut pas déduire localement si le nœud qu'il attend est tombé en panne ou s'il attend lui aussi un autre nœud.

Par exemple, pour un réseau formé de quatre nœuds, représenté par le graphe des points de rendez-vous de la figure 8.11 si le nœud D tombe en panne pendant son parcours du point de rendez-vous 4 au point de rendez-vous 2, alors le nœud C ne pourra pas savoir si c'est D qui est tombé en panne ou si D attend le nœud A qui pourrait lui aussi être tombé en panne.

Pendant les simulations sur différents exemples de réseau ad hoc, nous avons remarqué que : si les durées des tâches *(mouvements+tâches accomplies entre deux points de rendez-vous)* sont des constantes, alors le réseau pourra atteindre un état stable. Cet état stable est défini comme suit :

Definition 12 (État stable) *Un état stable correspond à un état du système dans lequel il est possible de prévoir :*

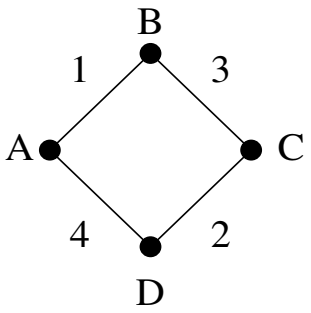


FIG. 8.11 – Le graphe des points de rendez-vous

1. *Quel nœud sera le premier à atteindre un point de rendez-vous.*
2. *Combien de temps il va attendre pour rencontrer son voisin.*

Ainsi, l'état stable correspond à un cycle dans l'automate produit complet qui a une longueur fixe. Ce cycle permet de détecter les pannes. En effet, dans l'état stable, si un nœud ne rencontre pas son voisin après son temps d'attente prédéfini, il pourra détecter localement que ce lien de communication est tombé.

Par exemple, si un nœud est toujours le premier à arriver dans un point de rendez-vous et que après un temps prédéfini il ne voit pas son voisin arriver, alors il peut déduire que ce « lien » n'existe plus. La même conclusion peut être déduite si le nœud est supposé à arriver deuxième dans le point de rendez-vous et que quand il arrive il n'y trouve pas son voisin. Ainsi, il pourra continuer à accomplir ses tâches.

Pour faire face aux dérives d'horloge, il suffit que les nœuds synchronisent leurs horloges quand ils se rencontrent dans un point de rendez-vous. Par exemple, supposons que deux nœuds n_i et n_j se rencontrent pour la première fois au point de rendez-vous p_k à l'instant t et pour la deuxième fois à l'instant $t + T$. Alors, leur troisième rencontre aura lieu à l'instant $t + 2T$ et ainsi de suite. Les nœuds n_i et n_j synchronisent leurs horloges à l'instant $t+T$, $t+2T$ et ainsi de suite. Ainsi, la dérive de l'horloge entre n_i et n_j est négligeable par rapport à T.

De plus, nous tolérons une marge d'erreur pour les durées des mouvements et empêchons les nœuds de quitter les points de rendez-vous avant les durées prédéfinies des mouvements et des temps d'attente dans ces points de rendez-vous. En d'autre termes, un nœud n_i attend jusqu'au délais maximum son

voisin n_j avant de quitter son point de rendez-vous. Mais si n_j arrive avant l'expiration de ce délais, alors ils communiquent tout de suite. Cela veut dire que la communication a lieu au plus tôt.

Ainsi, pour gérer les erreurs, nous avons fixé les durées des tâches entre le départ d'un point de rendez-vous et l'arrivée au point de rendez-vous suivant du même nœud. Cela n'est pas vraiment une nouvelle restriction puisque auparavant nous avons fixé les points de rendez-vous et leur parcours.

Nous avons ajouté ces nouveaux paramètres à notre algorithme d'ordonnancement des mouvements (figure 7.5). Le nouvel algorithme qui inclut la tolérance aux pannes est présenté dans la figure 8.12.

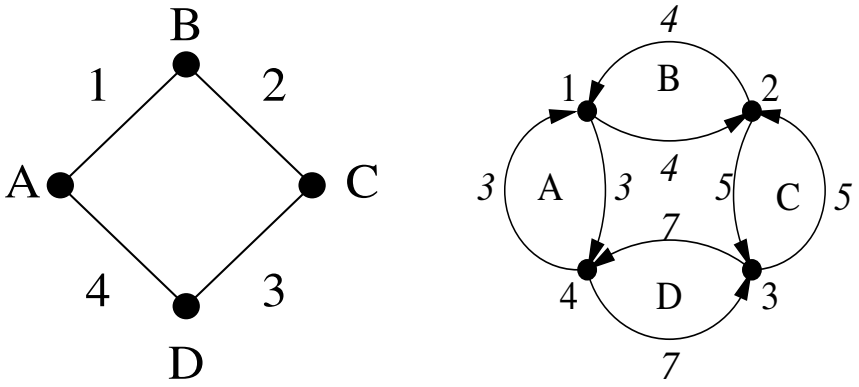
```
1: initialiser n_k dans son point de rendez-vous avec le plus petit numéro ;
2: for chaque point de rendez-vous de n_k dans l'ordre et en boucle do
3:    if un nœud est en attente dans ce point de rendez-vous then
4:       échanger des messages s'il y en a ;
5:       notify() ; //déclenche le mouvement vers le point de rdv suivant
6:       if si un message m est reçu then
7:          if la destination de m est n_k then
8:             traiter m ;
9:          else
10:            conserver m pour le routage ;
11:         end if
12:      end if
13:      ajouter le reste du temps d'attente à la tâche suivante ;
14:      continuer le mouvement vers le point de rendez-vous suivant ;
15:   else
16:      while temps d'attente de n_k pas expiré do
17:         wait() ; //reste en attente de l'autre nœud dans ce point de rdv
18:      end while
19:      ajouter le reste du temps d'attente à la tâche suivante ;
20:      continuer le mouvement vers le point de rendez-vous suivant ;
21:   end if
22: end for
```

FIG. 8.12 – Algorithme d'ordonnancement tolérant aux pannes du nœud n_k

Ce nouvel algorithme change du premier dans deux aspects :

- quand un nœud est le premier arrivé dans un point de rendez-vous, il attend son voisin comme dans le premier algorithme, mais si la durée réelle de la tâche ajoutée au temps d'attente dépasse la durée totale prédéfinie de la tâche, alors le nœud quitte le point de rendez-vous (ligne 20) et considère que son voisin est tombé en panne ;
- quand un nœud quitte le point de rendez-vous avant la durée totale prédéfinie de la tâche (lignes 13 et 19), le reste du temps de la tâche courante est ajouté à la durée prédéfinie de la tâche suivante (lignes 13 et 19). Ainsi, les temps sont non-corrélés et les nœuds opèrent de manière asynchrone.

Pour illustrer l'exécution de cet algorithme, prenons comme exemple le graphe des points de rendez-vous numéroté présenté dans la figure 8.13. Cette figure représente un graphe de points de rendez-vous numéroté (figure 8.13(a)) avec son graphe des déplacements (figure 8.13(b)).



(a) Graphe des points de rendez-vous

(b) Graphe des déplacements

FIG. 8.13 – Exemple de graphe de points de rendez-vous et son graphe des déplacements

Les arcs du graphe des déplacements sont valués pour représenter les durées des tâches entre les points de rendez-vous. Par exemple, pour le nœud D la durée des tâches entre ses points de rendez-vous est de 7 unités de temps. Fixons une marge d'erreur de 2 unités de temps.

Alors, selon l'algorithme, les nœuds A et B sont initialisés dans le point

de rendez-vous 1, C dans le point 2 et D dans 3. Après la rencontre dans le point 1 avec B, A se déplace vers le point de rendez-vous 4 où il est le premier à arriver et attend le nœud D. Supposons que son temps d'attente dans 4 soit 13 unités de temps. Ainsi, pour le nœud A la tâche « mouvement vers le point de rendez-vous 4 + temps d'attente dans le point de rendez-vous 4 + marge d'erreur » a une durée de $3+13+2=18$ unités de temps. Alors, si ce temps est dépassé et A n'a pas vu D arriver, alors il considère que D est tombé en panne. Sinon, si le nœud D arrive dans le point de rendez-vous 4 après, par exemple, 17 unités de temps, alors le reste du temps de la tâche, c'est-à-dire $18-17=1$ unité de temps, est ajouté à la tâche suivante c'est-à-dire à la tâche « mouvement vers le point de rendez-vous 1 + temps d'attente dans le point de rendez-vous 1 + marge d'erreur ».

Maintenant il faut démontrer que l'état stable du réseau, observé dans les exemples, pourra être généralisé à n'importe quel réseau et qu'il est possible de démarrer les nœuds dans cet état.

L'état stable du système est très important, car dans cet état l'ordre d'arrivée aux points de rendez-vous dans l'automate produit reste le même. Plus précisément, si tous les nœuds respectent l'algorithme des points de rendez-vous de la figure 8.12, alors dans un point de rendez-vous donné, il y aura toujours le même nœud qui arrivera le premier (ou deuxième) et toujours son voisin qui arrivera le deuxième (ou premier) au bout d'un temps constant. Dans ce cas, connaissant l'ordre de « visite » des points de rendez-vous, on peut détecter les pannes dans le système. Pour étudier l'état stable du système, on introduit d'abord la notion de l'automate produit complet temporisé. Dans cet automate, on ajoute une variable de temps aux états locaux qui composent les états de l'automate produit. Ensuite on discrétise le temps représenté par cette variable. Cela aura pour effet l'augmentation du nombre d'états de ce nouvel automate par rapport à l'automate produit. La définition de l'automate produit complet temporisé est la suivante :

Definition 13 (Automate produit complet temporisé) *On appelle automate produit complet temporisé, un automate produit complet dans lequel on ajoute une variable de temps à chaque état local. Dans cet automate, il existe une transition de l'état* $([q_1,t_1],\ldots,[q_n,t_n])$ *à l'état* $([q'_1,t'_1],\ldots,[q'_n,t'_n])$ *si pour chaque* $i \in [1..n]$ *:*

1. *soit* $q'_i = q_i$ *et* $t'_i = t_i$ *avec* t_i *égal à la durée de la tâche dont le point de rendez-vous destination est numéroté* $label(q_i)$*, et tel qu'il n'existe pas*

un j tel que, $label(q_i) = label(q_j)$ et t_j = durée de la tâche dont le point de rendez-vous destination est numéroté $label(q_j)$;

2. *soit $q'_i = q_i$ et $t'_i = t_i + \delta$ où δ est le « pas de discrétisation » et t_i est inférieur à la durée de la tâche dont le point de rendez-vous destination est numéroté $label(q_i)$;*
3. *soit $q'_i \neq q_i$ et $t'_i = \delta$ avec t_i égal à la durée de la tâche dont le point de rendez-vous destination est numéroté $label(q_i)$ et il existe j tel que, $label(q_i) = label(q_j)$ et t_j = durée de la tâche dont le point de rendez-vous destination est numéroté $label(q_j)$.*

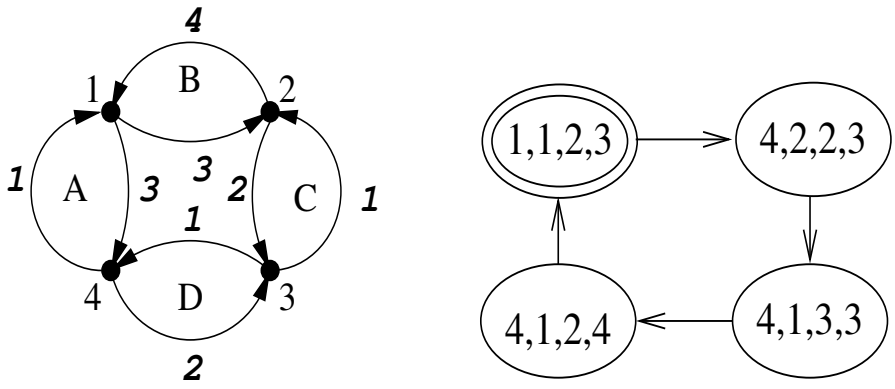
L'état initial de l'automate produit complet temporisé est l'état composé de tous les états initiaux des automates locaux avec la variable temps est égal à δ, $([i_1, \delta], ..., [i_n, \delta])$.

Pour illustrer cette définition, reprenons l'exemple du graphe des points de rendez-vous de la figure 8.13(a). Son graphe des déplacements est représenté dans la figure 8.14(a). Dans ce graphe, les nombres en italique sur les arcs représentent les durées des tâches entre les points de rendez-vous pour chaque nœud [4]. Son automate produit complet est représenté dans la figure 8.14(b), tandis que la figure 8.15 représente l'automate produit complet temporisé avec $\delta = 1$ et état initial $([1,1],[1,1],[2,1],[3,1])$.

Illustrons maintenant les 3 cas où il y a une transition dans l'automate produit complet temporisé comme ils sont définis par la définition 13. Pour ces illustrations nous ne considérons que l'automate local du nœud A, donc $i = 1$. Alors, la transition marquée $item1$ dans la figure 8.15 représente le cas 1 de la définition. Elle représente une transition entre les états $([4,3],[2,3],[2,1],[3,2])$ et $([4,3],[1,1],[3,1],[3,2])$ qui correspond à une attente dans le point de rendez-vous 4. Dans ce cas, $q'_1 = q_1 = 4$ et $t'_1 = t_1 = 3$ et t_1 est égal à la durée de la tâche du nœud A vers le point de rendez-vous destination $label(q_1) = 4$ et il n'existe pas un j tel que $label(q_1) = label(q_j)$ et t_j = durée de la tâche dont le point de rendez-vous destination est numéroté $label(q_j)$. En effet, on peut voir dans l'état $([4,3],[2,3],[2,1],[3,2])$ que les états locaux des autres automates sont tous différents de $label(q_1) = 4$.
Le cas 2 est représenté par la transition marquée $item2$ dans la figure 8.15. C'est une transition entre les états $([4,2],[2,2],[2,1],[3,2])$ et $([4,3],[2,3],[2,1],[3,2])$

[4] Par rapport à la figure 8.13(b), les poids sur les arcs ont changé afin de mieux illustrer la définition.

de l'automate qui correspond à l'arrivée du nœud A dans le point de rendez-vous 4. Dans ce cas, $q'_1 = q_1 = 4$ et $t'_1 = t_1 + \delta$ où $t'_1 = 3$, $t_1 = 2$ et $\delta = 1$.
Le cas 3 est représenté par la transition marquée $item3$ dans la figure 8.15. C'est une transition entre les états $([4,3],[1,3],[2,1],[4,1])$ et $([1,1],[1,4],[2,1],[3,1])$ qui correspond à la fin de la tâche des nœuds A et D dans le point de rendez-vous 4 . Dans ce cas, $q'_1 = 1$, $q_1 = 4$, $(q'_1 \neq q_1)$, $t'_1 = \delta = 1$ et $t_1 = 3$ est égal à la durée de la tâche du nœud A vers le point de rendez-vous 4 et il existe $j = 4$ (dans l'automate local du nœud D) tel que $label(q_1) = label(q_4) = 4$ et $t_4 = 1$ est égal à la durée de la tâche dont le point de rendez-vous destination est numéroté 4.



(a) Graphe des déplacements

(b) L'automate produit correspondant au graphe de la figure 8.1(b)

FIG. 8.14 – Exemple pour illustrer la définition 13

Nous avons utilisé ce type d'automate pour prouver la proposition suivante :

Proposition 5 *Étant donné un ordonnancement et un ensemble de durées pour chaque tâche entre deux points de rendez-vous, il existe un état stable où tous les nœuds sont périodiquement dans le même état.*

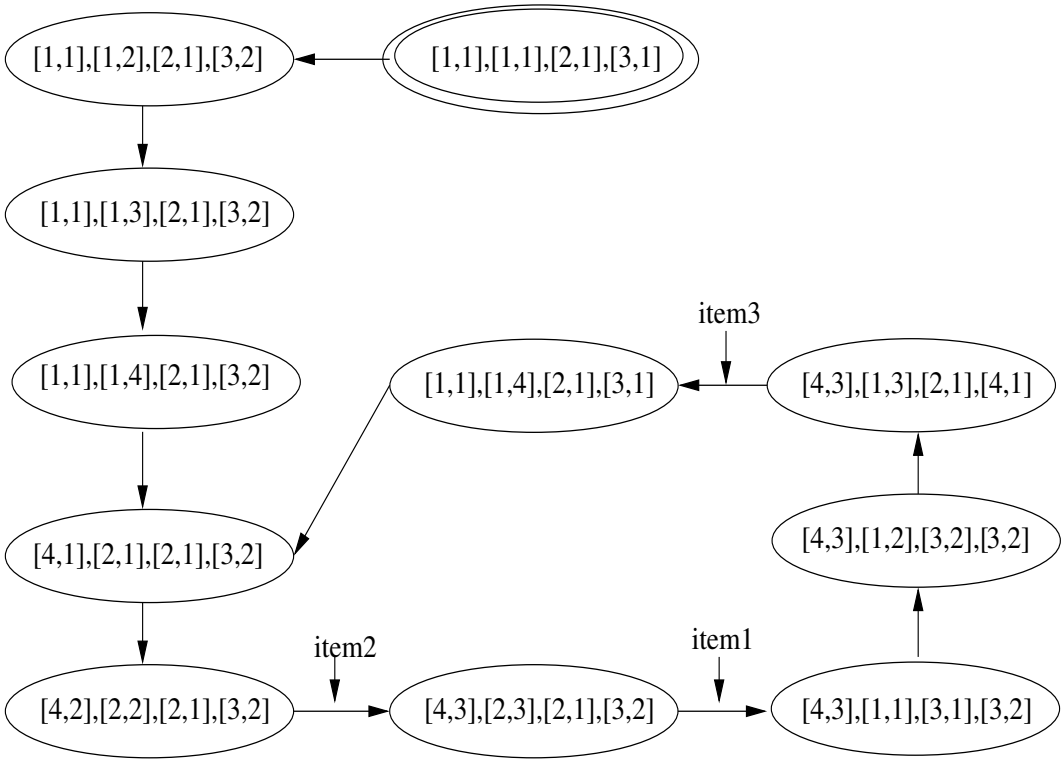


FIG. 8.15 – L'automate produit temporisé correspondant à l'automate de la figure 8.14(b)

Preuve Pour prouver cette propriété, on discrétise les temps[5] et on utilise l'automate produit complet temporisé.

Puisque, par définition, le nombre d'états dans cet automate produit complet temporisé est fini et que chaque état a seulement un successeur, chaque suite de transitions dans l'automate mène à un cycle. Ce cycle a, par construction, une durée fixe, ce qui prouve la proposition. □

Pour calculer les temps d'attente des nœuds dans un point de rendez-vous, on construit explicitement l'automate produit temporisé à partir de l'état initial. Afin de minimiser la taille de l'automate, on peut parfois construire

[5]Étant donnée une marge d'erreur sur la durée il est toujours possible de choisir des valeurs rationnelles pour le temps.

seulement les états où il y a des changements des états locaux des nœuds. Quand le cycle dans l'automate produit temporisé est trouvé, on déduit les temps d'attente pour chaque nœud dans ses points de rendez-vous à partir de l'état initial. Par exemple, dans la figure 8.15, on peut déduire que le nœud A attend 3 unités de temps dans le point de rendez-vous 4. Plus précisément, l'arrivée du nœud A dans le point de rendez-vous 4 est représentée par l'état $([4,3],[2,3],[2,1],[3,2])$. Son attente dans ce point de rendez-vous est alors représenté dans le cycle par les états : $([4,3],[1,1],[3,1],[3,2])$, $([4,3],[1,2],[3,2],[3,2])$, $([4,3],[1,3],[2,1],[4,1])$ qui restent tous sur $[4,3]$.

8.3.2 Routage dynamique

Quand un nœud tombe en panne, il ne peut plus parcourir ses points de rendez-vous. Si p_i est le nombre de ses points, alors p_i liens de communication ne sont plus valides. Dans ce cas, nous proposons d'utiliser un protocole à état de liens de type OSPF pour que tous les nœuds connaissent leurs ruptures de liens et recalculent les chemins les plus courts entre les nœuds (en fait le chemin le plus court vers un point de rendez-vous).

Contrairement aux protocoles à vecteur de distances comme RIP (*Routing Information Protocol*), les protocoles à état de liens n'envoient pas aux routeurs adjacents le nombre de sauts qui les sépare d'une destination, mais l'état de leurs liaisons. De cette façon, chaque routeur est capable de dresser une carte de l'état du réseau et peut par conséquent calculer à tout moment la route la plus appropriée pour un message donné.

Un protocole à état de liens prend en compte une rupture de lien dans le réseau, par exemple un routeur qui tombe en panne, et recalcule localement les plus courts chemins vers les autres routeurs. Puisque l'on opère dans un environnement ad hoc, une autre solution pourrait consister à utiliser une version modifiée des protocoles à état de lien comme OLSR ou TBRPF.

À la différence de ces protocoles, dans notre cas, on peut éviter l'envoi de certains messages (HELLO) sur l'état des liens, car notre topologie est figée et connue à l'avance. De plus, nous considérons des réseaux où les nœuds sont la plupart du temps déconnectés et où les conditions de rupture de liens sont différentes de celles de ces protocoles.

Dans notre cas, si un nœud tombe en panne, son voisin peut le détecter comme cela est décrit dans la section 8.3.1. Ce voisin diffuse alors un message dans le réseau pour avertir tous les autres nœuds. Ce message peut contenir un numéro de séquence pour éviter les boucles. Il faut noter ici que nous

considérons que le nœud tombe en panne et non la rupture d'un seul de ses liens. Ainsi, on ne diffuse qu'un seul message dans le réseau pour annoncer que le nœud est tombé en panne plutôt que p_i messages pour annoncer les ruptures des p_i liens.

Si le voisin d'un nœud n_i tombe en panne, alors n_i ne recalcule pas les temps d'attente dans ses points de rendez-vous.[6] Il continue à les parcourir comme avant. Il continue donc à détecter que le nœud est en panne, mais il ne diffuse pas systématiquement cette information. Toutefois, pour s'assurer que l'information n'est pas perdue, il diffuse périodiquement un message pour annoncer la panne toutes les kT unités de temps, où T est la période de parcours de ces points de rendez-vous. Ainsi, chaque nœud doit recevoir environ[7] toutes les kT unités de temps les messages qui annoncent un même nœud en panne. Si un nœud ne reçoit plus un tel message, cela veut dire que soit le nœud en panne est rétabli, soit que d'autres nœuds intermédiaires sont tombés en panne. Dans tous les cas, si un nœud ne reçoit pas de message indiquant la panne d'un autre nœud, il considère au bout d'un certain nombre de périodes (par exemple 3) que le nœud est rétabli.

Au rétablissement d'un nœud après une panne, il s'annonce par un message diffusé avec un numéro de séquence (comme lors de la détection d'une panne) pour assurer une réutilisation rapide de ce lien. Pour cela, il se dirige vers son point de rendez-vous le plus proche et attend son homologue. Si au bout de T unités de temps, il ne le rencontre pas, alors il peut considérer que son homologue est en panne. Il va alors dans le point de rendez-vous suivant et il recommence. Dès qu'il rencontre un homologue, il lui transmet l'information qu'il est rétabli pour qu'il la diffuse. De son coté, l'homologue lui transmet l'état des liens rompus du réseau. De plus, il « négocie » avec lui pour déterminer dans quel état de l'automate produit complet temporisé il se trouve. Cela lui permet de déduire combien de temps il doit attendre avant de commencer l'exécution de l'algorithme d'ordonnancement.

À ce stade, nous n'avons pas encore implémenté ce protocole dans nos simulations. Il reste à valider empiriquement que les constantes choisies (e.g. k) sont valables.

[6] Il faut noter ici que, nous considérons que les données des nœuds sont stockées dans une mémoire non-volatile. Ainsi, elles ne sont pas perdues après une panne d'énergie, par exemple.

[7] Cette durée peut varier à cause des changements de routes.

8.4 Évaluation comparative avec un algorithme d'auto-stabilisation

El Haddad et Haddad [36] ont introduit un schéma auto-stabilisant qui utilise l'ordonnancement des visites des points de rendez-vous selon notre premier algorithme présenté dans la figure 7.5. L'auto-stabilisation est la propriété d'un algorithme réparti capable de retrouver seul un fonctionnement correct après apparition d'une panne transitoire. Une panne transitoire affecte toute donnée volatile, qu'elle soit en cours d'acheminement (message) ou dans une mémoire.

Si la spécification de l'algorithme réparti concerne un comportement particulier (par ex. circulation sans fin d'un jeton sur chaque site), alors l'algorithme retrouvera en temps fini une séquence de configurations dans laquelle le comportement est réalisé (e.g. le jeton visite effectivement chaque site). Pour ce type d'algorithme, la spécification porte sur les exécutions (enchaînement de configurations). L'algorithme autostabilisant converge en temps fini vers une exécution correcte légitime.

Dans leur travail, El Haddad et Haddad s'intéressent au problème de coopération de robots mobiles et ils modélisent le système par un réseau de Petri. Pour cela, ils supposent une topologie telle que celle que nous avons décrit dans la section 6.1.3, mais y ajoutent en plus des hypothèses.

À chaque robot est associé une variable *timeout* prenant une valeur maximale égale au nombre de points de rendez-vous dans le système, N. De plus, une variable *position* représente la position courante du robot, c'est-à-dire s'il se trouve dans un point de rendez-vous ou en déplacement. Une autre variable *statut* représente l'état du robot. Elle prend la valeur *moving* si le robot est en déplacement ou la valeur *waiting* s'il est en attente dans un point de rendez-vous. L'algorithme applique la même stratégie que la notre, c'est-à-dire le parcours des points de rendez-vous dans l'ordre croissant. Le comportement d'un robot est événementiel : l'occurrence d'un événement déclenche l'exécution d'un code dépendant de l'état courant du robot. Le point limitant de leur algorithme est que le temps de parcours entre deux points de rendez-vous doit impérativement être de 1 unité de temps.

En plus de notre premier algorithme d'ordonnancement des déplacements, les auteurs proposent un algorithme auto-stabilisant qui se décompose en quatre parties.

– Si le robot est le premier à arriver dans le point de rendez-vous, alors

il attend que son *timeout* expire, il réarme son *timeout* à N unités de temps et se met en attente.
- Si un robot arrive dans un point de rendez-vous où son homologue l'attend, alors il doit attendre que son *timeout* expire avant de pouvoir communiquer. Donc la communication commence au plus tard et pas au plus tôt comme c'est le cas de notre algorithme.
- Si un robot détecte une panne (expiration du *timeout*) lors de l'attente dans un point de rendez-vous, alors il réarme son *timeout* à 1 unité de temps et soit reste sur place, soit se dirige vers son premier point de rendez-vous. Ce choix est effectué aléatoirement selon une distribution de probabilité uniforme.
- Sinon, si le robot tombe en panne (expiration du *timeout*) lors de son déplacement entre deux points de rendez-vous, alors il arme son *timeout* à 1 unité de temps et se dirige vers son point de rendez-vous ayant le plus petit numéro.

Dans les deux premiers cas, après une éventuelle communication, le robot réarme son *timeout* à 1 unité de temps et débute son déplacement vers le prochain point de rendez-vous.

Toutes ces contraintes sont fortes car tous les temps de déplacement doivent être accordés sur le temps maximum de parcours entre deux points de rendez-vous. Ce schéma est difficilement utilisable dans un réseau, car le délai de bout en bout d'une transmission de message pourra être très grand. De plus, les auteurs ne définissent pas précisément quelle sorte de panne est prise en compte par leur algorithme.

El Haddad et Haddad se sont uniquement intéressés au problème d'auto-stabilisation de l'ordonnancement des robots mobiles. Ils ne se sont pas intéressés aux fortes implications que peut avoir leur modèle d'auto-stabilisation sur le routage de messages dans le réseau. L'avantage de leur solution est qu'elle ne nécessite pas de construction préalable (construction de l'automate produit complet temporisé). Toutefois, nous ne sommes pas sûrs qu'un tel calcul ne serait pas nécessaire si les temps de parcours n'étaient pas fixés à 1 unité de temps.

8.5 Récapitulatif des propriétés de l'algorithme

Dans le chapitre 7, nous avons présenté un algorithme d'ordonnancement de mouvements des nœuds mobiles d'un réseau ad hoc, tandis que dans ce

chapitre nous avons ajouté à cet algorithme la tolérance aux pannes. Le but de cet algorithme est d'introduire un contrôle de la mobilité des nœuds, afin d'assurer une communication dans un temps borné, réduire la longueur des routes, la consommation de l'énergie et faire face aux défaillances.

Par rapport aux mouvements des nœuds, cet algorithme est totalement asynchrone, car plusieurs nœuds peuvent se déplacer en parallèle. Il l'est également par rapport aux communications car plusieurs messages peuvent être acheminer en même temps dans le réseau. Cet algorithme respecte les critères définit dans la section 5.1.

- le schéma des mouvements des nœuds sera le même pour différents modèles de trafic ;
- ce schéma est distribué. Ainsi, une fois les points de rendez-vous définit, il n'y a plus d'entité centrale qui calcule les mouvements des nœuds. Ils parcourent leurs points de rendez-vous, comme il est défini dans l'algorithme, pour lesquels il suffit qu'ils connaissent les positions des points de rendez-vous. Ainsi, il n'est pas nécessaire que les nœuds connaissent leurs positions exactes comme le font certains solutions décrites dans le chapitre 5. Ce schéma assure une passage à l'échelle efficace. L'utilisation des points de rendez-vous peut être très convenable dans les cas où il existe dans le réseau un trafic qui n'est pas uniformément reparti et qu'il faut plutôt utiliser certains points que d'autres pour le routage des messages ;
- ce schéma n'est pas seulement capable d'assurer une communication dans un temps borné, mais aussi de satisfaire d'autres contraintes. Ainsi, il réduit la consommation de l'énergie et de la bande passante, car grâce au mécanisme des points de rendez-vous les nœuds peuvent utiliser une petite zone de transmission.

L'algorithme supporte les parallélisme dans le réseau, c'est-à-dire que plusieurs tâches peuvent avoir lieu en parallèle. Il peut aussi faire face aux défaillances qui peuvent avoir lieu dans le réseau.

8.6 Résumé

Ce chapitre présente les améliorations que nous avons ajoutées à notre algorithme de base. Elles concernent d'abord l'heuristique de la coloration des arêtes du *graphe de points de rendez-vous*. À travers cette coloration

des arêtes on peut augmenter le degré de parallélisme dans le système. Ce chapitre présente aussi une heuristique sur la numérotation du graphe des points de rendez-vous. L'algorithme de numérotation présenté a pour but l'optimisation des déplacements qui sont gourmands en énergie. Nous avons aussi ajouté la tolérance aux pannes dans l'algorithme initial de la figure 7.5 et introduit un nouvel algorithme, celui présenté dans la figure 8.12. Enfin, notre travail a été l'objet d'une étude de El Haddad et Haddad [36] qui introduit un algorithme auto-stabilisant.

Chapitre 9

Simulations et implémentation

Introduction

Une des méthodes les plus importantes pour évaluer les caractéristiques des protocoles des réseaux ad hoc est la simulation. Elle fournit un certain nombre d'avantages significatifs, et en particulier, répéter des scénarios, isoler des paramètres et évaluer différentes métriques. La topologie du réseau et le mouvement des nœuds pendant la simulation sont des facteurs principaux de l'évaluation des protocoles. Une fois que les nœuds sont distribués, le modèle de mobilité définit leurs mouvements dans le réseau.

Dans le contexte des réseaux ad hoc, nous avons proposé une architecture de contrôle de la mobilité, qui a pour but l'amélioration des performances du réseau. Pour valider et prouver l'efficacité de notre solution nous avons procédé en deux étapes : d'abord par la simulation qui a permis de faire des tests et des mesures de performances, et ensuite par implantation concrète en utilisant des robots. Ces tests nous ont permis, en particulier, de vérifier l'exactitude des propriétés (e.g. absence des blocages) avant que nous ne les ayons démontrées et de prouver la faisabilité de notre solution. Nous avons conçu une plate-forme de tests composée d'une série de programmes en Java qui simulent un réseau ad hoc. Elle nous a également servi pour tester différentes métriques utilisées pour l'évaluation des performances du routage défini par notre algorithme.

Ce chapitre est composé de deux parties. Il commence par présenter les évaluations des performances par simulation (section 9.1). La deuxième partie (section 9.2) présente l'évaluation par implantation en démontrant ainsi la

faisabilité de notre solution.

9.1 Évaluation des performances par simulation

Pour tester les algorithmes et les protocoles de routage, il existe des plate-formes de tests comme par exemple, ns-2 [4], GloMoSim [3], ou JiST [66]. Dans ces plates-formes, en particulier, il n'existe pas de modules adaptés à notre architecture. Le routage et la mobilité sont, en général, traités séparément[1]. Ainsi, nous avons choisi de concevoir notre propre plate-forme adaptée à notre architecture.

9.1.1 Description de la plate-forme de simulation

Cette plate-forme est composée de deux *package* Java, un qui est utilisé pour valider par simulation les propriétés du système et un autre qui est utilisé pour simuler totalement un réseau ad hoc et faire des mesures de performances.

Le *package adhoc* simule totalement un réseau ad hoc en utilisant deux graphes :

- un graphe non-orienté. C'est le graphe des points de rendez-vous (voir section 7.1) dont les sommets représentent les nœuds du réseau et les arêtes les points de rendez-vous ;
- un graphe orienté. C'est le graphe des déplacements (voir section 7.6) dont les sommets représentent les points de rendez-vous et les arêtes les déplacements permettant de les parcourir.

Les nœuds sont aussi représentés par des *threads* et la communication entre eux par les synchronisations des *threads*. Cette synchronisation se fait avec l'aide d'un moniteur sur les arêtes du graphe des points de rendez-vous. En fait, une telle arête représente un point de rendez-vous dans lequel deux nœuds se rencontrent et peuvent échanger des messages. Une attente d'un nœud dans un point de rendez-vous est simulée avec la méthode *wait()* sur

[1] Par defaut, le modèle de mobilité utilisé est le modèle de promenade aléatoire avec temps de pause (RWpMM).

le moniteur de ce dernier, ce qui endort la *thread* représentant ce nœud. L'arrivée d'un nœud dans un point de rendez-vous est simulé par un *notify()* sur le moniteur ce qui réveille la *thread* endormie. Le déplacement entre deux points de rendez-vous est aussi simulé avec la méthode *wait()*. La durée du déplacement dépend de la vitesse des nœuds. Nous avons supposé une vitesse constante sauf pour les mesures du délais de transmission par rapport à la vitesse des nœuds. Le pseudo-code de la synchronisation est présenté ci-dessous :

```
for (;;) {
  ...
  synchronized ((edge)) {
    if (edge.onWait()) { //si un noeud est en train d'attendre
      ...
      edge.notify();     //réveiller la thread
      edge.onWait(false);
      ...
    } else {
      edge.onWait(true); //commencer à attendre le voisin
      ...
      edge.wait();       //endormir la thread
      ...
    }
  ...
}
```

Pour lancer un test, il suffit de définir le nombre de nœuds ainsi que leur vitesse. Le *package adhoc* génère le graphe des points de rendez-vous et sa numérotation de manière aléatoire. Pour cela, on utilise une instance de la classe *java.util.Random* initialisée par son constructeur *public Random(long seed)*. Le choix du constructeur avec argument est fait pour avoir le même générateur de nombres chaque fois qu'on exécute le programme. Autrement dit, pour chaque relance de test, le graphe sera le même.

Un paramètre définit le nombre minimum d'arêtes pour chaque sommet.[2] Ensuite, pour chaque sommet du graphe, on choisit aléatoirement, grâce à

[2]Comme dans le graphe des points de rendez-vous, les sommets représentent les nœuds et les arêtes les points de rendez-vous, alors ce paramètre représente le nombre minimum de points de rendez-vous pour chaque nœud.

l'objet *random*, d'autres sommets et on crée des arêtes entre eux. La méthode de la création du graphe est présentée ci-dessous :

```
/**
 * Création du graphe des points de rendez-vous.
 * @param int : paramètre qui définit le nombre minimum de
 * points de rendez-vous par noeud
 * @return la matrice des distances du graphe
 */
public int[][]  createEdgesAndMatrix(int minEdgeByState) {
   ...
   // Pour chaque sommet on crée les arêtes
   for (int i = states.length; --i >= 0;) {
      State state = states[i];
      for (int j = edgesCount; j < minEdgeByState; j++) {
        ...
        //choix aléatoire d'un sommet et création d'une arête
        do {
          aState = randomState(state);
          edge = new Edge(state, aState, edgeNumber);
        }
        while ((state.contains(edge)));
        ...
        edgeNumber++;
        state.addEdge(edge);
        aState.addEdge(edge);     //le graphe est non-orienté
        edges.put(edge,edge);
      }
    }
    ...
  }
```

À partir du graphe des points de rendez-vous, on génère ensuite le graphe des déplacements. Dans ce graphe, les sommets représentent les points de rendez-vous et les arêtes un parcours entre deux points.

De l'autre coté, le *package spector* est utilisé pour tester des propriétés du système. Il contient un programme pour la génération de l'automate produit (voir dans la section 7.4 la figure 7.7) et utilise une librairie pour l'interface graphique.

Nous avons dû développer cette plate-forme au fur et à mesure et parallèlement avec le travail de recherche de thèse. Elle est composée d'une trentaine de programmes qui simulent tout le comportement du réseau que l'on considère dans cette thèse. Cette plate-forme offre des capacités d'adaptation à d'autres algorithmes qui nécessitent une évaluation.

9.1.2 Tests de performances

Afin d'évaluer les performances de l'architecture, nous avons testé d'abord les propriétés de notre architecture. Ensuite, les tests effectués concernent l'évaluation de différentes métriques, comme le nombre de sauts qu'un message fait pour atteindre sa destination ou le délai de bout en bout de la transmission des messages. Tous les tests ont été effectués sur un PC avec 512 Mo de RAM et un processeur P4 2.8 GHz.

La taille de l'environnement où les nœuds sont déployés aléatoirement mesure entre 2500 sur $2500m$. Pour les tests, la vitesse des nœuds est supposée constante. Nous avons considéré que les nœuds se déplacent à une vitesse égale à $5m/s$.

Nous avons effectué des tests pour comparer notre algorithme avec une diffusion appliquée dans le type des réseaux ad hoc que nous considérons (voir section 5.2). Selon notre algorithme, les communications ont lieu dans un des points de rendez-vous d'un nœud tandis que pour la diffusion chaque nœud transmet le message dans tous ses points de rendez-vous. Pour ces tests, tous les nœuds du réseau envoient des messages *unicast* vers tous les autres nœuds. Ainsi, pour tester notre algorithme, si le réseau est formé de n nœuds, alors il y aura $n \times (n-1)$ transmissions de messages.

Pour le cas de la diffusion, chaque nœud transmet dans chacun de ses points de rendez-vous le message à ses voisins. Si un nœud reçoit un message qui ne lui est pas destiné, il le transmet à tous ses voisins dans chacun de ses points de rendez-vous et le rejette en gardant sa « trace » pour ne pas avoir de boucle dans le réseau. Dans ce cas, il y aura n communications dans le réseau. Ces messages sont transmis les uns après les autres. Pour chaque transmission de message, nous faisons différentes mesures de métriques et à la fin une moyenne est calculée pour produire les graphiques.

9.1.2.1 Tests contre le blocage

Nous avons voulu valider par simulation, avant les preuves théoriques, la propriété 1 du Non-blocage global de la section 7.4. Ainsi nous avons effectué des tests en simulant des réseaux ad hoc. D'abord nous avons fixé le nombre de points de rendez-vous d'un nœud au minimum à 2. Le premier test concernait un réseau formé par 10 nœuds où chaque nœud envoyait un message à tous les autres nœuds. Ensuite, nous avons augmenté progressivement le nombre de nœuds du réseau de 10. À partir du réseau avec 100 nœuds le nombre de nœuds a été augmenté de 100 en arrivant jusqu'à un réseau avec 1000 nœuds. Dans tous ces tests, la transmission des messages a été réussie sans aucun blocage (*deadlock*). Cela nous a permis de vérifier expérimentalement nos propriétés et, inversement, de s'assurer que notre plate-forme de tests avait bien été programmée.

9.1.2.2 Tests sur la longueur des chemins

Un facteur important dans la mesure des performances d'un réseau ad hoc est le nombre de sauts[3] qu'un message fait pour atteindre sa destination. En effet, puisque les liens sans fil sont vulnérables, des ruptures de liens peuvent se produire le long de ces routes longues (en nombre de sauts) rendant la communication impossible. Et puisque nous considérons des réseaux de grande taille, alors la longueur des routes devient un facteur très important tant pour le passage à l'échelle que pour la persistance des routes.

Les courbes produites représentent le nombre de sauts par rapport au nombre de nœuds du réseau. Par exemple, pour un réseau avec 50 nœuds, chaque nœud envoie un message vers tous les autres nœuds. Ainsi il y aura $50 \times 49 = 2450$ transmissions de messages. Pendant chaque transmission, on calcule le nombre de sauts que le message fait pour atteindre sa destination. Enfin, la moyenne des 2450 transmissions est calculée pour la présenter dans le graphique.

Dans tous ces tests la vitesse des nœuds était constante. La longueur des chemins est alors calculée en lançant plusieurs messages dans le réseau. La figure 9.1 représente la moyenne de la longueur des chemins (en nombre de sauts) en fonction du nombre de nœuds du réseau. La figure montre que les routes obtenues par notre algorithme sont meilleures que celles obtenues dans le cas d'une diffusion. Ceci est un résultat important car, en général,

[3] Un saut est une communication de message entre deux nœuds.

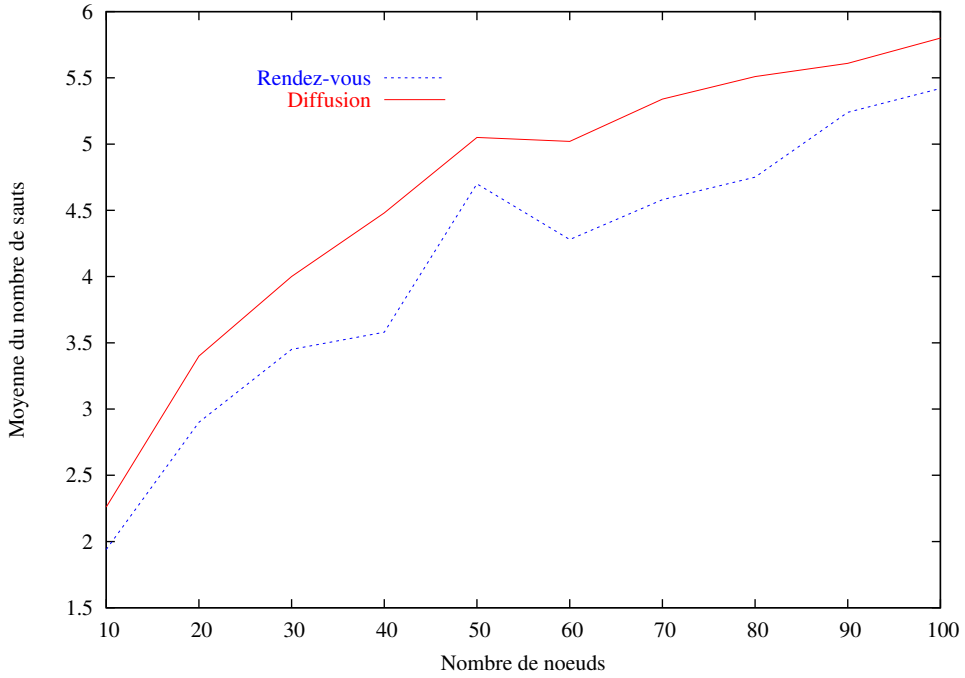


FIG. 9.1 – Le nombre de sauts par rapport au nombre de nœuds

les routes avec un petit nombre de sauts sont préférables. Ainsi, ces routes peuvent assurer un bon passage à l'échelle et rendre le mécanisme du routage plus robuste, car les ruptures de liens de communication dans ces routes sont moins probables grâce à leur petite longueur en nombre de sauts.

Les valeurs obtenues grâce à l'algorithme des points de rendez-vous sont meilleures que celles obtenues pour le cas de la diffusion. Cela est dû à notre calcul des plus courts chemins qui ne prend pas en compte les attentes dans les points de rendez-vous. De plus, on peut observer une petite anomalie dans les deux courbes pour les valeurs de la moyenne pour des réseaux formes entre 40 et 60 nœuds.

Pour expliquer cette anomalie, nous avons calculé le nombre moyen de points de rendez-vous par nœud une fois que le graphe des points de rendez-vous est généré. Ce graphique est présenté dans la figure 9.2. En fait, le graphique présente le nombre moyen d'arêtes, du graphe des points de rendez-vous, pour chaque sommet. Comme un point de rendez-vous appartient à deux nœuds, les arêtes ne sont comptés qu'une fois.

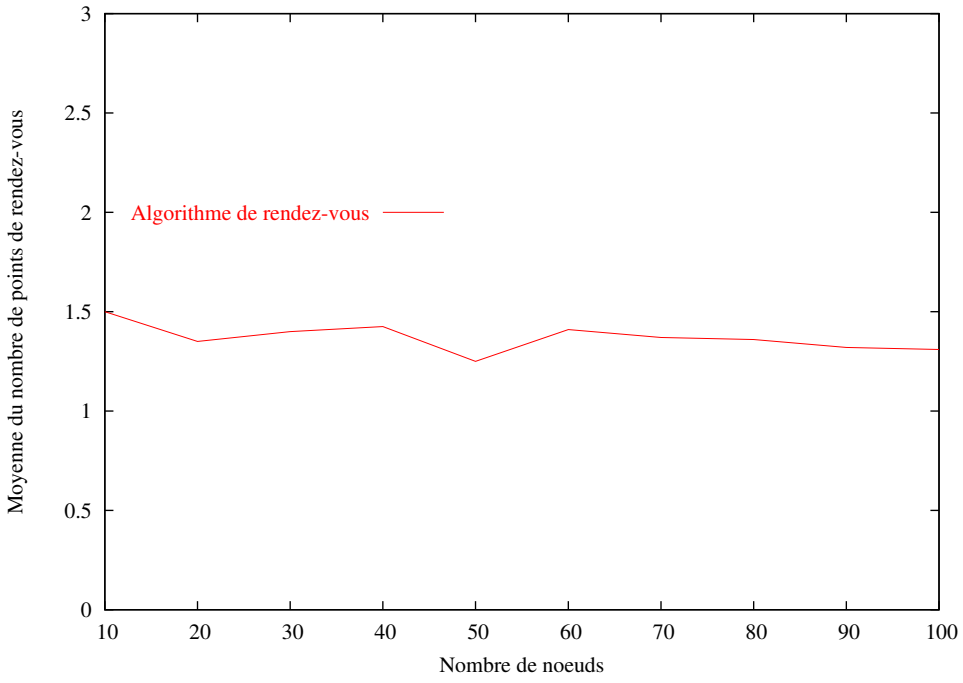


FIG. 9.2 – Le nombre moyen de points de rendez-vous par nœud

Dans cette figure on peut voir que pour les réseaux formés de 40 à 60 nœuds, le nombre moyen de points de rendez-vous par nœud diminue. Cela est dû à la génération aléatoire du graphe des points de rendez-vous qui n'est pas uniforme. Cette diminution explique l'anomalie du graphique de la figure 9.1. Ainsi, comme pour les réseaux formés de 40 à 60 nœuds il y a moins de points de rendez-vous par nœuds, les routes calculées passent par plus de nœuds donc il y a plus de sauts pour ces routes. En effet, plus il y a de points de rendez-vous pour chaque nœud, plus il y a de possibilités de chemins et plus il y a de chances de passer par moins de nœuds.

Vu les spécificités de notre algorithme (parcours et attentes des points de rendez-vous), nous avons aussi mesuré :

- le nombre de déplacements qu'un nœud fait, à partir du moment où il reçoit un message, jusqu'à ce qu'il arrive dans le point de rendez-vous approprié pour transmettre ce message ;

– le nombre de déplacements qu'un nœud fait pour aller dans un point de rendez-vous où l'attend un autre nœud avec un message pour lui.

Pour ces calculs, tous les nœuds envoient des messages vers tous les autres nœuds. Ces mesures sont dépendantes de notre architecture et sont faites afin de déterminer si les déplacements des nœuds sont ou non importants. La figure 9.3 représente la moyenne du nombre de déplacements qu'un nœud du réseau fait pour aller dans le point de rendez-vous approprié et transmettre le message. Cela est fait pour chaque nœud qui achemine un message et pour toutes les transmissions possibles dans le réseau.

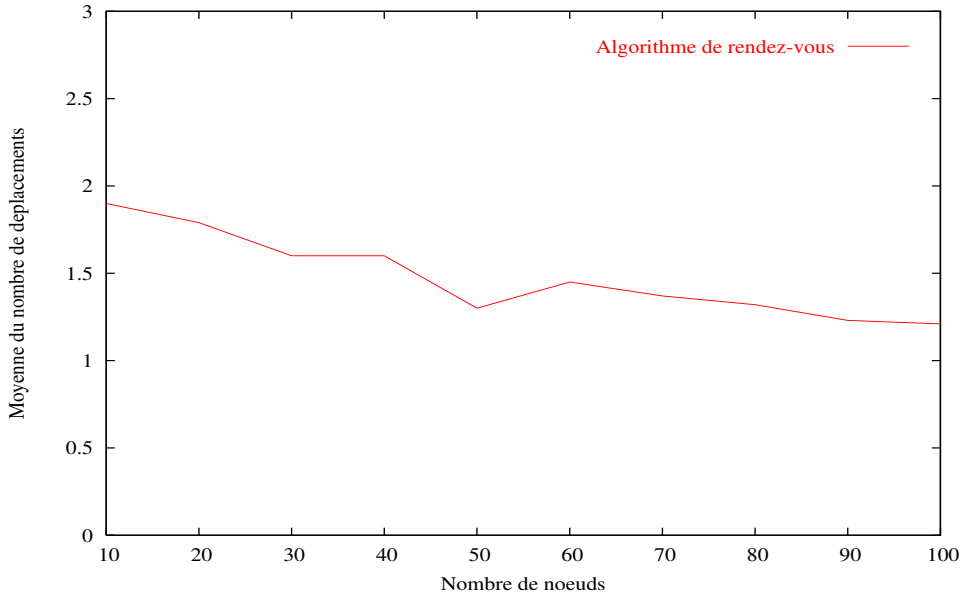


FIG. 9.3 – Le nombre de déplacements par rapport au nombre de nœuds

Enfin, une moyenne est calculée par rapport au nombre de nœuds du réseau. Elle ne dépasse pas les deux sauts, ce qui est un bon résultat car, les déplacements étant a priori gourmands en temps et en énergie, notre architecture peut être efficace.

La figure 9.4 représente la moyenne du nombre de déplacements qu'un nœud fait (quand il y a un message dans le réseau) pour aller dans un point de rendez-vous et prendre un message qu'il doit acheminer. Le calcul est fait pour chaque nœud qui participe au routage d'un message et pour toutes les

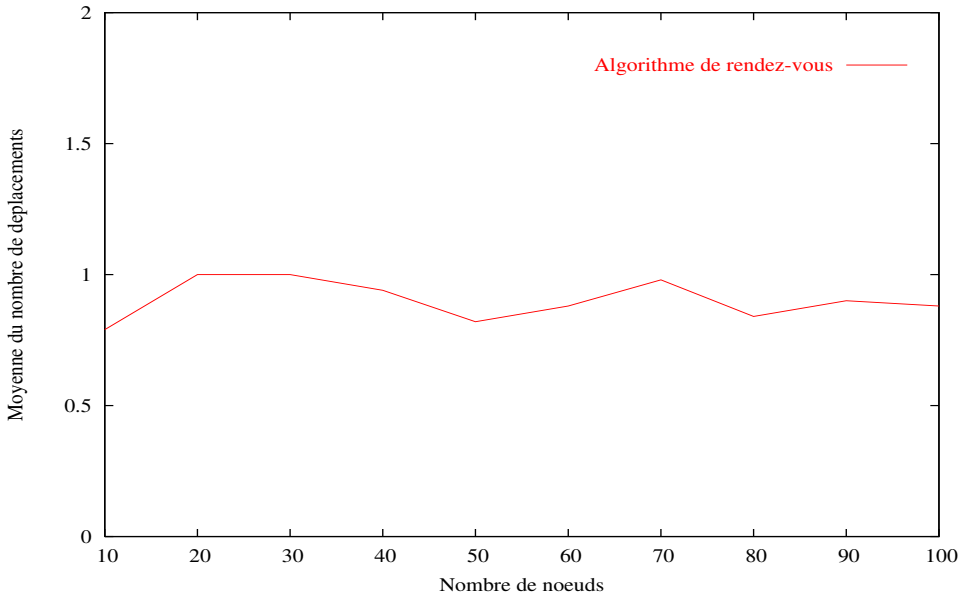


FIG. 9.4 – Le nombre de déplacements par rapport au nombre de nœuds

transmissions possibles dans le réseau, c'est-à-dire pour toutes les transmissions de chaque nœud vers tous les autres nœuds. Cette moyenne, présentée par rapport au nombre de nœuds du réseau, est d'environ un déplacement. Ceci signifie que les mouvements des nœuds sans message à acheminer ne sont pas très importants.

9.1.2.3 Tests sur le délai de bout en bout

Un autre facteur important pour le routage est le délai de bout en bout de transmission des messages. Nous avons lancé des tests et effectué des mesures pour comparer notre algorithme avec une diffusion lancée dans le réseau. Les mesures concernent deux valeurs :

- le délai de bout en bout en fonction de la vitesse des nœuds. Dans ce cas le nombre de nœuds dans le réseau était fixé à 50 ;
- le délai de bout en bout en fonction du nombre de nœuds. Dans ce cas la vitesse des nœuds était constante et égale à 5m/s.

Pour tester notre algorithme, chaque nœud envoie un message vers un autre nœud. Pendant une transmission de message, on mesure le délai de bout en bout de cette transmission. Ce test est fait pour tous les nœuds du réseau. Enfin, la moyenne est calculée pour les $n \times (n-1)$ (n est le nombre de nœuds du réseau) transmissions. Tandis que pour le cas de la diffusion, cette moyenne est calculée pour les n transmissions.

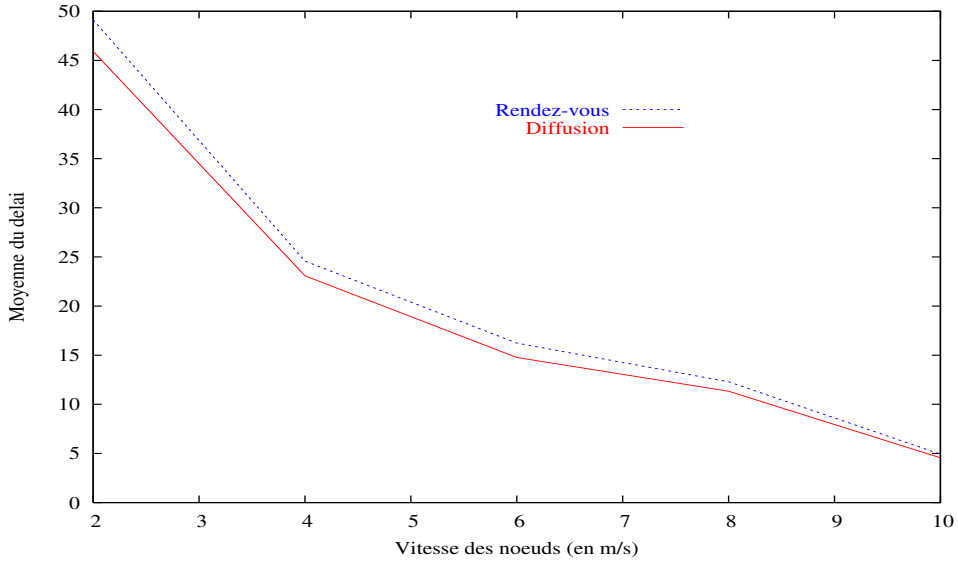


FIG. 9.5 – Le délai de bout en bout par rapport à la vitesse

La figure 9.5 représente le délai de bout en bout de la transmission d'un message en fonction de la vitesse des nœuds. Elle montre que le délai de transmission d'un message selon l'algorithme de rendez-vous est un peu plus grand que dans le cas de la diffusion. Mais avec l'augmentation de la vitesse des nœuds les deux courbes ont tendance à se rejoindre.

Dans la figure 9.6 les courbes représentent le délai de bout en bout en fonction du nombre de nœuds du réseau. Dans ce cas, le résultat montre que l'algorithme des points de rendez-vous peut ne pas être optimal pour des réseaux avec un grand nombre de nœuds, même si la différence entre les deux courbes n'est pas très grande.

Le fait que pour les deux cas le délai de bout en bout de la transmission des messages soit plus grand pour l'algorithme des points de rendez-vous est dû aux calculs des plus courts chemins. Pour ces calculs on ne considère pas

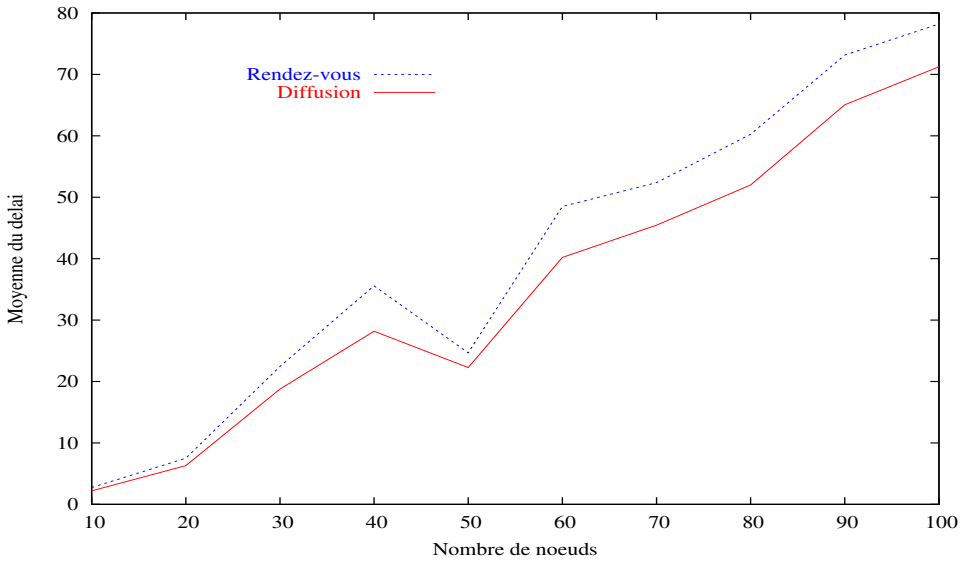


FIG. 9.6 – Le délai de bout en bout par rapport au nombre de nœuds

les temps d'attente dans les points de rendez-vous. Rappelons ici que, selon l'algorithme, chaque nœud attend son voisin dans un point de rendez-vous pour lui transmettre le message. Pour le graphique de la figure 9.6 on constate un phénomène surprenant pour les deux courbes. En effet, pour des réseaux formés de 40 à 60 nœuds le délai de bout en bout de la transmission des messages se stabilise (c'est le « trou » dans les courbes).

Cette anomalie peut être expliquée par le graphique de la figure 9.2. Comme pour les réseaux formés de 40 à 60 nœuds la plate-forme de tests à généré moins de points de rendez-vous par nœuds, alors il y aura moins de temps d'attente dans le points de rendez-vous du réseau, donc le délai diminue.

9.1.2.4 Tests sur l'efficacité de la transmission des messages

Selon le mécanisme de rendez-vous, chaque nœud parcourt ses points de rendez-vous dans l'ordre. Dans chaque point de rendez-vous, un nœud attend son voisin avant de continuer son déplacement vers le point suivant. Ainsi, un nœud peut attendre ou pas dans un point de rendez-vous. Cela dépend s'il est le premier ou le deuxième à arriver. Ces temps d'attente dans les points

de rendez-vous peuvent influencer le délai de bout en bout de la transmission des messages.

Nous avons effectué des tests pour évaluer l'efficacité de la transmission des messages. Pour cela, nous considérons tous les nœuds qui acheminent un message et nous évaluons le temps que ces nœuds (qui ont un message) passent en attendant les autres nœuds pour leur transmettre le message. Alors l'efficacité est calculée comme suit :

$$\textit{Efficacité} = 1 - \frac{\textit{temps perdu en attente}}{\textit{delai de bout en bout}}$$

La figure 9.7 présente cette efficacité par rapport au nombre de nœuds dans le réseau.

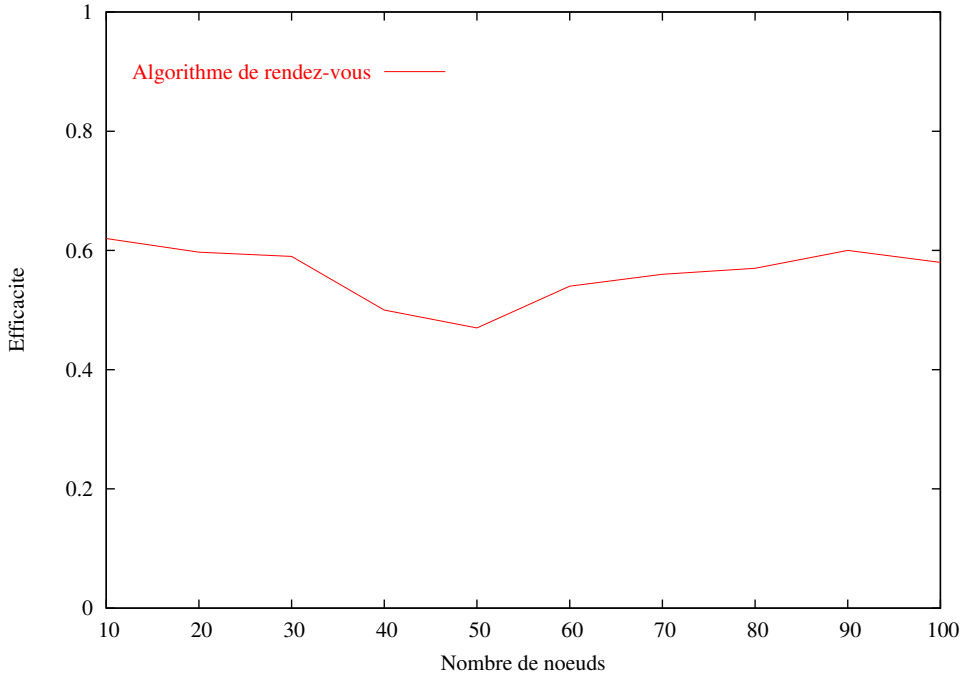


FIG. 9.7 – L'efficacité

Cette figure montre que l'efficacité de la transmission des messages ne dépend pas du nombre de nœuds dans le réseau. Pour les communications qui sont faites selon notre algorithme, elle est égale à $0,6$ ou (60%).

9.2 Évaluation par implantation

L'implantation est la deuxième étape de la validation de notre travail. Nous avons voulu voir si les algorithmes décrits dans les section précédentes peuvent être implémentés sur une plate-forme réelle utilisant des dispositifs avec des ressources limitées. Pour cela, nous avons utilisé un ensemble de robots. Cette implantation vérifiera aussi la faisabilité de notre solution.

9.2.1 Programmation des robots

Dans le passé, la robotique était plutôt limitée aux amateurs, aux chercheurs de grandes sociétés d'électronique, ou aux universités. L'introduction du *Robotics Invention System* par la société *LEGO Mindstorms* a aidé à rendre la robotique accessible au grand public. Le *LEGO Robotics Invention Kit* propose tous les composants nécessaires pour concevoir, construire et programmer un robot entièrement fonctionnel en peu de temps, exigeant une petite expérience antérieure [67, 48].

Le cerveau du LEGO Mindstorms est une brique qui inclut un microprocesseur programmable, appelée RCX. Le RCX est une évolution de la brique programmable du MIT (*Massachusetts Institute of Technology*), qui a été à l'origine développée de sorte qu'un programmeur puisse concevoir et construire de petits dispositifs mobiles qui fonctionnent indépendamment sans exiger une connexion physique à un ordinateur central. Les programmes sont créés sur un ordinateur et téléchargés à la brique, par un lien infrarouge. Une fois que le programme est téléchargé dans la brique, le dispositif mobile peut fonctionner indépendamment de l'ordinateur central.

Au cœur de la brique RCX se trouve un microprocesseur Hitachi H8 3297 avec 32K de RAM externe. Il y a également une mémoire ROM de 16K contenant les pilotes utilisés lorsque le RCX est mis en marche pour la première fois. La mémoire RAM est divisée en sections : 16K pour le progiciel (*firmware*), 6K pour stocker les programmes des utilisateurs, et 10K pour interpréter le byte code et assurer l'exécution du programme. Le RCX peut être programmé pour commander simultanément jusqu'à trois moteurs, trois capteurs, et une interface infrarouge de communication.

Le RCX peut être programmé à l'aide d'un langage de programmation graphique, legOS, qui abstrait les détails de la programmation et fournit une interface intuitive, où l'utilisateur peut assembler les composants à la souris et produire le programme.

Les progiciels par défaut peuvent être remplacés avec une variété de systèmes d'exploitation alternatifs, liés à différents langages de programmation. Dans notre cas, nous avons remplacé le système d'exploitation original par le système leJOS (*lego Java Operating System*) qui nous permet de développer des programmes en Java et les faire exécuter dans les robots.

Pour communiquer, le RCX utilise une interface infrarouge avec une fréquence de 38kHz. Cette communication est faite par des paquets dont le format est :

0x55 0xff 0x00 D1 ˜D1 D2 ˜D2 ... Dn ˜Dn C ˜C

où la partie en gras est le début de chaque message. D1,...,Dn sont des nombres en format hexadécimal qui forment le message. Chaque numéro est suivi par son complément ˜Dn. C = D1 + D2 + ... Dn est la somme de contrôle *checksum*. La partie des données de chaque message est précédée d'un opcode [57] qui indique si l'on a à faire à une requête (du PC au RCX) ou à une réponse (du RCX au PC). D1, D2, ..., Dn représentent les données qui sont codées en format hexadécimal pour être envoyées à travers l'interface infrarouge.

9.2.2 lego Java Operating System (leJOS)

leJOS [8] est un petit système d'exploitation *open source* d'environ 16 KB, basé sur Java et utilisé pour programmer les briques RCX. Le logiciel est principalement composé de trois parties :

- une machine virtuelle pour l'exécution du bytecode Java téléchargé sur le RCX ;
- une API pour la programmation du RCX ;
- des outils logiciels additionnels.

Étant un système basé sur Java, leJOS supporte :

- la programmation orientée objet ;
- les threads préemptives ;
- les tableaux multi-dimensionnels ;
- la récursivité ;
- la synchronisation ;
- les opérations sur les flottants ;

- les chaînes de caractères (String) ;
- la classe *java.lang.Math*.

Nous avons implanté les algorithmes en Java et ensuite nous les avons téléchargés du PC aux RCXs *via* une tour infrarouge connectée sur le port série du PC. Le *package* Java que nous avons conçu est composé d'une dizaine de classes. Dans ces classes, il y a des méthodes définissent les mouvements des robots et d'autres qui définissent la communication entre eux.

Le compilateur *lejosc* fait un appel au compilateur de la machine virtuelle Java. Quand un programme est téléchargé dans le RCX, il opère de manière indépendante du PC. Ainsi, tout le bytecode nécessaire dont un programme leJOS peut éventuellement avoir besoin, doit être localisé et téléchargé d'un coup dans les 6 KB disponibles pour l'utilisateur. L'outil *lejos* fournit par leJOS collecte tout le bytecode qu'un programme peut utiliser et le met dans une image temporaire qui est téléchargée dans le RCX. Pour les langages de programmation, cette technique est connue sous le nom de *static linking* ce qui n'est pas la philosophie classique de Java (*dynamic linking*).

leJOS offre des méthodes qui servent à initialiser les capteurs, mettre les moteurs en marche et les guider. Il contient aussi des méthodes qui servent pour la communication (*isPacketAvailable(), sendPacket() readPacket()*). Une de nos méthodes, utilisée pour l'implantation des algorithmes dans les RCX et qui sert à initialiser les capteurs est représentée ci-dessous :

```
public Robotrcx() {
  // Initialiser le premier capteur comme capteur de lumière
  Sensor.S1.setTypeAndMode(SENSOR_TYPE_LIGHT, SENSOR_MODE_PCT);
  Sensor.S1.activate();
  // Initialiser le troisième capteur comme capteur de lumière
  Sensor.S3.setTypeAndMode(SENSOR_TYPE_LIGHT, SENSOR_MODE_PCT);
  Sensor.S3.activate();
  // Initialiser le deuxième capteur comme capteur de pression
  Sensor.S2.setTypeAndMode(SENSOR_TYPE_TOUCH, SENSOR_MODE_BOOL);
  Sensor.S2.activate();

  // Ajouter un listener dans le capteur S1 pour agir en cas de
  // changement de valeur
  Sensor.S1.addSensorListener(new SensorListener() {
    public void stateChanged (Sensor src,int oldVal,int newVal){
```

```
      newVal = Sensor.S1.readValue();
        if (newVal <= 42) {   //40,42,45 pour la couleur noire
           detectS1 = true;
           Sensor.S1.setPreviousValue(0);
        }
        else {
          detectS1 = false;
        }
      }
  });
  // Ajouter un listener dans le capteur S3 pour agir en cas de
  // changement de valeur
  Sensor.S3.addSensorListener(new SensorListener() {
    public void stateChanged (Sensor src,int oldVal,int newVal){
       newVal = Sensor.S3.readValue();
       if (newVal <= 42) {    //40,42,45 pour la couleur noire
         detectS3 = true;
         Sensor.S3.setPreviousValue(0);
       }
       else {
         detectS3 = false;
       }
     }
  });

  Sensor.S2.addSensorListener(new SensorListener() {
     public void stateChanged (Sensor src,int oldVal,int newVal)
     { }
  });
}
```

9.2.3 Plate-forme réelle d'essais sur les robots

Pour tester l'algorithme, nous avons d'abord construit des robots capables de se déplacer et communiquer. Un de ces robots est représenté à la figure 9.8. Ces robots ont deux capteurs de lumière, deux capteurs de pression et deux moteurs. Les robots, n'ayant que 6 KB de mémoire à disposition des programmes des utilisateurs, beaucoup d'optimisations ont été faites afin de

rendre possible le téléchargement dans les robots.

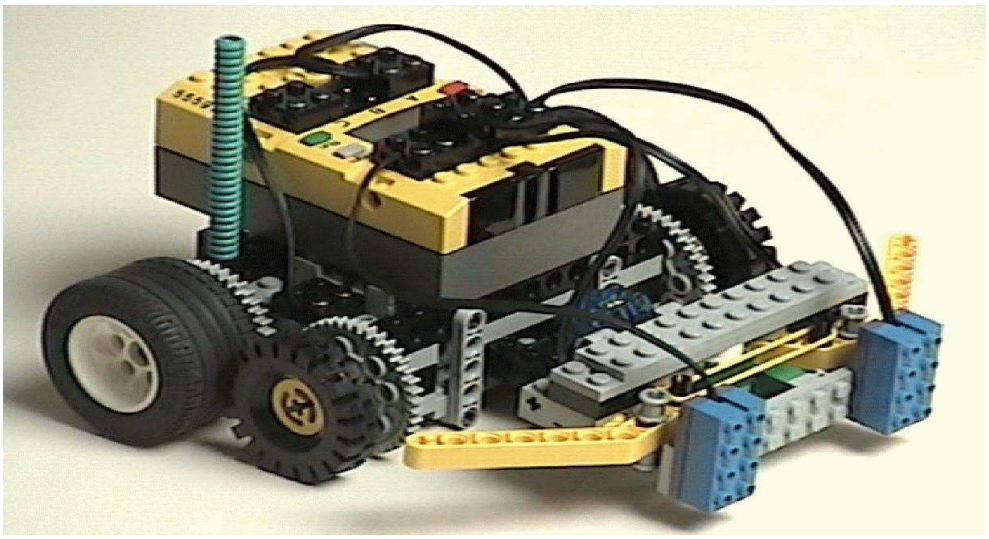


FIG. 9.8 – Le robot

Ensuite, une piste de simulation, représentée dans la figure 9.9 a été construite. Les robots suivent les lignes noires pour se déplacer d'un point de rendez-vous à l'autre. Pour lancer le test, un PC transmet un message *via* une tour infrarouge à un robot. Ensuite ce message est transmis jusqu'à sa destination *via* d'autres robots. Cette implantation nous a montré que notre approche était réaliste, même avec des robots ayant très peu de mémoire. De plus elle nous permet de conclure à la faisabilité de notre approche.

9.3 Résumé

Ce chapitre présente les tests de performances que nous avons effectués. D'abord, nous avons conçu une plate-forme de simulation en Java et implémenté nos algorithmes. Nous avons aussi implémenté une diffusion pour le comparer avec notre algorithme et présenté les résultats de cette comparaison. Les résultats montrent que notre algorithme des points de rendez-vous pourra minimiser le nombre de sauts qu'un message fait pour atteindre la destination. Cet algorithme peut aussi minimiser les déplacements faisant en sorte que les nœuds puissent économiser l'enérgie. Ensuite, l'algorithme a

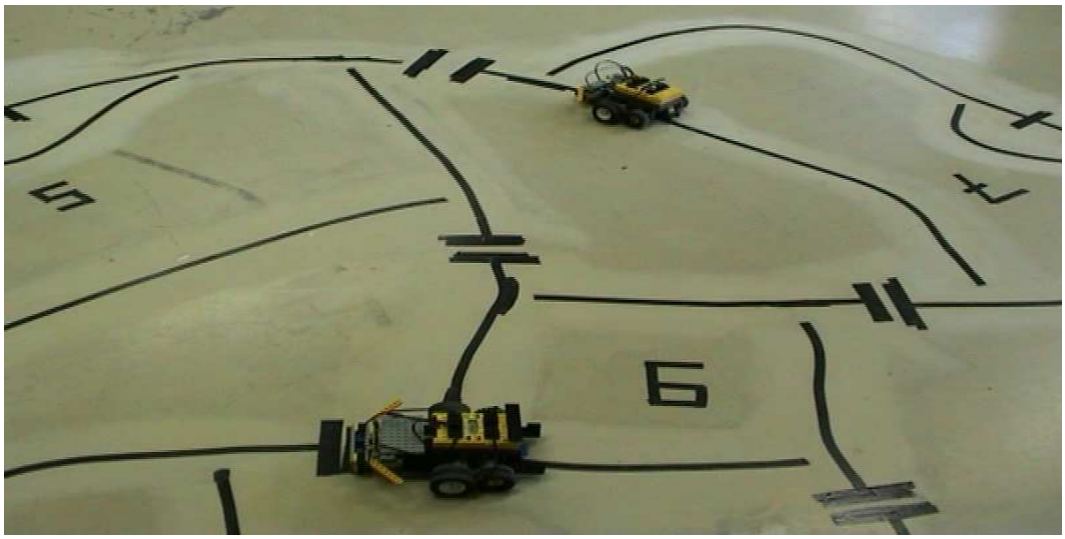


FIG. 9.9 – La piste de simulation

été implanté sur des robots spécialement conçus pour nos objectifs afin de montrer la faisabilité de notre solution.

Chapitre 10

Conclusions et perspectives

Ce chapitre présente les conclusions de cette thèse et quelques perspectives qui émergent pour la suite.

10.1 Conclusions

Les énormes progrès de la communications sans fil et de l'informatique omniprésente (*ubiquitous computing*), combinés avec l'évolution rapide des appareils et des dispositifs mobiles, produisent de nouveaux défis et soulèvent des problèmes exigeant des interactions entre différentes couches du modèle OSI et entre les applications afin d'offrir des services mobiles avancés.

Dans cette thèse nous avons présenté la conception d'une architecture de contrôle des mouvements des nœuds mobiles. Contrairement à d'autres architectures de contrôle des mouvements, qui essayent soit d'améliorer le passage à l'échelle, soit d'assurer la connectivité du réseau, notre architecture a pour but principal d'améliorer les performances de routage d'un réseau ad hoc. Ainsi, elle assure une transmission de messages de bout en bout en temps borné. Elle améliore le nombre de sauts qu'un message fait pour atteindre la destination (routes plus courtes). De plus, elle réduit la consommation de l'énergie des nœuds et assure la connectivité du réseau. Cette architecture est distribuée dans le sens où il n'y a pas d'entité centrale définissant les mouvements de tous les nœuds. En effet, une fois la numérotation initiale établie de manière globale, chaque nœud prend ses décisions de manière individuelle et autonome.

Plus particulièrement, nous avons considéré des réseaux ad hoc de grande taille où l'étendue géographique du réseau est très grande par rapport à la zone de transmission des interfaces sans fil des nœuds mobiles. Ainsi, les nœuds du réseau sont pour la plupart du temps déconnectés. Dans ce genre de réseaux, notre solution considère le mouvement des nœuds comme un facteur positif et en tire profit afin d'établir une communication fiable et robuste dans le réseau. Ainsi, nous avons mis en place un algorithme qui établit un ordonnancement des mouvements des nœuds. Les mouvements des nœuds et les communications sont asynchrones. Plusieurs nœuds peuvent se déplacer en parallèle et plusieurs messages peuvent être acheminés en même temps dans le réseau.

L'architecture mise en place fonctionne sur la base de points de rendez-vous. Contrairement à d'autres modèles [33, 64, 44], le nôtre ne demande pas aux nœuds de connaître leur position exacte mais seulement la position approximative de leurs points de rendez-vous. Pour définir ces points de rendez-vous, nous définissons d'abord les zones où se déplacent les nœuds du réseau. Pour cela, nous avons utilisé le diagramme de Voronoï qui divise un plan avec n points référence (les nœuds du réseau) en n polygones convexes tels que chaque polygone contienne exactement un point de référence et chaque point dans un polygone est plus proche de son point de référence que de tout autre point. Les points de rendez-vous peuvent alors être définis sur le segment du diagramme de Voronoï qui sépare ces deux zones voisines. En fait dans la réalité, les points de rendez-vous sont plutôt des zones de rencontres où les nœuds peuvent se trouver à une distance suffisamment réduite pour pouvoir communiquer. Dans un environnement hautement dynamique comme celui des réseaux ad hoc, cela rend notre architecture plus robuste. Ainsi, les nœuds n'ont pas besoin de se déplacer vers des points particuliers du plan, mais vers des zones de rencontre.

Notre premier algorithme établit un ordonnancement des mouvements des nœuds. Il considère une numérotation complétement aléatoire des points de rendez-vous et initialise chaque nœud dans *son* point de rendez-vous ayant le plus petit numéro. Les nœuds se donnent rendez-vous dans les points de rendez-vous où il peuvent échanger des messages. En l'absence de panne, cet ordonnancement est sans blocage, ce que nous avons prouvé par une série de propriétés.

Grâce au mécanisme de points de rendez-vous, les nœuds utilisent une

petite zone de transmission. La réduction de la zone de transmission a pour conséquence la conservation de l'énergie et de la bande passante.

La modélisation faite pour étudier cette architecture est basée sur un graphe que l'on appelle *graphe des points de rendez-vous*. Dans ce graphe non-orienté, un sommet représente un nœud du réseau et il y a une arête entre deux sommets si et seulement si il y a un point de rendez-vous entre les nœuds représentés par ces deux sommets. L'algorithme d'ordonnancement des mouvements peut être considéré comme un algorithme de mouvement dans ce graphe. Le graphe des points de rendez-vous nous a servi pour définir le mécanisme de routage. La base de ce mécanisme repose sur les tables de routage qui contiennent une entrée pour chaque couple (*point de rendez-vous, destination*).

Le premier algorithme d'ordonnancement ne gère pas les cas où un nœud tombe en panne. Pour faire face à de telles circonstances, nous l'avons amélioré dans une version tolérante aux pannes. Ainsi, nous avons introduit une nouvelle version de l'algorithme des points de rendez-vous. Pour cela on considère les durées des tâches entre deux points de rendez-vous. Ainsi, le deuxième algorithme change du premier sur deux aspects. Premièrement, quand un nœud est le premier arrivé dans un point de rendez-vous, il attend son voisin comme dans le premier algorithme, mais si la durée réelle de la tâche ajoutée au temps d'attente dans un point de rendez-vous dépasse la durée totale prédéfinie de la tâche, alors le nœud quitte le point de rendez-vous et considère que son voisin est tombé en panne. Deuxièmement, quand un nœud quitte le point de rendez-vous avant la durée totale prédéfinie de la tâche, le reste du temps de la tâche courante est ajouté à la durée prédéfinie de la tâche suivante. Ainsi, les temps des tâches de chaque nœud entre les points de rendez-vous sont non-corrélés et les nœuds opèrent de manière asynchrone. Il faut noter que : dès que les nœuds se rencontrent dans un point de rendez-vous, il communiquent. Ainsi, les communications établies par notre algorithme sont des communications au plus tôt.

Pour évaluer les performances de cette architecture, nous avons mis en place une plate-forme de simulations et évaluer différentes métriques comme le délais de bout en bout de la transmission des messages et le nombre de sauts qu'un message fait pour atteindre sa destination. Ainsi, nous avons pu tester différentes propriétés du système mis en place avant que nous les ayons

démontrées. Cette plate-forme est composée d'un ensemble de programmes Java qui simulent totalement un réseau ad hoc. Elle est adaptée à notre système et implémente les algorithmes d'ordonnancement.

Pour les tests, nous avons comparé notre algorithme avec une diffusion dans le réseau. À la différence de notre algorithme qui exige que la communication entre deux nœuds ait lieu dans un point de rendez-vous, dans la diffusion chaque nœud communique avec tous ses voisins, c'est-à-dire qu'il communique dans tous ses points de rendez-vous. Nous avons analysé les résultats des mesures effectuées pendant ces tests. Ainsi, notre architecture se révèle efficace pour minimiser les déplacements des nœuds pendant le routage des messages en assurant des routes avec un petit nombre de sauts, ce qui est préférable. Nous avons aussi évalué le délai de bout en bout de la transmission des messages en fonction de la vitesse et du nombre de nœuds dans le réseau. Ainsi, quand la vitesse des nœuds augmente, le délai calculé pendant l'exécution de l'algorithme des points de rendez-vous diminue et se rapproche de celui calculé pour le cas de la diffusion.

10.2 Perspectives

Les réseaux ad hoc ont attiré beaucoup d'attention dans la recherche grâce à leurs capacités d'auto-configuration, à leur potentiel de déploiement rapide et à leur utilisation dans différentes applications, s'étendant de la reconnaissance et des missions militaires aux réseaux véhiculaires VANET. La mobilité crée un environnement dynamique qui pose certains défis. Les mouvements des nœuds entraînent des ruptures de liens et changements de topologies qui affectent les performances des réseaux ad hoc. Différents mécanismes sont alors nécessaires pour réagir à une telle dynamique. Notre approche, validée aussi dans d'autres études [7], crée un lien logique entre la mobilité et la performance. Ainsi, la modélisation de la mobilité devient cruciale dans l'étude et l'évaluation des protocoles de routage.

Les principales perspectives de recherche à l'issue de cette thèse peuvent se résumer aux points suivants :

- étudier le problème de transmission multicast. L'environnement des réseaux ad hoc est très dynamique et les reconfigurations sont très fréquentes. La mise en place d'un protocole multicast engendrera plus de consommation de ressources pour maintenir un arbre multicast. Ainsi,

il sera intéressant de voir si l'architecture de contrôle de mouvements que nous avons mise en œuvre avec son mécanisme des points de rendez-vous, aura un impact dans l'amélioration des problèmes de multicast ;

- faire une modélisation à base de files d'attente et estimer le délai de bout en bout des transmissions des messages. Il s'agit notamment d'introduire des modèles basés sur les chaînes de Markov qui pourront modéliser les temps de parcours entre les points de rendez-vous. Ensuite, des modèles de files d'attente comme $M/M/1$ et $M/M/c$ peuvent être utilisés pour les évaluations. Il s'agit aussi de faire une évaluation probabiliste en considérant que des probabilités sont assignés à chaque point de rendez-vous. Ainsi, si on a une communication qui n'est pas uniformément distribuée, ces probabilités serviront pour évaluer par exemple le délais de bout en bout de la transmission des messages ou estimer le temps maximum de l'exécution de l'algorithme ;

- faire une évaluation comparative des protocoles de routage ad hoc en utilisant le mécanisme des points de rendez-vous. Cette évaluation pourra se faire par simulation ;

- faire une implémentation avec des dispositifs plus réalistes que les robots comme des applications dans les VANET. Nous pensons que notre architecture de contrôle de mouvements peut trouver sa place dans des applications pratiques, par exemple dans le cadre des réseaux VANET (*Vehicular Ad hoc NETworks*) [2, 62]. Dans ce type de réseaux, certaines limitations des réseaux ad hoc sont atténuées. Les VANET peuvent rassembler des dispositifs avec de larges ressources de calcul et des capacités d'énergie suffisantes. Les modèles de mobilité sont contraints par des routes de trafic avec des restrictions de vitesse.

Table des figures

Liste des tableaux

Bibliographie

[1] Bluetooth specifications. http ://www.bluetooth.org/spec/.

[2] Dedicated short range communications home. http ://www.leearmstrong.com/dsrc/dsrchomeset.htm.

[3] Glomosim. http ://pcl.cs.ucla.edu/projects/glomosim/.

[4] The network simulator - ns-2. http ://www.isi.edu/nsnam/ns/.

[5] ANSI/IEEE Std 802.11. Wireless lan medium access control (mac) and physical layer specifications, 1999.

[6] K.N. Amouris, S. Papavassiliou, and M. Li. A position-based multi-zone routing protocol for wide area mobile ad hoc networks. In *Proceedings of the IEEE Vehicular TEchnology Conference (VTC)*, pages 1365–1369, 1999.

[7] F. Bai, N. Sadagopan, and A. Helmy. Important : A framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks. In *Proceedings of IEEE INFOCOM (The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies)*, San Francisco, March/April 2003.

[8] Brian Bangall. *Programmation avancée des LEGO MINDSTORMS*. Campus Press, 2003.

[9] Azzedine Boukerche, Sajal K. Das, and Alessandro Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks. *J. Parallel Distrib. Comput.*, 61(7) :967–995, 2001.

[10] P. Bracka. Routage dans un réseau ad-hoc des robots. Memoire de dea, IGM - Université de Marne-la-Vallée, September 2001.

[11] P. Bracka, S. Midonnet, and G. Roussel. Trajectory based communication in an ad hoc network of robots. In *Proceedings of IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2005)*, Montreal, Canada, August 2005.

[12] P. Bracka, S. Midonnet, and G. Roussel. Routage dans un réseau ad-hoc des robots. In *Proceedings of the Algotel 2002*, Mèze, France, May 2002.

[13] P. Bracka, S. Midonnet, and G. Roussel. Scheduling and routing in an ad hoc network of robots. In *Proceedings of IASTED International Conference on Computer Science and Technology*, Cancun, Mexico, May 2003.

[14] J. Broch, D. Maltz, D. Johnson, Y-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, 1998.

[15] T. Camp, J. Boleng, and V. Davis. Mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC), Special issue on Mobile Ad Hoc Networking : Research, Trends and Applications*, 2002.

[16] S. Chakraborty, D.K.Y. Yau, and J.C.S Lui. On the effectiveness of mouvement prediction to reduce energy consuption in wireless communication(extended abstract). In *Joint International Conference on Measurement and Modeling of Computer Systems*, 2003.

[17] T. Clausen. Étude comparative de protocoles de routage pour les réseaux mobiles ad hoc. Rapport de recherche 5135, INRIA, Mars 2004.

[18] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. *draft-ietf-manet-olsr-07.txt, IETF*, July 2002.

[19] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE INMIC*, Descember 2001.

[20] ETSI STC-RES10 Committee. Radio equipment and systems :. *Hign Performance Radio Local Area Network Type 1, Conformity Test*, 2000.

[21] ETSI STC-RES10 Committee. Radio equipment and systems :. *Hign Performance Radio Local Area Network Type 2, Cell Based Convergence Layer*, 2000.

[22] ETSI STC-RES10 Committee. Radio equipment and systems :. *Hign Performance Radio Local Area Network Type 2, Data Link Control (DLC) layer*, 2000.

[23] ETSI STC-RES10 Committee. Radio equipment and systems :. *Hign Performance Radio Local Area Network Type 2, Packet Based Convergence Layer*, 2000.

[24] ETSI STC-RES10 Committee. Radio equipment and systems :. *Hign Performance Radio Local Area Network Type 2, Physical Layer Specifications*, 2000.

[25] ETSI STC-RES10 Committee. Radio equipment and systems :, June 1996.

[26] S. Corson and J. Macker. Mobile ad hoc networking (manet) :routing protocol performance issues and evaluation considerations. In *RFC 2501, IETF*, January 1999. http ://www.ietf.org.

[27] Samir Ranjan Das, Charles E. Perkins, and Elizabeth E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM (1)*, pages 3–12, 2000.

[28] Edsger W. Dijkstra and J.R. Rao. Constructing the proof of Vizing's Theorem. circulated privately, February 1990.

[29] Eric W. Weisstein et al. From mathworld–a wolfram web resource. http ://mathworld.wolfram.com/DijkstrasAlgorithm.html.

[30] J.J. Garcia-Luna-Aceves and E.L. Madrga. A multicast routing protocol for ad hoc networks. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 784–792, 1999.

[31] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. In *Proceedings of IEEE GLOBECOM 2000, San Francisco, CA.*, 2000.

[32] M. Gerla, X. Hong, Y. Yi, K. Xu, and T.J. Kwon. Scalable ad hoc routing in large, dense wireless networks using clustering and landmarks,. In *Proceedings of IEEE International Conference on Communications (ICC 2002)*, New York City, April 2002.

[33] D. Goldenberg, J. Lin, A.S. Morse, B. Rosen, and Y. Yang. Towards mobility as a network control primitive. In *Proceedings of Mobihoc '04*, 2004.

[34] M. Grossglauser and D.N.C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proceedings of the IEEE INFOCOM '00*, pages 1360–1369, 2001.

[35] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. Zone routing protocol (zrp). http ://www.ietf.org/proceedings/02jul/I-D/draft-ietf-manet-zone-iarp-02.txt.

[36] Joyce El Haddad and Serge Haddad. Self-stabilizing scheduling algorithm for cooperating robots. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03)*, Tunisie, 2003.

[37] K.P. Hatzis, G.P. Pentaris, P. Spirakis, and V. Tampakas. Fundamental control algorithms in mobile networks. *SPAA archive*, 1999.

[38] I. Holyer. The np-completeness of edge colorings. *SIAM Journal of Computing*, 10 :718–720, 1981.

[39] X. Hong, M. Gerla, and C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the ACM international workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*, 1999.

[40] IETF. Mobile ad hoc networks (manet) charter. http ://www.ietf.org/html.charter/manet-charter.html.

[41] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 217–229. ACM Press, 2003.

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

[43] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 66–75, 1998.

[44] Q. Li and Rus D. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, Boston, August 2000. ACM Press.

[45] B. Liang and Z. Haas. Predictive distance-based mobility management for pcs networks. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1999.

[46] J. Lin, A.S. Morse, and B. Anderson. Multiagent rendez-vous problem. In *Proceedings of the 42nd IEEE CDC*, 2003.

[47] NASA. Jet propulsion laboratory. http ://www.jpl.nasa.gov.

[48] Swiss Federal Institute of Technology (ETH). Autonomous robot architecture projects. http ://wlab.ethz.ch/robots/index.html.

[49] LAN MAN Standards Committee of the IEEE Computer Society. Wireless lan medium access control (mac) and physical layer specifications, Spetember 1999.

[50] F. Ogier, R. Templin and M. Lewis. Topology dissemination based on reverse-path forwarding (tbrpf), February 2004. ftp ://ftp.rfc-editor.org/in-notes/rfc3684.txt.

[51] G. Pei, M. Gerla, and X. Hong. Lanmar : Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of IEEE/ACM MobiHOC 2000, Boston, MA, pp. 11-18.*, Aug 2000.

[52] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing in mobile ad hoc networks. In *ICDCS Workshop on Wireless Networks and Mobile Computing*, pages D71–D78, 2000.

[53] C. Perkins. Ad hoc on-demand distance-vector routing, March 2001. http ://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt.

[54] C. Perkins. Ad-hoc on-demand distance vector routing. *MILCOM '97 panel on Ad Hoc Networks*, November 1997.

[55] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[56] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000.

[57] Kekoa Proudfoot. Rcx internals. http ://graphics.stanford.edu/ kekoa/rcx/Opcodes.

[58] M. Sanchez. Mobility models. http ://www.disca.upv.es/misan/mobmodel.htm.

[59] C. Savarese, J.M. Rabaey, and J. Beutel. Locationing in distributed ad hoc wireless sensor networks. In *Proc. 2001 Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2001)*, volume 4, pages 2037–2040. IEEE, Piscataway, NJ, May 2001.

[60] C. Savarese, J. Rabay, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Technical Annual Conference*, June 2002.

[61] A. Savvides, C.-C. Han, and M. B. Srivastava. Dynamic fine grain localizations in ad hoc networks of sensors. In *ACM Mobicom*, 2001.

[62] R. Sengupta and Q. Xu. Dsrc for safety systems. *Intellimotion*, vol. 10(n. 4), 2004.

[63] Ivan Stojmenovic and Xu Lin. Power-aware localized routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, 12(11) :1122–1133, 2001.

[64] William Su, Sung-Ju Lee, and Mario Gerla. Mobility prediction and routing in ad hoc wireless networks. *International Journal Netw. Manag.*, 11(1) :3–30, 2001.

[65] V. Tolety. Load reduction in an ad hoc network using mobile servers. Master thesis, Colorado School of Mines, 1999.

[66] Cornell University. Jist/swans java in simulation time/scalable wireless ad hoc network simulator. http ://jist.ece.cornell.edu/.

[67] Krishnan Viswanath. Robotics : Using lego mindstorms and java. http ://today.java.net/pub/a/today/2005/02/21/robotics.html.

[68] V. Vizing. On an estimate of the chromatic class of a p-graph. *Discrete Analiz*, 3 :23–30, 1964.

[69] Eric W. Weisstein. From mathworld–a wolfram web resource. http ://mathworld.wolfram.com/Bellman-FordAlgorithm.html.

[70] Eric W. Weisstein. From mathworld–a wolfram web resource. http ://mathworld.wolfram.com/VoronoiDiagram.html.

Résumé

Un réseau ad hoc peut se voir comme une généralisation ultime d'un réseau sans fil. Les protocoles qui assurent la communication dans un réseau ad hoc doivent également prendre en compte la mobilité et la variation de la connectivité des nœuds. En fait, la recherche sur les algorithmes de routage dans les réseaux ad hoc s'est principalement concentrée sur des réseaux totalement connectés dans lesquels la mobilité des nœuds n'est pas utilisée.

Dans cette thèse, nous nous sommes intéressés à des techniques qui mettent la mobilité au service du routage dans un réseau ad hoc de grande taille. Certains inconvénients du routage dans ces réseaux sont l'information imprécise sur le routage des nœuds à longue distance, la variation de la connectivité et le surcoût généré par les protocoles de routage. Cette thèse offre une solution basée sur le contrôle de la mobilité des nœuds pour améliorer les performances des protocoles de routage d'un réseau ad hoc de grande taille. Nous pensons que le mouvement des nœuds présente une opportunité : si les nœuds se déplacent d'une manière « approprie », alors les algorithmes de routage peuvent en tirer profit. Ainsi, notre solution met en oeuvre un mécanisme de rendez-vous qui réduit, les effets de la mobilité, le surcoût dû aux mises à jour et assure une communication en temps borné dans le réseau. Une autre conséquence de ce mécanisme est la réduction de la consommation de l'énergie et de la bande passante car la zone de transmission des nœuds peut être réduite. Nous présentons deux algorithmes, dont un tolérant aux pannes et développé un simulateur pour faire des tests et des mesures de performances. De plus, nous avons montré la faisabilité de la solution par une implantation sur des robots.

Mots clés : Réseaux ad hoc, modèles de mobilité, contrôle de mobilité, routage, ordonnancement, mécanisme de rendez-vous

Abstract

Ad hoc networks present the next great challenge for distributed systems research. Protocols that support communication in ad hoc networks have to take into account the mobility of the participants and the variation in the connectivity between associated parties. In fact, most research on routing algorithms in an ad hoc network is mainly based on algorithms dedicated to completely connected networks in which the node mobility is not used.

In this thesis, we are interested on solutions which use mobility to improve routing performances. Some critical issues of routing in a large size ad hoc network are : inaccurate routing information about remote nodes, the connectivity and the protocol overhead.

This thesis proposes a control mobility architecture to improve performances of the routing protocols in a large scale ad hoc network. We think that the node mobility represents an opportunity : if nodes move in an appropriate fashion, then routing algorithms can take advantage. Thus, our architecture with its rendez-vous mechanism try to reduce mobility and routing overhead. The use of the rendez-vous mechanism reduces energy consuption and node bandwidth, because the node transmission range can be reduced. We also present two algorithms, one fault tolerant in conjuction with simulations, tests and performances mesures. We also proved the feasibility of our solution by an implementation using real rebots.

Keywords : Ad hoc networks, mobility models, mobility control, routing, scheduling, rendez-vous mechanism