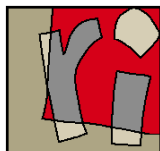


Routage multichemin et QoS dans les réseaux ad hoc mobiles

Ignacy GAWĘDZKI <i@lri.fr>

1 mars – 31 juillet, 2004

Au sein du
Groupe Réseaux du



Laboratoire de Recherche en Informatique,
Université Paris-Sud, Orsay.

Sous la direction de
Khaldoun AL AGHA <alagha@lri.fr>

et l'encadrement de
Hakim BADIS <badis@lri.fr>

Résumé

Le groupe Réseaux du LRI travaille intensivement sur le protocole QOLSR qui est une extension d'OLSR, un protocole de routage pour les réseaux ad hoc mobiles sans fil. Le stage a porté sur la mise au point d'une méthode de routage avec qualité de service (QoS) sans réservation pour le protocole QOLSR ainsi que son extension au routage multichemins. Des outils de simulation ont été développés pour pouvoir comparer les résultats de ces méthodes avec le routage QOLSR classique.

Table des matières

1	Introduction	1
2	Présentation du domaine	3
2.1	Le vocabulaire utilisé	3
2.1.1	Nœuds, sommets, liens, arêtes et arcs	3
2.1.2	Notions de métrique	3
2.1.3	Probabilité de blocage	4
2.2	Les protocoles de routage pour réseaux ad hoc	4
2.2.1	Les protocoles réactifs	4
2.2.2	Les protocoles proactifs	5
2.3	Le protocole OLSR	6
2.3.1	Présentation du protocole	6
2.3.2	OLSR et la QoS	7
3	Routage avec QoS	9
3.1	La réservation de ressources	9
3.2	Routage avec la métrique du débit disponible	10
3.2.1	Le problème de l'oscillation des flux	10
3.2.2	Le critère de changement de chemin	11
3.2.3	Le problème de collision des flux	11
3.2.4	La solution du backoff	12
3.2.5	Publication de la méthode	14
4	Routage multichemin	15
4.1	Les avantages des multichemins	15
4.1.1	Augmentation de la capacité	16
4.1.2	Augmentation de la fiabilité	16
4.2	La capacité des liens dans les réseaux sans fil	16
4.2.1	Illustration de la corrélation des débits	16
4.2.2	Contraintes linéaires de corrélation	18
4.3	Les méthodes de recherche de multichemin	19
4.3.1	Notre approche	19
4.3.2	Notre méthode finale	21

4.3.3	Exemples de résultats	23
4.4	Les outils de simulation statique	24
4.4.1	Principe de fonctionnement	24
5	Conclusions	27
5.1	Questions de performance	27
5.1.1	Calcul des ensembles corrélés	27
5.1.2	Résolution du programme linéaire	28
5.2	Perspectives	28
5.2.1	Simulations dynamiques	28
5.2.2	Simulations statiques	28
5.3	Remarques générales	28
	Glossaire	31

Chapitre 1

Introduction

Les réseaux ad hoc mobiles sans fil (ou plus simplement **réseaux ad hoc – mobile ad hoc networks** ou MANets en anglais) sont des réseaux formés spontanément par un ensemble de nœuds mobiles (ordinateurs personnels stationnaires ou portables, assistants personnels, téléphones mobiles, etc) munis de cartes réseau sans fil et situés à portée radio d’au moins une partie d’entre eux. Pour communiquer avec les nœuds situés hors de sa propre portée, un mobile doit donc passer par des nœuds intermédiaires. Cette communication à plusieurs sauts, que l’on appelle **multihop** nécessite l’utilisation de protocoles et algorithmes de routage afin de permettre à chaque nœud mobile de savoir à quel voisin (en termes de connectivité radio) transmettre les données à destination d’un nœud distant. Les réseaux ad hoc apportent une solution dans les situations où une infrastructure filaire n’existe pas ou n’est pas envisagée ou pas envisageable comme par exemple pour l’organisation des secours dans les zones sinistrées, la communication entre les unités sur un champ de bataille. Leur utilisation est aujourd’hui développée aussi pour le grand public, pour vendre des services dans les lieux publics.

Le fait que les nœuds mobiles ne sont pas contrôlés par une seule entité implique que leur mouvement est très difficilement prévisible et que par conséquent la connectivité radio change au cours du temps. On est donc en présence de réseaux dont la topologie est dynamique et dont les nœuds ont des caractéristiques particulières (ressources d’énergie et de calcul limitées). Ces spécificités des réseaux ad hoc font que les solutions de routage mises au point pour les réseaux filaires classiques ne sont pas du tout adaptées et de nouvelles techniques on dû être élaborées.

Il existe aujourd’hui plusieurs solutions apportées au problème du routage qui est l’objet de recherche active depuis une dizaine d’années. Plusieurs protocoles de routage sont actuellement en cours de standardisation à l’IETF au sein du groupe de travail “manet”¹ [1, 2, 3].

Les protocoles les plus aboutis, la plupart d’ores et déjà publiés sous forme de RFC expérimental,

¹<http://www.ietf.org/html.charters/manet-charter.html>

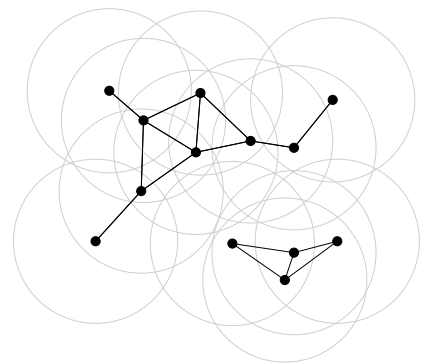


FIG. 1.1 – Un réseau ad hoc

permettent de trouver un chemin faisable en termes de connectivité vers n'importe quelle destination atteignable. La seule forme d'optimalité des chemins est la longueur en nombre de sauts, c'est-à-dire que les chemins fournis sont ceux qui offrent le plus petit nombre de sauts parmi les chemins possibles.

Cependant, les chemins optimaux en nombre de sauts ne satisfont pas nécessairement les requis de **qualité de service** (QoS) des applications pour les flux qu'elles initient. Il se peut par exemple qu'une application de transport de voix sur IP ne puisse pas utiliser le chemin le plus court en nombre de sauts simplement parce que le débit disponible sur celui-ci est inférieur au débit minimal requis pour permettre la transmission fluide des données audio. Il se peut par contre qu'il existe un chemin avec un nombre de sauts supérieur qui satisfait le requis de débit disponible. Il existe ainsi d'autres critères ou **métriques** que la recherche de chemins peut prendre en compte pour satisfaire les requis de qualité de service des applications (comme le débit disponible mais aussi le délai total de transmission, le taux de perte des paquets de données, etc).

Aujourd'hui, la recherche s'oriente donc vers le routage avec qualité de service et le groupe Réseaux du LRI travaille plus particulièrement sur une extension du protocole de routage OLSR pour permettre la recherche de chemins en prenant en compte plusieurs métriques à la fois[4, 5].

D'un point de vue plus abstrait, le routage sert à trouver le moyen d'acheminer les paquets de données correctement et, dans le cas du routage avec QoS, de le faire en respectant des contraintes supplémentaires. Rien ne dicte a priori, que le même chemin doit être emprunté par les paquets successifs sur une échelle de temps macroscopique. Il est tout à fait envisageable de diriger des paquets successifs sur plusieurs chemins différents afin de satisfaire un requis en considérant les métriques cumulées des chemins utilisés. Cette approche est déjà employée dans les réseaux filaires, principalement pour accroître le débit disponible. Dans le cas des réseaux ad hoc, le problème est encore une fois plus compliqué à cause du fait que les métriques sur les chemins ne sont pas indépendantes les unes des autres. Néanmoins il est parfois plus avantageux de router selon plusieurs chemins (ou un **multichemin**) dans les réseaux ad hoc.

Après une présentation du domaine des protocoles de routage pour réseaux ad hoc au Chapitre 2, nous verrons comment l'effort d'implémentation des nouvelles méthodes de routage avec QoS a fait remonter de nouveaux problèmes concernant le routage avec la métrique du débit disponible au Chapitre 3. Au Chapitre 4, nous verrons la mise au point d'une méthode de calcul de routes pour du routage multichemin avec QoS et enfin les conclusions et perspectives seront présentées au Chapitre 5.

Chapitre 2

Présentation du domaine

Ce chapitre présente les diverses familles de protocoles de routage existantes et insiste sur le protocole OLSR qui intéresse le groupe Réseaux du LRI en particulier. Avant de rentrer dans le vif du sujet, il est nécessaire d'introduire quelques notions utilisées tout au long de ce rapport.

2.1 Le vocabulaire utilisé

2.1.1 Nœuds, sommets, liens, arêtes et arcs

Le vocabulaire employé dans le domaine du routage est emprunté autant au réseau qu'aux graphes et certaines notions équivalentes sont utilisées alternativement.

On parle ainsi de **nœuds** ou de **stations** dans un réseau ou bien de **sommets** dans un graphe pour désigner la même chose. De même, on parle de **lien** dans un réseau comme on parle d'**arc** ou d'**arête** dans un graphe. Dans les réseaux radio, les liens ne sont pas forcément symétriques et c'est pourquoi il est nécessaire de distinguer les directions et de parler d'arcs. Par la suite, on considère qu'une arête $\{i, j\}$ est équivalente à l'ensemble des arcs (i, j) et (j, i) .

2.1.2 Notions de métrique

Une **métrique** est une mesure qualitative ou quantitative de chaque lien du réseau qui se combine avec celle des autres liens pour donner la mesure d'une route. La métrique naturellement utilisée par les protocoles de routage sans QoS est le nombre de sauts. Alors la métrique d'un lien est toujours 1 et la métrique d'une route est la somme des métriques de ses liens.

Selon le type de métrique, l'opération de combinaison pour obtenir la métrique d'une route est différente. Pour les métriques dites **additives** comme le nombre de sauts ou le délai de transmission, la mesure sur la route se fait par la somme des mesures sur les liens. Les métriques **multiplicatives**, se combinent, comme leur nom l'indique, par le produit des métriques des liens (comme le taux de perte par exemple). Il existe aussi des métriques **convexes** et **concaves** qui se combinent respectivement par le maximum et le minimum des métriques des liens (le débit disponible est une métrique concave).

Il a été prouvé [6] que rechercher une route optimale pour n métriques additives et m métriques multiplicatives à la fois est un problème NP-complet si $n + m \geq 2$. Il se peut même qu'une route op-

timale n'existe pas, bien qu'il existe des routes optimales pour les métriques séparées. Les méthodes de recherche de route sont donc basées sur des approximations ou des heuristiques.

2.1.3 Probabilité de blocage

Dans le cas du routage avec QoS, il se peut qu'un chemin satisfaisant les requis n'existe pas au moment où une application en a besoin. On appelle alors cette situation un **blocage** et la **probabilité de blocage** est simplement la mesure de la quantité moyenne de blocage qui arrivent pour un protocole de routage dans une configuration de fonctionnement donnée.

2.2 Les protocoles de routage pour réseaux ad hoc

Puisque les nœuds d'un réseau ad hoc peuvent apparaître, se déplacer et disparaître spontanément dans le réseau, aucune information préalable concernant la topologie et l'état du réseau n'est disponible. C'est par l'échange de **messages de contrôle**, c'est-à-dire de paquets spécifiques au protocole de routage (par opposition aux paquets de données habituels), que les protocoles parviennent à effectuer la recherche de chemins vers une destination. Cependant, il existe fondamentalement deux approches pour arriver à ce résultat et il en découle deux familles de protocoles.

2.2.1 Les protocoles réactifs

Les protocoles **réactifs** effectuent une recherche de chemin à la demande, lorsqu'un paquet de données doit être émis vers une destination pour laquelle un chemin n'est pas actuellement connu. Après calcul du chemin, celui-ci est conservé dans une mémoire cache tant qu'il s'avère utilisable.

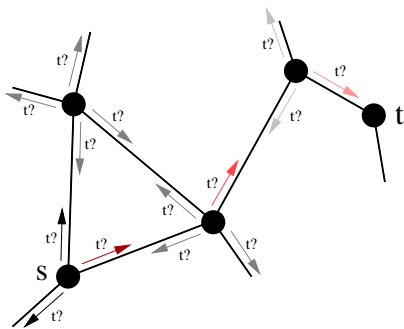


FIG. 2.1 – Envoi de sondes

Les messages de contrôle sont essentiellement des **sondes** qu'un nœud source diffuse à tous ses voisins et que ces derniers rediffusent à leur tour. Le réseau est ainsi **inondé** de sondes qui enregistrent sur leur passage les adresses des nœuds par lesquelles elles passent. Lorsque la destination reçoit la sonde, elle extrait la liste des nœuds par lesquels elle est passée et la renvoie vers la source dans un autre message de contrôle par le chemin inverse de la sonde. Cette réponse parvient directement à la source (l'inondation n'est plus nécessaire puisque le chemin est déjà connu) qui en extrait le chemin à utiliser.

Si la topologie change au cours du temps, d'autres messages de contrôle sont émis par les nœuds intermédiaires vers la source pour avertir celle-ci qu'il faut effectuer une recherche

de chemin à nouveau.

Cette approche a l'avantage d'être simple et de ne pas nécessiter de grandes ressources en mémoire sur les nœuds intermédiaires. En effet la façon naturelle d'acheminer les paquets vers leur destination est d'utiliser ce que l'on appelle le **routage par la source** (source routing), c'est-à-dire que la source définit le chemin que les paquets de données doivent emprunter et rajoute cette information dans leur en-tête. Cette technique s'oppose au routage **saut-par-saut** (hop-by-hop) où chaque

nœud utilise des tables de routage pour déterminer à quel voisin transmettre un paquet pour qu'il arrive à destination.

Un réseau est dit **dense** quand, au même titre que pour les graphes, la probabilité d'existence d'un lien entre deux nœuds quelconques est élevée. Dans ce type de réseau, l'utilisation de l'inondation "naïve" est très coûteuse et le nombre de retransmission croît tellement vite que le réseau est vite saturé de messages de contrôle.

Le protocole AODV[1] est un exemple de protocole réactif dans lequel l'inondation est limitée par la réutilisation des messages des autres nœuds. Le protocole DSR[3] est un exemple de protocole réactif qui utilise le routage par la source.

2.2.2 Les protocoles proactifs

Les protocoles **proactifs** utilisent l'échange régulier de messages de contrôle pour maintenir en chaque nœud des **tables de routage** (qui associent à chaque destination ou groupe de destinations un voisin direct par lequel les paquets doivent être relayés) vers toute destination atteignable depuis celui-ci. Ces tables sont maintenues même quand les routes ne sont pas utilisées, cette approche permet de disposer d'une route vers chaque destination immédiatement au moment où un paquet doit être envoyé.

Lorsque la topologie du réseau change, les tables de routage sont modifiées en conséquence et aucune "réparation" (action explicite d'envoi de messages de contrôle spécifiques) de route n'est nécessaire.

Il existe plusieurs approches pour le routage proactif, dont le routage par **vecteur de distance** et par **état de lien** sont les plus significatives.

Routage par vecteur de distance

Dans cette approche, les messages de contrôle sont échangés entre voisins directs¹ uniquement et contiennent les entrées des tables de routage des nœuds émetteurs². Ainsi un nœud établit ses tables de routage en comparant le contenu des tables de ses voisins. Si par exemple un voisin A fournit une route vers D avec un nombre de sauts de n , alors les voisins de A peuvent au pire fournir une route vers D à $n + 1$ sauts et donc éventuellement remplacer leurs routes existantes si elles offrent une distance supérieure à $n + 1$.

Cette technique a été initialement mise au point pour maintenir des routes minimales en nombre de sauts. Elle s'adapte bien à toute métrique additive ou multiplicative, puisqu'elle permet d'effectuer un calcul distribué de plus court chemin. Pour d'autres types de métriques (comme le débit disponible, métrique convexe), cette approche n'est pas naturelle.

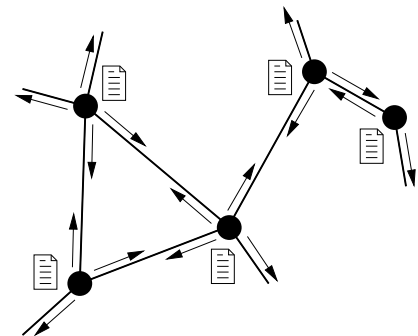


FIG. 2.2 – Maintien des tables de routage

¹C'est-à-dire entre nœuds à portée radio mutuelle.

²Ici les tables contiennent, en plus des adresses des destinations, la distance en nombre de sauts.

Routage par état de lien

Dans cette approche, les messages de contrôle sont diffusés par les nœuds à tout le réseau et contiennent la topologie locale de l'émetteur, c'est-à-dire la liste de ses voisins. On parle alors de messages topologiques. En regroupant les topologies locales de tous les nœuds du réseau, un nœud peut reconstituer la topologie globale et utiliser cette information pour appliquer une méthode de calcul de route appropriée.

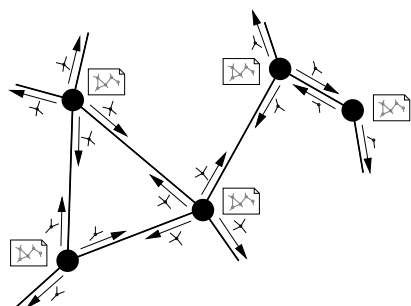


FIG. 2.3 – Messages topologiques

L'avantage majeur de cette approche est de permettre à chaque nœud, par l'ajout d'informations supplémentaires sur l'état du réseau dans les messages topologiques, de maintenir non seulement la topologie du réseau (c'est-à-dire les informations de connectivité entre les nœuds), mais encore de connaître l'état global du réseau et donc de rendre le routage avec QoS possible naturellement.

Le revers de la médaille est que la diffusion des messages topologiques à tout le réseau ne passe pas bien à l'échelle et donc cette technique a longtemps été négligée par les chercheurs.

2.3 Le protocole OLSR

2.3.1 Présentation du protocole

Optimized Link State Routing est un protocole par état de lien qui utilise une technique optimisée pour la diffusion des messages topologiques.

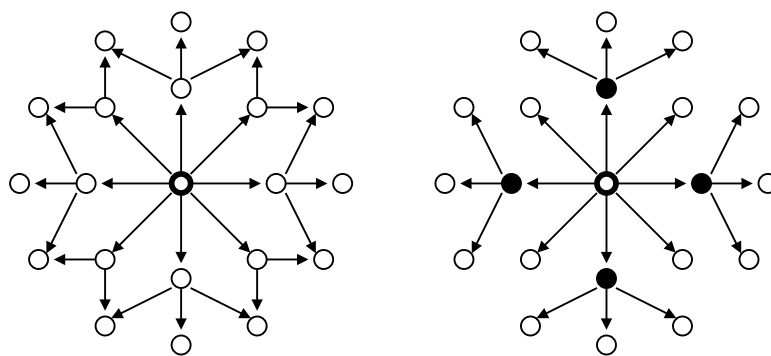


FIG. 2.4 – Diffusion par inondation (à gauche) et diffusion optimisée (à droite)

La solution consiste à ne permettre qu'à un sous-ensemble de voisins de retransmettre les messages (voir FIG. 2.4). Ces voisins sont appelés les **relais multipoint** ou **MPRs** (multipoint relays). Chaque nœud effectue la sélection de ses MPRs en se basant sur la connaissance de son voisinage à deux sauts. L'ensemble des MPRs doit être le plus petit possible, tout en assurant que la diffusion par leur intermédiaire permet d'atteindre le voisinage à deux sauts dans sa totalité. Le problème qui

consiste à trouver le plus petit ensemble de MPRs est analogue au problème de la recherche d'ensemble dominant minimal dans un graphe, qui est connu pour être NP-complet. Dans OLSR, les nœuds appliquent une heuristique qui permet de se rapprocher de l'ensemble minimal dans la majeure partie des cas.

Découverte de voisinage

Chaque nœud émet périodiquement des messages appelés **HELLO** qui contiennent essentiellement la liste des liens connus vers les voisins directs. La fonction de ces messages est multiple. Ils servent tout d'abord à détecter les voisins directs et la qualité des liens vers ceux-ci, à savoir s'ils sont symétriques ou asymétriques. Comme chaque nœud y publie la liste de ses voisins, il est possible pour un nœud d'acquérir des informations sur son voisinage à deux sauts. Par ailleurs, une fois qu'un nœud a effectué la sélection de ses MPRs, il indique dans ses messages HELLO lesquels de ses voisins sont ses MPRs. Ceci permet à un nœud de savoir quels voisins l'ont choisi comme MPR, autrement dit de constituer son ensemble de **MPR-selectors**.

Diffusion optimisée

Lorsque qu'un nœud reçoit un message de contrôle OLSR, il le traite et ne le transmet que si l'émetteur du message (l'adresse source du message, qui peut être différente de l'adresse de l'émetteur si le message a été généré par un nœud distant) appartient à l'ensemble des MPR-selectors.

Cette technique permet de diffuser des messages dans tout le réseau en évitant la saturation [7].

Les messages topologiques

Les messages topologiques, appelés **TC** (topology control) ne sont émis par un nœud qu'à condition que son ensemble de MPR-selectors n'est pas vide, c'est-à-dire qu'il est MPR d'un de ses voisins. Les messages contiennent la liste des MPR-selectors du nœud. Les nœuds du réseau reconstituent donc une topologie globale mais partielle, puisque tous les nœuds atteignables sont connus, mais pas tous les liens. Cette topologie partielle est néanmoins suffisante pour calculer des chemins minimaux en nombre de sauts vers toute destination.

Le fait de ne permettre qu'aux MPRs de générer des messages TC permet de limiter la quantité de messages diffusés dans le réseau et le fait de ne diffuser que la liste des MPR-selectors permet de limiter la taille des messages.

Les changements topologiques

À chaque changement de topologie, le calcul des routes vers toutes les destinations est déclenché pour mettre à jour les tables de routage. Par ailleurs, lorsque son ensemble de voisins directs ou à deux sauts change, un nœud doit effectuer la sélection de ses MPRs à nouveau.

2.3.2 OLSR et la QoS

Tel que décrit dans le RFC [8], OLSR ne prévoit pas de calculer des chemins avec d'autres métriques que le nombre de sauts. La nature link-state du protocole permet néanmoins de rajouter

naturellement d'autres critères pour la recherche des chemins. Le fait de travailler directement sur des graphes de topologie (globale partielle dans le cas présent) permet beaucoup de liberté de choix quant à la méthode de calcul de chemin.

Le RFC laisse la possibilité de rajouter de nouveaux types de message qui sont retransmis de façon optimisée. Afin de permettre aux nœuds de calculer des chemins avec des métriques QoS, il suffit de rajouter les informations de QoS soit dans les messages TC directement (en cassant la compatibilité avec OLSR), soit dans des messages d'un nouveau type. Le graphe de topologie partielle des nœuds est alors augmenté pour contenir les informations de QoS sur les arcs et/ou les sommets.

QOLSR : La QoS avec OLSR au LRI

Le groupe Réseaux du LRI travaille notamment sur des méthodes de calcul de chemins selon plusieurs métriques simultanément [4, 5, 9]. Une autre extension de QOLSR est la modification de la méthode de sélection des MPRs pour garantir l'optimalité en débit disponible des routes calculées sur le graphe de topologie partielle.

L'extension naturelle et intuitive d'OLSR pour l'ajout de la QoS est que les routes avec contraintes de QoS sont recalculées à chaque changement d'état des paramètres réseaux sur les liens en plus des changements purement topologiques. Nous allons voir que cette approche révèle des problèmes d'implémentation plus fondamentaux.

Chapitre 3

Routage avec QoS

Le groupe Réseaux dispose d'une implémentation modulaire en C++ du protocole OLSR¹ que j'ai écrite l'an dernier, pendant mon stage de fin d'école. Cette implémentation a été dirigée dès le début par la possibilité de rajouter et modifier des fonctionnalités dans le futur. C'est bien sûr la perspective d'implémenter les méthodes de calcul de chemin multimétriques qui était présente dès le départ.

Les activités de recherche du groupe Réseaux sur le routage multimétrique portaient jusqu'à présent sur les méthodes de calcul de chemin étant donné un graphe topologique. Les détails d'implémentation ont été mis de côté en attendant de disposer de méthodes fiables et éprouvées en simulation.

C'est en voulant implémenter des versions successives du routage avec QoS que nous nous sommes rendus compte d'un problème fondamental concernant le routage avec la métrique du débit disponible.

3.1 La réservation de ressources

Dans les réseaux filaires, qui, contrairement aux réseaux ad hoc, sont stables dans le temps, les techniques de routage avec QoS utilisent des mécanismes de **réservation de ressources**. Les nœuds source émettent un message de contrôle vers la destination et sur son passage, les nœuds intermédiaires réservent leurs ressources pour le flux à venir. Si les nœuds intermédiaires ne disposent pas de ressources suffisantes pour assurer le niveau de QoS demandé, un message de contrôle avertit la source que la requête n'est pas possible. Lorsque la source décide que le flux est terminé et que les ressources ne sont plus utilisées, elle émet un message de contrôle vers la destination en vue de libérer les ressources précédemment réservées.

La nature dynamique des réseaux ad hoc fait qu'une telle technique ne donnerait pas de bons résultats en termes de performances, puisque des messages de contrôles devraient être émis à chaque changement d'état du réseau (topologique et/ou QoS) pour garantir que les ressources sont toujours réservées et que les ressources inutiles sont bien libérées. Cette approche n'est vraiment pas naturelle, d'autant plus qu'on s'est placé dans le cadre d'un protocole proactif.

¹Voir sur <http://qolsr.lri.fr/code>

Il est pourtant admissible de se passer de réservation de ressources si les mécanismes proactifs sont suffisamment évolués.

3.2 Routage avec la métrique du débit disponible

Il s'agit de calculer des chemins qui satisfont des contraintes de débit disponible minimal. En supposant que l'on dispose d'une méthode convenable pour calculer les chemins avec contrainte de débit disponible, il apparaît deux problèmes importants à résoudre lorsque l'on veut utiliser cette méthode dans le cadre d'un protocole de routage proactif sans réservation de ressources.

3.2.1 Le problème de l'oscillation des flux

Supposons qu'un nœud source initie une émission de flux de paquets vers une destination. Il applique la méthode de calcul de chemin avec contrainte de débit disponible et si l'état du réseau le permet, il trouve un chemin praticable. Il commence à émettre les paquets en direction de la destination. Les nœuds intermédiaires successifs appliquent la même méthode.

Comme les paquets représentent un certain trafic, la mesure de débit disponible décroît au moins pour les liens du chemin emprunté par les paquets du flux. Par conséquent, les nœuds émettent des messages topologiques pour indiquer le changement d'état du réseau. À la réception de ces messages, les nœuds changent leur connaissance partielle du réseau et relancent le calcul des chemins pour les flux QdS.

Alors trois scénarii peuvent se produire :

1. Le débit disponible sur les liens du chemin emprunté par le flux courant est toujours supérieur au débit minimal requis et la méthode de calcul trouve toujours le même chemin.
2. Le débit disponible sur les liens du chemin emprunté par le flux courant est inférieur au débit minimal requis mais un chemin praticable existe encore.
3. Le débit disponible sur les liens du chemin emprunté par le flux courant est inférieur au débit minimal requis et aucun autre chemin praticable n'existe.

Dans le cas 1, rien de remarquable ne se produit et le flux continue de passer comme prévu.

Dans le cas 3, la source doit interrompre l'émission de son flux puisqu'elle constate que l'état du réseau ne permet plus d'assurer la QdS.

Dans le cas 2, la source et les nœuds intermédiaires sont susceptibles de "dévier" le flux vers un autre chemin. Alors l'état du réseau change à nouveau, puisque les liens empruntés précédemment ne le sont plus et des messages topologiques sont émis en conséquence. Un des trois scénarii se produit à nouveau et ainsi de suite.

Bien que les conséquences du cas 3 soient pour le moins dérangeantes, celles du cas 2 produisent le pire effet sur le réseau que nous avons baptisé **oscillation**. En effet, les nœuds calculent continuellement de nouveaux chemins et émettent des messages topologiques (FIG. 3.1), le tout complètement inutilement. C'est ce cas qui nous a guidés dans la recherche d'une solution.

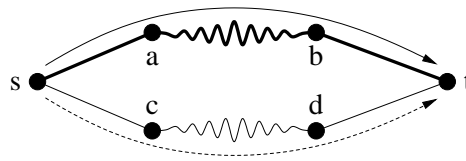


FIG. 3.1 – L’oscillation d’un flux : s envoie alternativement son flux par a et par c , et l’état du réseau change sans arrêt.

3.2.2 Le critère de changement de chemin

Le problème des cas 2 et 3 est le même : en calculant un chemin pour un flux, un nœud est incapable de distinguer la part de trafic induite sur les liens par ce flux des parts induites par les autres flux.

La solution consiste donc à ce qu’un nœud recalcule un chemin pour un flux QdS si et seulement si le fait de garder le chemin courant ne permettrait pas de garantir le niveau de QdS. Pour ce faire, chaque nœud doit conserver dans son graphe topologique partiel les étiquettes des flux sur les arcs appartenant aux chemins qu’ils empruntent couramment. Si tous les nœuds ont la même connaissance partielle du réseau, alors on peut garantir que le chemin calculé par un nœud pour un certain flux est bien celui que les paquets vont parcourir.

À chaque changement d’état du réseau (topologique et/ou QdS), chaque nœud applique ce changement sur son graphe topologique et peut constater directement quels flux sont concernés par le changement. Si le changement concerne uniquement les mesures de QdS des liens et pas la topologie du réseau, alors seuls les chemins des flux passant par les liens en question doivent être considérés pour le recalcul.

Le nœud applique ensuite ce que nous appelons le **critère de changement de chemin** et qui consiste à rajouter sur les arcs du chemin courant le débit effectivement émis pour le flux en question, à retirer l’arc liant le nœud suivant courant au nœud courant et à vérifier s’il existe un chemin satisfaisant la contrainte de débit disponible depuis le nœud suivant courant. Si la dernière étape donne une réponse positive, alors le nœud suivant est conservé ; si la réponse est négative, alors un nouveau chemin est calculé depuis le nœud courant. Dans les deux cas, les étiquettes des flux sur les arcs du graphe topologique partiel doivent être mises à jour pour refléter le chemin effectif que vont maintenant emprunter les paquets du flux.

Remarquons que le critère de changement de chemin est aussi bien appliqué dans le cas d’un changement topologique, puisqu’alors le débit mesuré pour un flux doit être ajouté au débit disponible sur les arcs du graphe topologique avant l’application du changement. Ceci assure que les débits disponibles des arcs ne sont pas “faussement” diminués par le flux lui-même.

Le critère de changement de chemin permet de décider de recalculer un chemin quand cela est vraiment nécessaire, ce qui conserve au maximum la stabilité du réseau.

3.2.3 Le problème de collision des flux

Un autre problème provoqué par le manque de réservation de ressources est dû à ce que deux nœuds ou plus, ayant la même “vue” instantanée de l’état du réseau, peuvent initier des flux qui vont

concourir pour une même ressource, c'est-à-dire qu'ils vont passer par les mêmes arcs pensant être les seuls à utiliser leurs débits. Nous appelons ce phénomène la **collision** des flux.

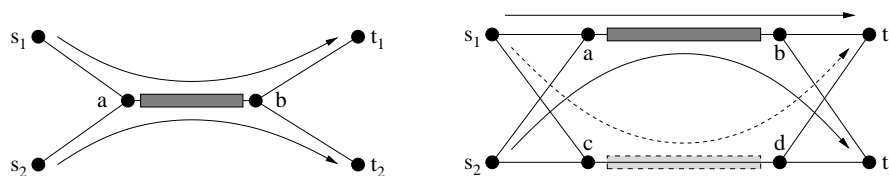


FIG. 3.2 – La collision de flux. À gauche, les flux n'ont pas de chemin alternatif. À droite, les flux oscillent entre deux chemins possibles.

Le critère de changement de chemin introduit pour éviter le phénomène d'oscillations ne résout pas le problème, puisque les chemins alternatifs trouvés par cette méthode peuvent toujours avoir des arcs en commun. On peut alors se retrouver assez paradoxalement avec un comportement oscillatoire (FIG. 3.2 à droite), alors que c'est justement ce qu'on cherchait à éviter.

3.2.4 La solution du backoff

Ce second type d'oscillations est dû principalement à la symétrie du système : rien ne distingue a priori une source ou un nœud intermédiaire d'un certain flux d'un autre flux concurrent et tous se comportent de façon analogue et prennent les mêmes décisions. La solution consiste donc à introduire de la dissymétrie, selon un procédé déjà appliqué dans les réseaux : le tirage d'un temps d'attente aléatoire (**random backoff timer**).

Déviation des flux

Selon le critère de changement de chemin, ce sont les nœuds situés en amont d'un lien en collision qui prennent la décision de dévier un flux. Il faut donc introduire la dissymétrie au niveau de ces nœuds-là pour éviter qu'ils dévient les flux d'une façon indésirable. Chaque nœud au départ d'une bifurcation pour un flux en collision tire un backoff, à l'expiration duquel il prend la décision de dévier le flux ou non, selon l'état du réseau à ce moment-là. De cette façon, c'est le flux pour lequel le backoff le plus long a été tiré qui conserve son chemin.

Nouveau critère de changement

Pour cette méthode, le critère de changement de chemin doit être augmenté pour pouvoir détecter la saturation de la capacité d'un lien. Supposons un lien sur lequel on mesure initialement un débit disponible de C et deux flux de débits respectifs δ_1 et δ_2 tels que $\delta_1 + \delta_2 > C$. Alors pendant tout le temps où les flux sont en collision sur le lien, la mesure du débit disponible sur ce dernier est de 0. Si le critère de changement de chemin est appliqué comme en 3.2.2, alors la collision n'est pas détectée, puisque pour chacun des flux, le lien a toujours un débit disponible suffisant (exactement égal au débit de chacun des flux). Il faut donc un moyen de mesurer des débits disponibles "négatifs", reflétant un dépassement du débit initialement disponible qui, pour le cas des deux flux de débits δ_1 et δ_2 correspond à $C - \delta_1 - \delta_2$. Alors, en ajoutant le débit d'un flux à ce débit mesuré, on

La taille de la fenêtre de backoff

Le temps d'attente aléatoire est tiré dans un intervalle appelé **fenêtre de backoff** dont la taille peut être manipulée pour induire des priorités entre les flux.

Prenons deux fenêtres de backoff de tailles S_1 et S_2 . Soient X_1 et X_2 les variables aléatoires correspondant aux valeurs tirées respectivement sur les intervalles $[0, S_1]$ et $[0, S_2]$. Alors la probabilité pour que X_1 soit supérieure à X_2 est de :

$$\int_0^{S_1} P(X_1 = \sigma) \cdot P(X_2 < \sigma) d\sigma = \begin{cases} \frac{S_1}{2S_2} & \text{si } S_1 \leq S_2 \\ 1 - \frac{S_2}{2S_1} & \text{si } S_1 > S_2 \end{cases}$$

Avec cette relation, il est simple de définir les règles d'augmentation de taille de la fenêtre dans le cas où la déviation d'un flux en collision provoque une autre collision. Par ailleurs, il est possible de définir des priorités a priori entre les flux en basant la taille de la fenêtre sur les requis de QoS du flux. On peut donner plus de priorité aux flux qui ont besoin d'un grand débit par rapport aux flux qui ont besoin d'un débit inférieur. Ainsi, les "gros" flux seront plus stables que les "petits", ce qui peut sensiblement décroître la quantité de données perdues en cours de collision.

3.2.5 Publication de la méthode

Cette méthode a été décrite dans un article [10] co-signé par Hakim Badis, Khaldoun Al Agha et moi-même, soumis et accepté à la conférence internationale IEEE VTC Fall 2004, qui aura lieu à Los Angeles du 26 au 29 septembre 2004.

Chapitre 4

Routage multichemin

Dans une bonne partie de protocoles de routage pour réseaux ad hoc, il existe des mécanismes de **réparation** de route en cas de changement de topologie. Dans le cas des protocoles de type best effort (par opposition à ceux offrant des fonctionnalités de QoS), les mécanismes de réparation ne sont requis que pour les protocoles réactifs, vu que les protocoles proactifs sont sensés réagir implicitement à chaque changement topologique.

Nous avons vu dans le chapitre précédent que des mécanismes analogues sont nécessaires toutefois quand des fonctionnalités de QoS sont envisagées dans un protocole proactif. Il s'agit en fait de considérer non plus une seule route d'une source à une destination, mais bien plusieurs routes.

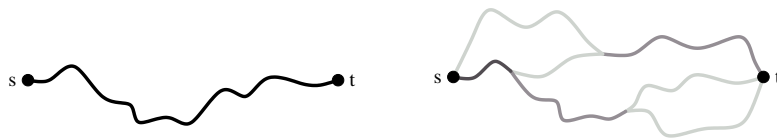


FIG. 4.1 – Monochemin et multichemin

En poussant le raisonnement encore plus loin, on se dit qu'au lieu de considérer plusieurs routes et d'en utiliser une seule à un moment donné, on peut considérer plusieurs routes à la fois et les utiliser simultanément.

En soi, l'idée n'est pas nouvelle, puisque cette approche est déjà employée dans les réseaux filaires, principalement pour arriver à faire de l'**équilibre de charge**. En effet, si on utilise deux chemins ou plus disjoints entre une source et une destination, on peut en théorie atteindre un débit cumulé égal à la somme des débits possibles sur les chemins séparément.

4.1 Les avantages des multichemins

Il se trouve encore une fois que la nature dynamique des réseaux ad hoc fait que l'utilisation de multichemins est plus versatile que la simple augmentation du débit possible.

4.1.1 Augmentation de la capacité

Dans le cas où les nœuds disposent chacun de deux interfaces, il est naturellement possible de doubler la capacité du médium radio en utilisant les interfaces sur deux canaux séparés. Alors on peut envisager un équilibrage de charge comme pour les solutions filaires.

Néanmoins, si les nœuds ne disposent pas tous de deux interfaces, l'utilisation de multichemins permet une utilisation plus équilibrée des ressources du réseau. En effet, en répartissant le flux sur plusieurs chemins, on répartit par la même occasion l'utilisation des ressources des nœuds intermédiaires et le débit utilisé sur les liens. Sachant que les nœuds d'un réseau ad hoc sont souvent limités en capacité de traitement, cette répartition est souhaitable. Par ailleurs, comme nous allons le voir plus loin, les capacités en débit disponible des liens ne sont pas indépendantes, à cause de la nature du médium radio.

4.1.2 Augmentation de la fiabilité

À chaque changement d'état du réseau, que ce soit un changement de mesure de QoS ou de topologie, des paquets de données sont susceptibles d'être perdus. L'utilisation de multichemins suggère deux solutions au problème.

D'abord implicitement, en répartissant les paquets successifs sur plusieurs chemins. En cas de rupture d'un des chemins, seuls les paquets émis le long du chemin défectueux peuvent être perdus, le temps que les multichemins soient recalculés. Ensuite explicitement, en émettant une copie du même paquet sur chacun des chemins possibles. Cette duplication n'est effectuée qu'aux bifurcations, pour ne pas gaspiller inutilement les ressources et simplifier le routage. Cette deuxième technique est toutefois inadaptée pour les protocoles qui ne gèrent pas les numéros de séquence au niveau de la couche transport (le protocole UDP par exemple), sans lesquels il est impossible d'éliminer les doublons à l'arrivée.

4.2 La capacité des liens dans les réseaux sans fil

À moins d'utiliser les antennes directionnelles parfaites ou bien des liaisons par laser, les capacités des liens dans un réseau sans fil ne sont pas indépendantes les unes des autres. Une bonne partie de la littérature sur le routage avec QoS dans les réseaux ad hoc fait abstraction de ce détail et considère le réseau simplement comme un graphe dynamique sur lequel des méthodes de recherche avec diverses métriques sont appliquées.

Dans le cas spécifique du standard IEEE 802.11, les mécanismes d'**évitement de collisions** (collision avoidance) impliquent l'utilisation de contrôle de flux et envoi d'accusés de réception pour les paquets **unicast**. Ces mesures préventives augmentent la corrélation des débits entre les liens jusqu'à deux sauts.

4.2.1 Illustration de la corrélation des débits

Prenons un exemple simple pour illustrer la corrélation des débits dans les réseaux radio. Soient cinq nœuds a , b , c , d et e sur une "branche", tous équipés d'une seule interface réseau.

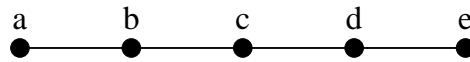


FIG. 4.2 – Exemple de topologie

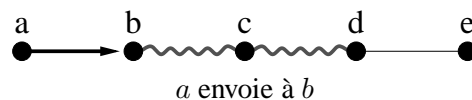
Bien que la topologie ne l'indique pas, lorsque par exemple b émet un paquet sur le médium radio, le paquet est envoyé dans toutes les directions à la fois et donc sur les arêtes $\{a, b\}$ et $\{b, c\}$.

Supposons maintenant que le paquet est émis par b à destination de c . On sait déjà que $\{a, b\}$ est occupée, de plus le mécanisme de **virtual carrier sense** empêche l'utilisation au même moment des arêtes $\{c, d\}$ et $\{d, e\}$. Voici les arcs bloqués et la raison de ce blocage :

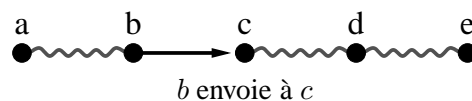
(a, b)	b est en train d'émettre et ne peut pas recevoir.
(b, a)	b ne peut émettre qu'un seul paquet à la fois.
(b, c)	l'arc est utilisé.
(c, b)	b émet et c reçoit.
(c, d)	c est en train de recevoir.
(d, c)	c ne peut recevoir qu'un seul paquet à la fois.
(d, e)	l'émission d'un paquet par d générerait la réception de c
(e, d)	d ne peut pas répondre à e dans la procédure de contrôle de flux ou bien d'accusé de réception.

Cet exemple illustre déjà la corrélation à deux sauts des débits disponibles sur les liens. Voyons maintenant quel débit maximal peut être atteint sur cette topologie. Nous allons voir ce qui peut se passer lorsqu'on envoie un flux de a vers e .

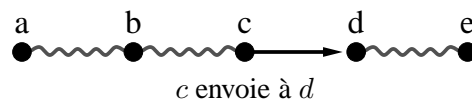
Dans une première étape, a envoie un paquet à b et bloque par là-même les arêtes $\{b, c\}$ et $\{c, d\}$.



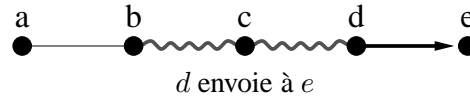
Ensuite b envoie le paquet à c et bloque les arêtes $\{a, b\}$, $\{c, d\}$ et $\{d, e\}$.



Puis c envoie le paquet à d et bloque les arêtes $\{a, b\}$, $\{b, c\}$ et $\{d, e\}$.



Enfin d envoie le paquet à e et ne bloque que les arêtes $\{b, c\}$ et $\{c, d\}$, permettant à a d'envoyer le paquet suivant à b .



On constate bien dans cet exemple que dans le cas d'une topologie rectiligne d'au moins trois sauts et en supposant que toutes les capacités à vide des liens sont de C , le débit maximal d'un flux unique est de $C/3$.

4.2.2 Contraintes linéaires de corrélation

La corrélation des capacités des arcs est liée à l'exclusion des utilisations des arcs dans le temps. Sur une échelle de temps macroscopique, ces exclusions peuvent être vues comme des contraintes linéaires sur les débits utilisés sur les arcs.

Soit \mathcal{C} la collection des ensembles d'arcs corrélés entre eux. Tous les arcs appartenant à un même ensemble de \mathcal{C} partagent une même ressource temporelle, c'est-à-dire que tous les arcs de l'ensemble sont bloqués si l'un d'entre eux est utilisé. Soit C un ensemble de la collection \mathcal{C} , (i, j) un arc de C de capacité à vide c_{ij} et de débit mesuré u_{ij} . Alors la relation suivante est toujours satisfaite :

$$\sum_{(i,j) \in C} \frac{u_{ij}}{c_{ij}} \leq 1 \quad \forall C \in \mathcal{C}$$

Calcul des ensembles d'arcs corrélés

Pour déterminer tous les ensembles de \mathcal{C} , nous avons considéré le graphe des corrélations entre les liens. Il s'agit d'un graphe non orienté, dans lequel les sommets sont les arcs du graphe topologique et une arête existe entre deux sommets si les arcs correspondants sont corrélés.

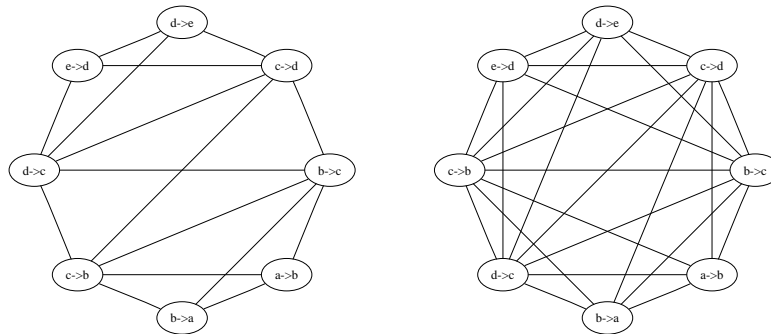


FIG. 4.3 – Les graphes de corrélation à un saut (à gauche) et deux sauts (à droite)

La figure 4.3 montre un exemple de graphes de corrélations pour la topologie donnée en section 4.2.1. On remarque que le graphe de corrélations à deux sauts est plus dense que celui de corrélations à un saut.

Les ensembles d'arcs corrélés de \mathcal{C} sont les ensembles de tous les arcs qui sont corrélés mutuellement. Cette définition correspond exactement aux cliques¹ du graphe des corrélations.

¹Sous-graphes complets maximaux

4.3 Les méthodes de recherche de multichemin

Le routage multichemin pour réseaux ad hoc a déjà été évoqué dans cadre de protocoles réactifs [11, 12, 13, 14]. Ces solutions envisagent cependant les multichemins de façon limitée, puisqu'il s'agit principalement de disposer d'un chemin de secours en cas où le chemin principal serait rompu. Par ailleurs, l'intérêt pour les protocoles proactifs n'a pas été énorme ces dernières années et on peut suspecter que cette négligence est la raison pour laquelle les tentatives d'effectuer du routage multichemin avec cette famille de protocoles sont restées rares voire existantes.

4.3.1 Notre approche

Dans [15], Lee et al. présentent une solution de calcul de multichemin pour réseaux filaires pour un ensemble de flux simultanés. Soit $G = (V, E)$ le graphe topologique et K l'ensemble des flux. Les variables X_{ij}^k représentent la part de la demande du flux k qui passe par l'arc (i, j) . Une fois que les X_{ij}^k sont fixés, il est possible de reconstituer les multichemins utilisés par les flux.

La méthode originale

La méthode fonctionne en deux phases. D'abord il s'agit de résoudre le programme linéaire suivant :

$$\begin{aligned} \text{Minimiser : } & \alpha & (4.1) \\ \text{s/c} & \end{aligned}$$

$$\left\{ \begin{array}{ll} \sum_{j:(i,j) \in E} X_{ij}^k - \sum_{j:(i,j) \in E} X_{ji}^k = 0, & k \in K, i \neq s_k, t_k \quad (4.2) \\ \sum_{j:(s_k,j) \in E} X_{s_k j}^k - \sum_{j:(s_k,j) \in E} X_{j s_k}^k = 1, & k \in K \quad (4.3) \\ \sum_{j:(t_k,j) \in E} X_{t_k j}^k - \sum_{j:(t_k,j) \in E} X_{j t_k}^k = -1, & k \in K \quad (4.4) \\ \sum_{k \in K} d_k \cdot X_{ij}^k \leq c_{ij} \cdot \alpha, & (i, j) \in E \quad (4.5) \\ 0 \leq X_{ij}^k \leq 1, 0 \leq \alpha & (4.6) \end{array} \right.$$

avec :

- K l'ensemble des flux
- E l'ensemble des arcs
- s_k la source du flux k
- t_k la destination du flux k
- d_k la demande en débit du flux k
- c_{ij} la capacité à vide de l'arc (i, j)
- X_{ij}^k la part de demande du flux k passant par l'arc (i, j)

Puis, une fois α fixé, résoudre le programme linéaire formé avec la fonction objectif :

$$\text{Minimiser : } \sum_{(i,j) \in E} \sum_{k \in K} X_{ij}^k \quad (4.7)$$

et les mêmes contraintes (4.2), (4.3), (4.4), (4.5) et (4.6) que précédemment.

La fonction objectif (4.1) est de minimiser le maximum d'utilisation d'un lien, que l'on retrouve dans la contrainte (4.5) qui spécifie que la part d'utilisation d'un lien est majorée par α . Les contraintes (4.2), (4.3) et (4.4) sont les contraintes de conservation de flot respectivement en tout nœud intermédiaire, à la source et à la destination.

La deuxième phase est nécessaire pour obtenir des multichemins acycliques. L'article décrit ensuite une autre méthode qui prend en compte la longueur des multichemins utilisés et permet de spécifier de combien de sauts les longueurs des multichemins peuvent dépasser la longueur d'un plus court chemin entre une source et une destination.

L'adaptation de la méthode

Nous nous sommes donc inspirés de cette méthode pour trouver un moyen de calculer un multichemin pour un seul flux, étant donné un réseau avec des flux existants. Pour cela, nous avons utilisé des variables X_{ij} et une demande d puisque nous n'avons plus qu'un seul flux et nous avons modifié la contrainte (4.5) pour incorporer les utilisations a priori des liens :

$$d \cdot X_{ij} + u_{ij} \leq c_{ij} \cdot \alpha, \quad (i, j) \in E \quad (4.8)$$

u_{ij} étant le débit mesuré sur l'arc (i, j) .

Nous avons bien sûr ajouté les contraintes linéaires sur les débits des arcs :

$$\sum_{(i,j) \in C} \frac{d}{c_{ij}} X_{ij} \leq 1 - \sum_{(i,j) \in C} \frac{u_{ij}}{c_{ij}}, \quad C \in \mathcal{C} \quad (4.9)$$

avec \mathcal{C} la collection des ensembles d'arcs corrélés.

Les défauts de la méthode

Telle quelle, la méthode ne donne pas de résultats satisfaisants, puisque rien ne force le programme de résolution à fournir des solutions dans lesquelles le flux est "étalé" au plus possible sur les chemins possibles. Si par exemple un multichemin est composé de chemins qui partagent tous un arc (i_0, j_0) , alors forcément la part de débit qui passe par cet arc est de 1, ce qui implique qu'on minore α par $\frac{d+u_{ij}}{c_{ij}}$.

Cette minoration fait que l'on ne contraint pas le résolveur à étaler le flux sur les autres chemins pour lesquels le taux d'utilisation du lien est inférieur. Pour la deuxième phase, il est complètement équivalent que le flux passe entièrement par un seul chemin ou par plusieurs, donc il a fallu trouver un moyen de forcer cet étalement.

4.3.2 Notre méthode finale

Après plusieurs tests avec les outils de simulation statique que nous avons développés, nous sommes arrivés à la conclusion que l'approche qui consiste à trouver les multichemins par programmation linéaire n'est pas efficace et ne donne pas de résultats satisfaisants. Même en adaptant un peu plus la méthode précédente pour forcer le résolveur à étaler encore plus le flux sur les chemins possibles, il nous arrivait de trouver des multichemins comportant des cycles, ce qui n'est clairement pas souhaitable.

Nous avons donc décidé d'effectuer un précalcul des chemins-candidats et d'utiliser la programmation linéaire uniquement pour décider de la part de trafic qui doit passer par chaque chemin, en respectant les contraintes de corrélation.

Précalcul des chemins-candidats

Le principe de calcul des chemins possibles est assez simple. Il s'agit de trouver des chemins qui satisfont tous le requis de délai. Comme la recherche de tous les chemins qui satisfont le délai est exponentielle, il faut adopter une méthode plus restreignante. Pour le moment, nous avons utilisé une méthode qui consiste à calculer une carte de distance (en délai) des sommets du graphe à la destination, c'est à dire une fonction qui à chaque sommet associe la longueur du plus court chemin vers la destination. Puis, en partant de la source, on sélectionne tous les arcs qui relient la source à des nœuds plus proches de la destination que la source elle-même. On prend ensuite chaque sommet ainsi relié par les arcs sélectionnés et on effectue la même opération. L'algorithme s'arrête quand tous les chemins ainsi formés par la juxtaposition des arcs sélectionnés ont atteint la destination.

Programme linéaire d'équilibrage

La méthode pour équilibrer le débit sur les chemins-candidats procède en deux phases :

$$\text{Minimiser : } \sum_{l=1}^L \alpha_l \quad (4.10)$$

s/c

$$\left\{ \begin{array}{l} \sum_{p=1}^P X_p = 1 \quad (4.11) \\ d \sum_{p=1}^P X_p A_{ij}^{lp} \leq \alpha_l c_{ij} - u_{ij}, \quad (i, j) \in E, \quad 1 \leq l \leq L \quad (4.12) \\ \sum_{(i,j) \in C} \frac{d}{c_{ij}} \sum_{p=1}^P X_p \sum_{l=1}^L A_{ij}^{lp} \leq 1 - \sum_{(i,j) \in C} \frac{u_{ij}}{c_{ij}}, \quad C \in \mathcal{C} \quad (4.13) \\ 0 \leq X_p \leq 1, \alpha_l \geq 0 \quad (4.14) \end{array} \right.$$

avec :

- E l'ensemble des arcs
- P le nombre de chemins possibles
- L la longueur maximale d'un chemin
- s et t la source et la destination
- d la demande de trafic à équilibrer
- c_{ij} la capacité à vide de l'arc (i, j)
- u_{ij} l'utilisation a priori de la capacité de l'arc (i, j)
- \mathcal{C} la collection des ensembles d'arcs corrélés
- A_{ij}^{lp} constante qui vaut 1 si le chemin p passe par l'arc (i, j) au saut l , 0 sinon
- X_p la fraction de trafic passant par le chemin p
- α_l le maximum d'utilisation des arcs au saut l

puis :

$$\text{Minimiser : } \sum_{l=1}^L \beta_l \quad (4.15)$$

s/c

$$\left\{ \begin{array}{l} \sum_{p=1}^P X_p = 1 \quad (4.16) \\ d \sum_{p=1}^P X_p A_{ij}^{lp} \leq \alpha_l c_{ij} - u_{ij} \quad (i, j) \in E, \quad 1 \leq l \leq L \quad (4.17) \\ \sum_{p=1}^P X_p A_{ij}^{lp} \leq \beta_l \quad (i, j) \in E, \quad 1 \leq l \leq L \quad (4.18) \\ \sum_{(i,j) \in \mathcal{C}} \frac{d}{c_{ij}} \sum_{p=1}^P X_p \sum_{l=1}^L A_{ij}^{lp} \leq 1 - \sum_{(i,j) \in \mathcal{C}} \frac{u_{ij}}{c_{ij}}, \quad \mathcal{C} \in \mathcal{C} \quad (4.19) \\ 0 \leq X_p \leq 1, \beta_l \geq 0 \quad (4.20) \end{array} \right.$$

Dans la première phase, nous utilisons des α_l qui majorent le taux d'utilisation par saut et pas globalement comme dans la méthode originale. L'idée de minimiser la somme des α_l dans la fonction objectif (4.10) donne de bons résultats (on voudrait en fait minimiser les α_l séparément). Puisque les chemins sont précalculés, les contraintes de conservation de flot sont inutiles. La contrainte (4.11) spécifie juste que toute la demande doit passer, (4.12) est la contrainte de majoration par α_l de des parts de débit utilisées au saut l , (4.13) est la contrainte de corrélation.

Dans la deuxième phase, nous avons fixé les α_l et introduit des β_l dont le rôle, similaire aux α_l , est de majorer la somme des parts de débit au saut l . L'idée est d'équilibrer encore plus le trafic entre les chemins, tout en restant dans les bornes imposées par les α_l .

4.3.3 Exemples de résultats

Voici quelques résultats obtenus lors de tests préliminaires avec les outils de simulation statique développés pendant le stage.

La figure 4.4 montre la topologie du réseau pris en exemple.

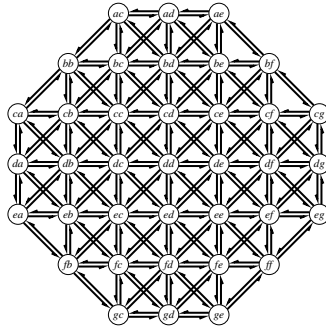


FIG. 4.4 – La topologie du réseau

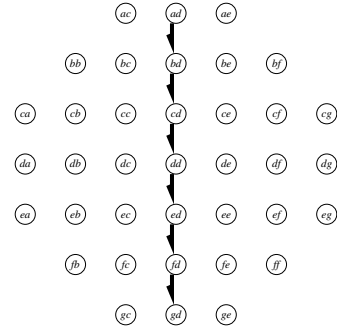


FIG. 4.5 – Un monochemin de débit 0.3

Les flux transversaux

Un avantage de l'utilisation des multichemins est la meilleure utilisation des ressources. Si on prend par exemple un réseau dans lequel tous les liens ont des capacités à vide de 1, alors on sait qu'on peut faire passer un flux de débit 1/3 sur un chemin unique d'une source à une destination distantes d'au moins trois sauts (FIG. 4.5). Mais à cause des corrélations entre les capacités des liens, une partie des nœuds par lesquels le chemin passe ne peuvent plus servir de nœud intermédiaire pour d'autres chemins simultanément. Si, au lieu d'utiliser un chemin unique, on utilise un multichemin, alors, pour peu qu'au moins deux chemins suffisamment distants les uns des autres soient utilisés, l'ensemble des nœuds "saturés" est bien plus petit. On peut ainsi réduire la probabilité de blocage du réseau.

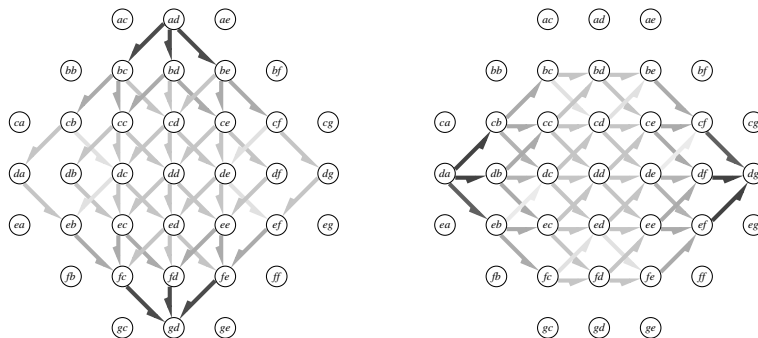


FIG. 4.6 – Deux multichemins simultanés de débit 0.3

Dans le cas du monochemin (FIG. 4.5), les flux partant "à gauche" du flux existant et à destination d'un nœud "à droite" ne peuvent passer. Dans le cas du multichemin (FIG. 4.6), bien que le flux est

de même débit, le fait d'être étalé permet à un deuxième flux transversal de débit 0.3 de passer simultanément. On voit bien ici la baisse de la probabilité de blocage.

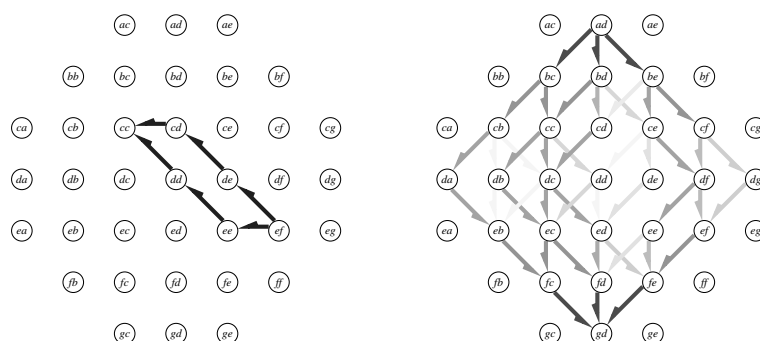


FIG. 4.7 – Deux autres multicheminés simultanés de débit 0.3

Dans la figure 4.7, tout comme dans la figure 4.6, le flux de gauche est initié d'abord et celui de droite ensuite. On voit mieux sur cette deuxième figure, comment le flux arrivant en deuxième "contourne" les arcs utilisés par le premier.

4.4 Les outils de simulation statique

Par opposition à la simulation dynamique, pour laquelle on considère des paramètres qui varient en fonction du temps pour simuler le fonctionnement d'un protocole de routage, nous avons d'abord considéré des simulations statiques. L'idée est d'être capable de faire une étude qualitative de la méthode de calcul des multicheminés, autrement dit de comparer l'utilisation de monochemins classiques et de multicheminés. Ce qui nous intéresse ici est surtout de montrer que les multicheminés induisent une probabilité de blocage moindre par rapport aux monochemins.

4.4.1 Principe de fonctionnement

L'approche adoptée pour les outils de simulation est celle de petits programmes séparés qui effectuent chacun une tâche précise et communiquent entre eux par l'intermédiaire de fichiers partageant un format commun. Ainsi tous les programmes n'ont pas été écrits dans le même langage. Là où des performances étaient requises, le C++ a été employé, alors que dans les autres cas, pour plus de souplesse dans le développement, Perl a été utilisé.

Le principal format de fichier était celui de description de graphe. Nous avons opté pour DOT qui a l'avantage d'être simple et suffisant pour nos besoins. Pour faciliter la manipulation de fichiers .dot, une bibliothèque statique en C++ et un module en Perl ont été développés pour factoriser le code d'interprétation et de génération de ce format de fichier.

Génération de graphes topologiques

Il s'agit de générer dans un premier temps des graphes topologiques aléatoires que l'on va considérer pour les simulations. Comme on veut simuler des réseaux ad hoc, le plus naturel est de définir

une portion du plan sur lequel on place aléatoirement les sommets du graphe, selon une loi uniforme. On définit ensuite une longueur de portée radio et on crée une arête entre tous les sommets distants d'au plus cette longueur.

Pour la génération des capacités à vide, une technique consiste à tirer aléatoirement selon une loi uniforme une valeur γ sur l'intervalle $[0, 1]$ pour chaque sommet. Cette valeur correspond à la disponibilité du nœud, où à la part de temps que le nœud peut consacrer à la transmission de données. Ainsi une capacité d'un arc (i, j) est donnée par $\min(\gamma_i, \gamma_j)$.

Pour la génération des délais sur les arcs, il suffit de tirer deux valeurs ρ^I et ρ^O aléatoirement selon une loi uniforme sur l'intervalle $[0, 1]$ qui correspond aux taux de remplissage respectifs des buffers d'entrée et de sortie de l'interface réseau du nœud. Ainsi un délai sur un arc (i, j) est donné par $\Delta \cdot \max(\rho_i^O, \rho_j^I)$, où Δ est le délai induit par un buffer plein.

Notre programme de génération de graphes topologiques prend en paramètres la taille de la surface, la distance minimale entre deux nœuds, la portée radio et le nombre de nœuds à placer. Le programme ne génère que des graphes connexes, c'est à dire que lors du tirage aléatoire de la position d'un point, celui-ci est conservé que s'il est à portée radio d'un point existant (sauf le premier point qui est placé inconditionnellement).

Génération de trafic a priori

La génération du trafic a priori est vraiment une affaire délicate, puisqu'il faut nécessairement que les débits respectent les contraintes de corrélations des capacités des arcs. Une technique simple mais coûteuse serait de générer du trafic à partir d'un réseau vide en calculant des multichemins pour un ensemble de flux. Pour simplifier et surtout gagner du temps, nous n'avons envisagé que la génération aléatoire de trafic, bien conscients que cette technique ne donne pas de résultats intéressants.

Calcul du graphe de corrélations

Le programme de calcul du graphe de corrélation lit en entrée un graphe sous forme de fichier `.dot` et prend en paramètre le nombre de sauts. Il renvoie en sortie un fichier `.dot` contenant le graphe de corrélations.

Calcul des cliques dans le graphe de corrélations

Pour le calcul de cliques, l'algorithme de Bron-Kerbosch [16] a été implémenté. Dans l'article originel, deux versions, une simple et une optimisée, sont décrites en langage Algol 60. Pour nous assurer de la correction de notre implémentation, nous avons d'abord implémenté ces deux versions. L'algorithme a la particularité d'être récursif, avec comme profondeur maximale l'ordre de la clique maximale (c'est-à-dire le nombre de sommets). Toutefois pour disposer d'une version dans laquelle cette profondeur maximale soit maîtrisable, nous avons procédé à la transformation du code des deux versions, pour obtenir un algorithme itératif qui utilise une pile explicitement. Dans la version actuelle du code, la gestion de la pile n'est pas optimisée, ce qui fait que la version itérative est moins performante que la version récursive.

Précalcul des chemins-candidats

Le précalcul des chemins-candidats est une implémentation de l'algorithme décrit en section 4.3.2. Le programme en C++ prend la topologie du réseau, la source, la destination et le délai maximal en entrée. Il sort la liste des chemins sous forme de suite de sommets.

Génération et résolution du programme linéaire

Pour la résolution de programmes linéaires, nous avons utilisé le logiciel GLPK, qui supporte le langage de description GNU MathProg. L'avantage de ce logiciel, c'est qu'il permet de décrire séparément le modèle et les données du programme linéaire. Ainsi notre programme de génération ne fait que sortir les données, le modèle restant inchangé et fourni dans un fichier séparé. L'autre avantage est que le langage MathProg est assez expressif, ce qui permet de décrire les modèles avec beaucoup de souplesse, d'une façon analogue à l'écriture mathématique.

Le programme de génération des données est un script Perl qui prend en entrée la topologie du réseau, le graphe de corrélations, la liste des chemins candidats et la demande en débit du flux. Il génère les données du programme linéaire et gère les appels au logiciel GLPK avec les modèles et données correspondants aux deux phases de l'équilibrage. Enfin il extrait les résultats de l'équilibrage et les rend en sortie.

Les programmes de visualisation

Toute une panoplie de petits programmes de visualisation on dû être écrits pour ces situations où aucun programme existant n'offrait les fonctionnalités requises.

Ainsi pour la visualisation des graphes générés, des délais sur les arcs, des capacités à vide, des débits a priori, des chemins-candidats et de la charge des multichemins, des scripts Perl qui prennent en entrée le fichier `.dot` de la topologie du réseau et d'autres données si nécessaire ont été développés. Ils génèrent des fichiers EPS qui peuvent être autant utilisés pour la visualisation sur écran que pour l'illustration dans les rapports et les transparents.

Chapitre 5

Conclusions

Pour arriver à intégrer la méthode de calcul de multichemin dans un prototype de protocole, des étapes essentielles doivent encore être franchies. Une méthode de calcul de routes seule ne suffit pas à faire un protocole de routage, même s'il s'agit d'étendre un protocole existant. Le meilleur exemple en est l'expérience avec le routage monochemin avec QoS qui a mené à la découverte des problèmes décrits en section 3.2.

Les chemins fournis par la méthode de calcul ne permettent pas pour le moment de faire du routage saut-par-saut, vu que des chemins différents peuvent avoir des arcs en commun et donc un paquet arrivant à un nœud intermédiaire devrait porter une sorte de marque pour indiquer quel chemin il a suivi jusque là pour être aiguillé correctement. Une solution simple pour réaliser ce type de routage, consiste à faire du source-routing, mais cette solution paraît peu naturelle pour le protocole OLSR.

5.1 Questions de performance

Les opérations de calcul de routes sont fréquentes pour un nœud participant à un réseau ad hoc et la puissance de traitement des stations est limitée. Ceci implique que les méthodes de calcul de routes doivent être efficaces pour être réalisables en temps réel.

La méthode de calcul de multichemin donne des résultats intéressants dans sa version actuelle, mais sa complexité est hors de portée d'une implémentation temps-réel.

5.1.1 Calcul des ensembles corrélés

Le calcul des cliques du graphe de corrélation est un problème NP-complet, ce qui fait qu'on ne dispose pas d'algorithme polynomial pour calculer les cliques d'un graphe quelconque. Tous les algorithmes connus n'ont cependant pas été inspectés et il se peut qu'on puisse gagner en complexité si on prend en compte la nature spécifique des graphes de topologie partielle utilisés dans OLSR (leur faible densité notamment).

Une autre idée qui nous semble intéressante à approfondir est l'utilisation d'approximations. Bien qu'elles ne donnent pas de solutions exactes, il se peut que des ensembles d'arcs corrélés approximatifs soient suffisant pour conserver une fiabilité des résultats suffisante.

5.1.2 Résolution du programme linéaire

La méthode du simplexe, utilisée pour la résolution des deux programmes linéaires, a une complexité polynomiale acceptable, surtout que ces programmes ont un faible nombre de variables (autant de variables que de chemins-candidats). En fait la résolution des programmes est rapide, mais la génération des matrices par GLPK à partir de la description du modèle prend un temps considérable. Cela est partiellement dû à la façon très compacte de décrire les contraintes de corrélation, mais aussi au fait que les ensembles sont nombreux.

Une optimisation simple que nous n'avons pas eu le temps de réaliser consiste, une fois les chemins-candidats déterminés, à réduire le graphe topologique afin d'éliminer les arcs qui sont trop éloignés (en nombre de saut) pour avoir une quelconque corrélation avec les arcs empruntés par les chemins. Ceci permettrait de réduire remarquablement le nombre d'ensembles corrélés et par là le nombre de contraintes de corrélation.

Enfin si l'heuristique suggérée en 5.1.1 fournit des cliques trop grandes, l'ensemble total de cliques sera certainement plus petit. Ceci réduira d'autant le nombre de contraintes de corrélation. Remarquons par ailleurs qu'il est préférable d'avoir des cliques trop grandes, donc des ensembles corrélés trop grands, puisque ceci implique des contraintes de corrélation trop fortes. Bien que cela puisse accroître la probabilité de blocage, on s'assure quand même que les capacités des liens ne sont pas dépassées.

5.2 Perspectives

5.2.1 Simulations dynamiques

Nous avons manqué de temps jusqu'à présent pour effectuer les simulations dynamiques que nous comptons faire pour approuver la méthode du backoff pour le routage avec débit disponible sans réservation de ressources. Ces simulations seront effectuées à l'aide du simulateur OPNET, sur lequel le groupe Réseaux du LRI est très compétent, puisque c'est en son sein que la couche protocolaire OLSR a été développée pour OPNET.

5.2.2 Simulations statiques

La méthode de calcul de multichemin avec QoS ne peut pas être simulée dynamiquement telle quelle et ne peut être que testée par simulation statique pour le moment. Une fois que les résultats seront confirmés, alors les optimisations prévues devront être étudiées et simulées à leur tour.

Quand tout le cadre du routage sera prêt (intégration dans OLSR, routage saut-par-saut, méthodes efficaces, etc), alors les simulations dynamiques seront intéressantes.

5.3 Remarques générales

Les propriétés des réseaux ad hoc entraînent des problèmes difficiles à résoudre, surtout lorsqu'une bonne partie de la littérature se concentre sur l'adaptation de solutions filaires. La difficulté est d'autant accrue qu'il faut trouver des méthodes efficaces, adaptées au routage dynamique.

Parfois certains problèmes ne ressortent que lorsqu'on s'attache à mettre en œuvre des techniques qui semblent donner de bons résultats dans les simulateurs, mais qui s'appuyaient sur des hypothèses trop fortes.

Il arrive que les thèmes que l'on veut aborder sont méconnus ou négligés et l'état de l'art inexistant. Ainsi les méthodes proactives de recherche de multichemin ont été peu étudiées, probablement en raison de l'a priori sur le manque de scalabilité de cette famille de protocoles.

Tous ces éléments font des réseaux ad hoc un domaine de recherche passionnant, ayant le double avantage d'être appliqué et présentant néanmoins des problèmes théoriques difficiles.

Mon travail au sein du groupe Réseaux du LRI a porté ses fruits et présente de nouvelles perspectives de recherche que j'espère pouvoir explorer dès le début de ma thèse.

Glossaire

backoff	Temps d'attente aléatoire choisi dans un intervalle prédéterminé.
blocage	Impossibilité de trouver un chemin satisfaisant les requis de QoS dans un état donné du réseau.
collision	Se dit de flux concourant pour la même ressource réseau.
équilibrage de charge	Envoi d'un flux sur un multichemin de façon à disposer de plus grande QoS.
évitement de collision	Procédé visant à éviter, sans pouvoir les détecter, les situations où deux stations se brouillent mutuellement en émettant simultanément.
inondation	Diffusion "naïve" de données dans un réseau en utilisant la retransmission des paquets par tous les nœuds.
lien	Possibilité de communiquer dans une seule direction (lien asymétrique) ou dans les deux (lien symétrique) entre deux stations.
métrique	Un paramètre réseau utilisé pour estimer la qualité d'une route.
multiplicative	Se dit d'une métrique si la métrique du chemin est le produit des métriques des liens qui le composent.
additive	Se dit d'une métrique si la métrique du chemin est la somme des métriques des liens qui le composent.
convexe	Se dit d'une métrique si la métrique du chemin est le maximum des métriques des liens qui le composent.
concave	Se dit d'une métrique si la métrique du chemin est le minimum des métriques des liens qui le composent.
multichemin	Ensemble de chemins utilisés simultanément ou alternativement pour acheminer les données d'une source à une destination.

multihop	Se dit d'un réseau où les nœuds ne sont pas tous connectés directement mais peuvent utiliser d'autres nœuds pour relayer les données.
nœud	Un dispositif matériel prenant part à un réseau.
qualité de service	Mesure de plusieurs paramètres réseau tels que le délai de transmission, le débit, le taux de perte et plus généralement, ensemble de procédés destinés à assurer leur niveau minimal requis.
réparation	Procédé de recherche de nouveau chemin lorsque le chemin couramment utilisé n'est plus valide.
réseau ad hoc	Réseau de mobiles munis d'interfaces réseau sans fil et pouvant communiquer les uns avec une partie des autres.
routage par la source (source routing)	Routage dans lequel la source inclut la liste des nœuds intermédiaires successifs à utiliser dans l'en-tête des paquets transmis.
saut-par-saut (hop-by-hop)	Routage avec tables où chaque nœud intermédiaire décide du successeur auquel envoyer les paquets.
station	(voir nœud)
virtual carrier sense	Mesures prévues par le standard IEEE 802.11 pour éviter les collisions.

Bibliographie

- [1] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF, July 2003.
- [2] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF). RFC 3684, IETF, February 2004.
- [3] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet-Draft Version 10, IETF, July 2004.
- [4] A. Munaretto, H. Badis, K. Al Agha and G. Pujolle. A Link-state QoS Routing Protocol for Ad Hoc Networks. *In the proceedings of IEEE MWCN2002, Stockholm, Sweden, September 2002.*
- [5] H. Badis, A. Munaretto, K. Al Agha and G. Pujolle. QoS for Ad hoc Networking Based on Multiple Metrics : Bandwidth and Delay. *In the proceedings of IEEE MWCN2003, Singapore, October 2003.*
- [6] Zheng Wang and Jon Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal of Selected Areas in Communications*, 14(7) :1228–1234, 1996.
- [7] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying : An efficient technique for flooding in mobile wireless networks. Technical Report Research Report RR-3898, INRIA, February 2000.
- [8] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol. RFC 3626, IETF, October 2003.
- [9] H. Badis and K. Al Agha. A Distributed Algorithm for Multiple-Metric Link State QoS Routing Problem. *In the proceedings of IEEE MWCN2003, Singapore, October 2003.*
- [10] H. Badis and K. Al Agha and I. Gawędzki. QoS Routing in Ad Hoc Networks Using QOLSR With No Need of Explicit Reservation. September 2004.
- [11] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM (3)*, pages 1405–1413, 1997.
- [12] A. Nasipuri and S. Das. Demand multipath routing for mobile ad hoc networks, 1999.
- [13] S. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks, 2001.
- [14] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks, 2001.

-
- [15] Youngseok Lee, Yongho Seok, Yanghee Choi, and Changhoon Kim. A constrained multipath traffic engineering scheme for MPLS networks. In *IEEE International Conference on Communications*, volume 4, pages 2431–2436. IEEE, April 2002.
- [16] Coen Bron and Joep Kerbosch. Algorithm 457 : finding all cliques of an undirected graph. *Commun. ACM*, 16(9) :575–577, 1973.