Routage Multichemins et Codage à Description Multiple dans les Réseaux Ad Hoc

Eddy Cizeron

UNIVERSITÉ DE NANTES

École Doctorale :

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHÉMATIQUES »

Année : 2009

Thèse de Doctorat de l'Université de Nantes

Spécialité : Automatique et Informatique Appliquée

Présentée et soutenue publiquement par

Eddy Cizeron

le 21 septembre 2009

à l'École Polytechnique de Nantes

Routage Multichemins et Codage à Description Multiple dans les Réseaux Ad Hoc

Jury

Rapporteurs :	Houda Labiod	Maître de Conférence à Telecom ParisTech
	Rodolphe Vauzelle	Professeur à l'Université de Poitier
Examinateurs :	Xavier Gandibleux	Professeur à l'Université de Nantes
	David Simplot-Ryl	Professeur à l'Université de Lille 1

Directeur de Thèse : Jean François Diouris

Professeur à l'École Polytechnique de Nantes, laboratoire IREENA

Co-encadrante : Salima Hamma

Maître de conférence à l'Université de Nantes, laboratoire IRCCyN

Table des matières

1 Les réseaux ad hoc **8**

I Les Réseaux mobiles . . . 9

I.1 Terminologie et modélisation des réseaux . . . 9

I.2 Caractéristiques générales des réseaux sans fil . . . 9

I.3 Avec ou sans infrastructure . . . 10

I.3.a Réseaux mobiles avec infrastructure . . . 10

I.3.b Réseaux mobiles sans infrastructure . . . 11

I.3.c Ondes radio et modulation . . . 11

I.3.d Détérioration du signal . . . 12

I.3.e Méthode d'accès . . . 13

CSMA . . . 14

CDMA . . . 14

Autre méthodes . . . 15

Dans le cas ad hoc . . . 15

I.4 Les technologies sans fil . . . 15

I.4.a Le Bluetooth et les normes IEEE 802.15 . . . 15

I.4.b Les normes IEEE 802.11 . . . 16

Le signal physique dans les normes 802.11 . . . 16

La liaison de données dans les normes 802.11 . . . 17

Les services proposés . . . 18

I.4.c Les réseaux cellulaires . . . 19

1ère génération . . . 19

2ème génération . . . 19

3ème génération . . . 19

I.4.d Le WIMAX et les normes IEEE 802.16 . . . 20

I.4.e Conclusion . . . 20

I.5 Conclusion . . . 20

II Le routage dans les réseaux ad hoc . . . 20

II.1 Caractéristiques du routage en contexte ad hoc 21
II.2 Algorithmes de routage dans le contexte filaire 21
II.2.a Protocole de routages à état de lien 22
II.2.b Protocole de routage à vecteur de distance 22
II.2.c Comptage à l'infini 23
II.3 Les classifications des protocoles de routage pour le ad hoc 24
II.4 Principaux protocoles proactifs 24
II.4.a Le protocole DSDV (Dynamic Destination-Sequenced Distance-Vector Routing Protocol) 24
Table de routage et messages de contrôle 24
Mise à jour de la table de routage 25
Remarques 26
II.4.b Le protocole OLSR (Optimized Link State Routing Protocol) 26
Gestion des voisins 26
Table de topologie et messages de contrôle 27
Mise à jour de la table de topologie 27
Mise à jour de la table de routage 28
Conclusion 29
II.4.c Le protocole TBRPF (Topology Broadcast Based on Reverse-Path Forwarding Protocol) 29
Table des voisins et messages HELLO 29
Mise à jour de la table des voisins 29
Nœuds rapportés et arbre rapporté 30
Table de topologie et messages de mise à jour 30
Conclusion 31
II.4.d Les protocoles GSR (Global State Routing Protocol) et FSR (Fisheye State Routing Protocol) 31
II.4.e Le protocole HSR (Hierarchical State Routing) 32
II.4.f Autres protocoles proactifs 33
II.5 Principaux protocoles réactifs 33
II.5.a Le protocole AODV (Ad hoc On-Demand Distance-Vector Routing Protocol) 33
Table de routage et HELLO 33
Requêtes 33
Réponses 34
Routage 34
Message d'erreur 35

Conclusion . . . 35
II.5.b Le protocole DSR (Dynamic Source Routing) . . . 35
Requêtes . . . 35
Réponses . . . 36
Routage . . . 36
Messages d'erreurs . . . 37
Conclusion . . . 37
II.5.c Autres protocoles réactifs . . . 37
II.6 ZRP, un protocole hybride . . . 38
II.7 Protocoles à routes multiples . . . 38
II.7.a Le protocoles SMR (Split Multi-path Routing) . . . 38
II.7.b Le protocoles AODV Multipath . . . 39
II.7.c Le protocoles AOMDV (Ad hoc On demand Multi-path Distance Vector) 39
II.8 Synthèse sur les algorithmes existants . . . 40
II.9 Services supplémentaires . . . 41
II.9.a Sécurité du routage . . . 41
ARAN . . . 41
ARIADNE . . . 42
SEAD . . . 42
SAODV . . . 43
SRP . . . 43
II.9.b Multicast . . . 43
II.9.c Qualité de service et réseau ad Hoc . . . 44
FQMM . . . 45
iMAQ . . . 45
INSIGNIA . . . 45
II.10 Simulation et comparaisons des protocoles . . . 45
III Conclusion . . . 46

2 Représentation multiple de l'information 48

I Méthode de codage source standard . . . 48
I.1 Format de la source . . . 49
I.2 Les étapes du codage . . . 49
I.2.a La quantification . . . 50
I.2.b Le codage entropique . . . 51
I.3 Optimisation . . . 51
II La description multiple . . . 52
II.1 Le principe et contexte d'utilisation . . . 53

II.2 Modélisation . . . 54
II.3 Quelques méthodes proposées dans la littérature . . . 54
II.3.a Protection égale . . . 55
II.3.b Protection inégale et encodage prioritaire . . . 56
II.4 Conclusion . . . 56
III Transformation Mojette . . . 56
III.1 Introduction . . . 56
III.2 Morphologie mathématique . . . 57
III.2.a Dilatation et érosion . . . 57
III.2.b Ouverture et fermeture . . . 57
III.2.c Élément structurant à deux pixels . . . 58
III.2.d Notion de connexité dans $\mathbb{Z}^n$ . . . 58
III.3 Définitions de la transformation Mojette . . . 59
III.3.a Transformée de Radon . . . 59
III.3.b Définition générale de la transformation Mojette . . . 60
III.3.c Transformation Mojette Dirac . . . 61
III.3.d Transformation Mojette discrète . . . 61
III.3.e Transformation Mojette Spline . . . 62
III.3.f Représentation matricielle . . . 62
III.4 Inversion de la transformation Mojette . . . 63
III.4.a Cas d'un ensemble de définition convexe . . . 63
III.4.b Cas d'un ensemble de définition rectangulaire . . . 63
III.4.c Algorithme de reconstruction . . . 64
III.4.d Reconstructibilité partielle . . . 64
III.5 Choix des projections et du support . . . 65
III.5.a Redondance . . . 65
III.5.b Cas du support rectangulaire . . . 66
III.5.c Cas du support hexagonal . . . 67
III.6 Mojette et codage MD . . . 68
III.6.a Concaténation de projections Mojette . . . 68
III.6.b Concaténation de support Mojette . . . 68
III.7 Conclusion sur la transformation Mojette . . . 68
IV Description multiple et routage . . . 69
V Conclusion . . . 69

3 Simulations de protocoles standard sur NS2 71

I Le logiciel NS-2 . . . 71
I.1 De la nécessité de la simulation . . . 71

I.2 Les choix de NS2 . . . 72
I.3 Le fonctionnement de NS2 . . . 72
I.4 Le paramétrage . . . 73
I.4.a L'espace de simulation et la mobilité . . . 73
I.4.b Les transferts de données . . . 73
I.4.c Les paramètres physique . . . 74
I.5 L'analyse des résultats . . . 75
I.5.a Les fichiers traces . . . 75
I.5.b Les critères d'évaluation . . . 76
I.5.c Outils d'analyse des résultats . . . 77
II Principe des tests . . . 77
II.1 Les Modèles de mobilité . . . 78
II.1.a Modèle Random Waypoint . . . 78
II.1.b Modèle Random Direction . . . 78
II.1.c Modèle Proba Walk . . . 79
II.1.d Modèle Gauss-Markov . . . 79
II.1.e Modèle Random Walk . . . 80
II.1.f Modèle Manhattan . . . 80
II.1.g Conclusion sur les modèles . . . 80
II.2 Les protocoles et les paramètres des tests . . . 80
III Résultat des tests . . . 80
III.1 Impact de la charge . . . 82
III.2 Impact de la mobilité . . . 84
III.3 Impact de la densité du réseau . . . 84
IV Analyse . . . 86
V Conclusion . . . 87

4 Descriptions multiples sur chemins multiples 88

I Introduction . . . 88
II L'intérêt des transfert multichemins . . . 89
II.1 Problèmes et objectif . . . 89
II.2 Répartir l'information . . . 91
II.3 Une redondance contrôlée . . . 92
II.4 Conclusion . . . 93
III Sélectionner des routes multiples . . . 94
III.1 Récupérer de l'information du réseau . . . 94
III.1.a Méthodes des procotoles existants . . . 94
III.1.b Le choix par la source . . . 95

III.2 Les contraintes du routage multichemins et la réalité pratique . . . 96
III.3 Objectif théorique . . . 97
III.3.a Modélisation théorique du réseau . . . 98
III.3.b Modélisation stochastique du fonctionnement des routes . . . 99
III.3.c Modélisation stochastique du fonctionnement des liens . . . 100
III.3.d Maximisation de la fiabilité . . . 102
III.4 Algorithmes existants à objectif similaires . . . 102
III.5 Notre proposition . . . 105
III.5.a Spécification . . . 105
III.5.b Le rôle des fonctions incrémentales . . . 106
III.5.c Complexité algorithmique . . . 107
III.6 Conclusion . . . 108
IV Répartition sur les routes . . . 108
IV.1 Reconsidération du problème d'optimisation de la fiabilité . . . 108
IV.2 Calcul pratique de la fiabilité . . . 109
IV.2.a Interdépendance des routes . . . 109
IV.2.b Notion d'état . . . 110
IV.2.c Construction progressive des routes . . . 110
IV.2.d Mise à jour des probabilités . . . 111
IV.2.e Exemples de mises à jour . . . 112
IV.2.f Calcul final . . . 113
IV.3 Problème simplifié . . . 113
IV.4 L'espace des répartitions . . . 114
IV.4.a Sous espace de $\mathcal{T}_k$ . . . 114
IV.4.b Vecteurs de répartitions équivalents . . . 114
IV.5 Proposition d'heuristiques . . . 115
IV.5.a Dénominateurs . . . 116
IV.5.b Numérateurs . . . 116
IV.6 Conclusion . . . 118
V Simulations . . . 118
V.1 Spécification des tests . . . 118
V.2 Résultats . . . 119
V.2.a Cas non systématique . . . 119
V.2.b Cas systématique . . . 121
V.3 Analyse . . . 122
V.4 Conclusion . . . 123
VI conclusion . . . 124

5 Le protocole MPOLSR **127**

I Le choix d'OLSR . . . 127

II Spécifications de MPOLSR . . . 128

II.1 Rappel sur le fonctionnement d'OLSR . . . 128

II.2 Intégration de l'algorithme de sélection des routes . . . 129

II.3 Routage par la source . . . 130

II.4 Rupture de routes . . . 131

II.5 À propos du regroupement des paquets . . . 131

II.6 Module de descriptions multiples . . . 134

II.7 Le projet ANR SEREADMO . . . 134

II.8 Conclusion . . . 135

III À propos de la mise en œvre dans NS2 . . . 135

III.1 Files d'attente des paquets et descriptions . . . 136

III.2 Création des routes . . . 136

III.3 Choix de stratégies de descriptions multiples . . . 136

III.4 Feed-back . . . 137

III.5 Maintenance des routes ("Routes recovery") . . . 137

IV Tests NS . . . 137

IV.1 Tests concernant les chemins multiples . . . 138

IV.1.a Spécification des tests . . . 138

IV.1.b Résultats . . . 138

IV.1.c Analyse . . . 141

IV.2 Tests sur l'utilisation de descriptions multiples . . . 142

IV.2.a Spécification des tests . . . 142

IV.2.b Résultats . . . 142

IV.2.c Analyse . . . 143

IV.3 Conclusion des tests . . . 144

IV.4 Conclusion . . . 144

6 Le protocole TMR **146**

I Introduction . . . 146

II Spécifications de TMR . . . 146

II.1 Les tables . . . 147

II.1.a La table d'accès . . . 147

II.1.b La table de topologie . . . 147

II.1.c Information commune . . . 148

II.2 Requête et réponse . . . 148

II.2.a Procédure de requête . . . 148

II.2.b Procédure de réponse . . . 149
L'ellipse : zone de dispersion des réponses . . . 150
Traitement de la réponse par un nœud intermédiaire . . . 150
Parcours de jetons . . . 151
II.2.c Réactualisation de la topologie . . . 152
II.3 Création de routes . . . 153
II.4 Répartition de l'information . . . 153
Cas du RoundRobin . . . 153
Cas MDC non systématique . . . 153
Cas MDC systématique . . . 154
II.5 Utilisation des routes . . . 154
II.5.a Ruptures . . . 154
II.5.b Court-circuits . . . 154
II.6 Conclusion . . . 155
III Tests . . . 155
III.1 Paramètres . . . 155
III.2 Résultats . . . 155
III.2.a Impact de la méthode de codage . . . 155
III.2.b Impact du débit . . . 158
III.2.c Impact de la densité des nœuds . . . 159
III.3 Analyse . . . 160
IV Conclusion . . . 161

A Terminologie des réseaux sans fil **166**

B Format des traces de NS2 **168**

C Protocoles et modèles de mobilité **171**

Table des figures

1.1 Réseau mobile avec infrastructure 10
1.2 Réseau mobile ad hoc 11
1.3 Dispersion de chemins (a) et étalement du spectre par séquence directe (b) 13
1.4 Le protocole DCF 18
1.5 Algorithmes d'état de lien (a) et de vecteur de distance (b) 22
1.6 Comptage à l'infini 23
1.7 MPR du nœud `V` (en gris) 26
1.8 Transmission d'un message de contrôle de topologie 28
1.9 Arbre rapporté du nœud `V` 30
1.10 Circulation de RREP et RREQ dans AODV 35
1.11 Portage (piggybacking) de la réponse RREP1 par une nouvelle requête RREQ2 36

2.1 Décomposition d'un codeur / décodeur 50
2.2 S-connexité : l'ensemble G_1 est S-connexe, mais pas G_2 59
2.3 Droite du plan support de l'intégration pour la transformée de Radon 60
2.4 Projection suivant $(p, q) = (2, 1)$ d'une fonction définie sur un ensemble G discret et borné 61
2.5 Ensemble R pour les directions de projection $(-1, 1)$, $(0, 1)$ et $(2, 1)$ (a) et exemples d'ensemble G non-reconstructible (b) et reconstructible (c) 63
2.6 Exemple de reconstruction avec les projections de vecteur $(-1, 1)$, $(1, 1)$ et $(1, 0)$ 65
2.7 Fantôme associé aux directions de projection $(-1, 1)$, $(0, 1)$ et $(2, 1)$ 66
2.8 Exemple de support hexagonal et de projections associées 67
2.9 Modèle PET obtenu par concaténation de projections Mojette calculées pour différents calques d'une source scalable 68
2.10 Modèle PET obtenu par concaténation de différents calques d'une source scalable projetés suivant un ensemble d'angles donné 69

3.1 Taux de paquet délivré, en fonction de la charge 82
3.2 Délai, en fonction de la charge 82
3.3 Gigue, en fonction de la charge 83
3.4 Coût du routage, en fonction de la charge 83

3.5 Concentration de l'activité, en fonction de la charge 83
3.6 Taux de paquet délivré, en fonction de la mobilité 84
3.7 Délai, en fonction de la mobilité 84
3.8 Gigue, en fonction de la mobilité 85
3.9 Coût du routage, en fonction de la mobilité 85
3.10 Taux de paquet délivré, en fonction de la taille du réseau 86
3.11 Délai, en fonction de la taille du réseau 86
3.12 Gigue, en fonction de la taille du réseau 86
3.13 Concentration de l'activité, en fonction de la taille du réseau 87

4.1 Risque d'enconbrement dû à la concentration des flux sur certains nœuds 90
4.2 Les différentes stratégies d'utilisation des routes 91
4.3 Exemple de transformation de paquets en descriptions Mojette ($N = 3$,$M = 2$) 92
4.4 Exemple de transformation de paquets en descriptions par utilisation de Xor ($N = 3$,$M = 2$) 93
4.5 Convergence des routes 95
4.6 Noeud pendant 97
4.7 Allongement des routes sous la contrainte 97
4.8 Arbre source du nœud `S` 99
4.9 Répartition de $N = 10$ descriptions (6 pseudo-descriptions et 4 descriptions de redondance) sur $k = 4$ routes 101
4.10 Etape de dèsentrelacement opérée dans l'algorithme de Suurballe 105
4.11 Exemple du fonctionement de l'algorithme 4 105
4.12 Partie du graphe à mettre à jour entre deux étapes 108
4.13 Découpage de l'ensemble des routes en sous-ensembles indépendants 110
4.14 Pour $\mathcal{K} = (R_1, R_2, R_3, R_4)$ et un lien e, exemple de différence entre états dépendants et indépendants de e 112
4.15 Fiabilités de toutes les répartitions sur l'espace $\mathcal{T}_3$ avec $\vec{p} = (0.87, 0.64, 0.55)$ et un rapport constant $\rho = M/N = 3/5$ 114
4.16 Fiabilité pour trois vecteurs $\vec{p}$ différents avec $\rho = 0.4$ 115
4.17 Fiabilité les mêmes $\vec{p}$ avec cette fois $\rho = 0.2$ 115
4.18 Sélection d'un vecteur dans chaque sous-ensemble 115
4.19 Sélection d'un vecteur dans chaque sous-ensemble avec valeurs ordonnées 116
4.20 Sélection d'un vecteur de fraction de dénominateur au plus égal à 4 116
4.21 Différentes valeurs possibles pour le dénominateur en fonction de ρ 117
4.22 Différentes valeurs possibles pour les numérateurs en fonction de ρ et pour $k = 6$ 117
4.23 Variations globales de la fiabilité pour $\lambda = 15Mb/s$, cas non systématique 121
4.24 Variations globales de la fiabilité pour $\lambda = 6Mb/s$, cas non systématique 122
4.25 Variations globales de la fiabilité pour $\lambda = 22.5Mb/s$, cas non systématique 123

4.26 Variations globales de la fiabilité pour $\lambda = 15Mb/s$, cas systématique 124
4.27 Variations globales de la fiabilité pour $\lambda = 6Mb/s$, cas systématique 125
4.28 Variations globales de la fiabilité pour $\lambda = 22.5Mb/s$, cas systématique 126

5.1 Taux de paquets délivrés 133
5.2 Délai 134
5.3 Taux de paquet délivré 140
5.4 Coût du routage 140
5.5 Délai moyen 141
5.6 Concentration de l'activité 142
5.7 Taux de paquet délivré 143
5.8 Coût du routage 144

6.1 Algorithmes de traitement des paquets de contrôle 149
6.2 Tableau des jetons 150
6.3 Ellipse 151
6.4 Processus de collecte des jetons 152
6.5 Taux de paquets délivrés, 100 nœuds, 10 paquets/s, portée de 175 m 157
6.6 Délai, 100 nœuds, 10 paquets/s, portée de 175 m, codage systematique 158
6.7 Coût du routage, 100 nœuds, 10 paquets/s, portée de 175 m, codage systematique 158
6.8 Taux de paquets délivrés, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique 159
6.9 Coût du routage, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique 159
6.10 Délai, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique 160
6.11 Taux de paquets délivrés, 25 paquets/s, portée de 250 m, codage systematique 161
6.12 Délai, 100 nœuds, 25 paquets/s, portée de 250 m, codage systematique 162
6.13 Coût du routage, 75 nœuds, 25 paquets/s, portée de 250 m, codage systematique 162

A.1 Arbre source de `V` 167

C.1 Taux de paquets délivrés en fonction de la charge 173
C.2 Délai en fonction de la charge 174
C.3 Coût du routage en fonction de la charge 175
C.4 Gigue en fonction de la charge 176
C.5 Concentration de l'activité en fonction de la charge 177

Introduction

"Le monde bouge" et, si l'on en croit la rengaine journalistique, il bouge de plus en plus vite. La mobilité des individus, des flux, des marchandises, de l'information est devenu un objectif de nos sociétés. On peut trouver dans cette agitation une première justification : la recherche de l'indépendance. Ne plus être lié au lieu c'est souvent savoir mieux s'adapter et donc gagner en indépendance. Or, cette dernière n'est a priori pas gratuite. Un des principaux inconvénients de la mobilité est en effet qu'elle complique la communication. Il a longtemps été difficile de rester joignable pour celui qui ne pouvait rester en un même endroit. Car transmettre de l'information, cela commence tout d'abord par déterminer à qui (et donc où) la transmettre.

L'arrivée de la téléphonie mobile a constitué une transformation fondamentale dans ce rapport à la communication. Puisque l'outil de communication devenait transportable, son utilisateur pouvait de n'importe où joindre (presque) n'importe qui. L'utilisateur y gagna l'indépendance physique. Encore que, pour que ce service lui soit accessible, il devait - et doit encore de nos jours - payer les services d'un opérateur téléphonique. "La véritable indépendance consiste à dépendre de qui on veut" disait Frédéric Dard. Si un téléphone ou un ordinateur portable permet de s'affranchir d'une contraignante immobilité géographique, son fonctionnement implique en revanche de façon quasi-systématique l'existence d'intermédiaires, de structures spécialisées, dont l'objectif va être de faire transiter des données entre différents acteurs mobiles.

Le monde des technologies sans fil n'est donc pas celui de l'indépendance pure. Les réseaux pair-à-pair, bien que non nécessairement sans fil, ont toutefois montré qu'il pouvait exister d'autres types d'indépendance que la mobilité. Une structure privée de hiérarchie (mais pas pour autant d'organisation) dispose de l'énorme avantage qu'aucun de ses composants n'est irremplaçable. Si différents rôles peuvent être mis en place afin d'en assurer le bon fonctionnement, le fait que tout acteur peut assurer n'importe lequel de ces rôles garantit que les intermédaires spécialisés ne sont plus nécessaires. Autrement dit, la structure existe grâce au bon vouloir de chaque utilisateur. Les réseaux ad hoc (décrits dans le chapitre 1) sont nés de la volonté d'appliquer ce type d'idée au monde sans fil : il s'agit de réunir un certain nombre d'unités mobiles pouvant tour à tour jouer le rôle d'émetteur, de récepteur et d'intermédiaire dans la transmission d'information. Chaque utilisateur accepte de participer au bon fonctionnement d'échanges

qui ne le concerne pas directement parce qu'il sait pouvoir à son tour bénéficier de l'aide des autres participants pour envoyer ou recevoir des données. Les réseaux ad hoc constituent donc l'aboutissement de la volonté d'indépendance vis-à-vis d'une infrastructure spécialisée pré-existante.

Cependant, cette nouvelle forme d'indépendance a également un coût. On peut ainsi penser aux complications que cela entraîne du point de vue de la confidentialité des données ou la gestion des arrivées et départs des participants ; mais plus que tout, la problématique majeure des réseaux ad hoc est sans nul doute le routage. Déterminer le trajet le plus adapté pour faire transiter les données, maintenir ce trajet et l'adapter sont des mécanismes assez complexes dans un univers où il n'y a plus aucune garantie de pérennité. Le principe d'adresses IP, tellement commun dans les réseaux filaires, ne fonctionne pas dans un environnement où l'adresse ne peut plus représenter, en plus d'un identifiant, une localisation dans le réseau. Malgré ces difficultés, un nombre important de propositions a été fait concernant le routage en contexte ad hoc. Parmi le vaste éventail de protocoles proposés, DSR, AODV et OLSR se sont démarqués de leur concurrents. Quel que soit le protocole utilisé, les réseaux ad hoc demeurent néanmoins sensibles à l'augmentation du nombre d'unité mobile et du trafic ; ce qui aboutit bien souvent à un engorgement de certaines unités, ou à la perte de routes en cours de communication.

La problématique de cette thèse de doctorat se situe dans l'étude d'une stratégie multi-route en contexte ad hoc conjointement avec l'utilisation de codage par descriptions multiples. En effet, une idée simple pour lutter contre l'engorgement du réseau consiste à ne plus se restreindre à une seule route, comme c'est traditionnellement le cas en filaire. Par ailleurs, l'introduction de redondance dans l'information routée semble intuitivement pouvoir diminuer la sensibilité des données à la disparition d'une route. Le codage à description multiple (MDC, voir le chapitre 2) consiste à transformer l'information originale en différentes unités indépendamment manipulables et redondantes appelées descriptions. Il s'agit donc d'un mécanisme intéressant dans le contexte de notre étude.

Nous cherchons donc ici à non seulement distribuer l'information sur plusieurs chemins, mais également à savoir comment introduire de la redondance dans celle-ci. Le but à atteindre est évidemment d'obtenir de meilleures performances grâce à l'introduction conjointe de ces deux mécanismes. Evaluer ces performances nécessite cependant de définir des critères particuliers permettant d'établir des comparaisons. Le chapitre 3 détaille ces critères et les applique à l'analyse des protocoles standards du routage ad hoc dans différents contextes. Les analyses sont réalisées au moyen du simulateur NS2.

L'objet du chapitre 4 est d'étudier dans le détail la façon de combiner ad hoc, routes multiples et codage à description multiple, ainsi que de montrer comment cette approche peut théoriquement améliorer la réception des données. Il s'agit du cœur de notre proposition : déterminer un mécanisme de sélection des routes et une stratégie de répartition des descriptions sur ces routes qui soient capables de rendre le

routage moins sensible au débit et à l'instabilité naturelle des réseaux ad hoc.

Nous proposons dans le chapitre 5 une variante du protocole OLSR, appelée MPOLSR (Multi-Path OLSR), dans laquelle nous avons introduit les idées développées au précédent chapitre (les protocoles du type de OLSR, dit proactifs, se prêtant en effet bien à l'application de telles idées). L'élaboration de ce protocole a en outre constitué une des étapes du projet ANR SEREADMO, projet visant la sécurisation du routage ad hoc. Après la description des spécifications de MPOLSR, une évaluation par simulation de celui-ci est réalisée montrant notamment les variations de performances par rapport à OLSR.

Enfin, le chapitre 6 se consacre à examiner si ces mêmes idées fonctionnent également dans un contexte réactif ; c'est-à-dire dans lequel la recherche de route n'est effectuée qu'en cas de transfert. À cette fin, le protocole TMR (Topology Multipath Routing) reprend le fonctionnement de MPOLSR en ce qui concerne le codage de l'information et la répartition de celle-ci sur plusieurs routes. En revanche, la recherche de routes, inspirée par des protocoles comme DSR, est basée sur un mécanisme de requêtes et réponses ayant pour but de collecter à la demande des informations sur la topologie du réseau. TMR constitue donc la seconde de nos propositions de routage multiroutes utilisé conjointement avec une méthode MDC.

Notations

Généralités	
Ω	Espace probabilisé
$\mathbb{P}(\cdots)$	Mesure de probabilité
$\mathbb{E}(\cdots)$	Espérance
$\mathbf{1}_E(\cdots)$	Fonction indicatrice de l'ensemble E
$\mathbf{H}(\cdots)$	Echelon de Heaviside
Codage	
$\mathfrak{C}, \mathfrak{D}$	Codeur et décodeur
α, β	Codeur par quantification et décodeur de reproduction
γ, γ^{-1}	Codeur entropique et décodeur associé
$\mathcal{A}^{\mathcal{N}}$	Ensemble des vecteurs de dimension $\mathcal{N}$ sur l'ensemble $\mathcal{A}$
$d(\cdots, \cdots)$	Distance sur $\mathcal{A}^{\mathcal{N}}$
$\vec{X}, \vec{X}^{(1)}, \vec{X}^{(2)}, \cdots$	Variables aléatoires sur $\mathcal{A}^{\mathcal{N}}$
	modélisant un signal source
$\hat{X}, \hat{X}^{(1)}, \hat{X}^{(2)}, \cdots$	Estimations
$\vec{x}, \vec{x}^{(1)}, \vec{x}^{(2)}, \cdots$	Réalisations correspondantes
$\hat{x}, \hat{x}^{(1)}, \hat{x}^{(2)}, \cdots$	Estimations des réalisations
$\vec{Y}$	$\vec{X}$ décorrélé
$\mathcal{I}$	Ensemble de symboles
R	Débit
Δ	Distortion
Graphes et nœuds	
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \text{cout})$	graphe valué
`S`,`D`	Source et destination
`V`,`U`,`W`	Nœuds
`e`	Lien

$\mathtt{o}$	Lien ou nœud
	Routes
$\mathcal{K}$	k-uplet de routes
$\mathcal{K}_1, \cdots, \mathcal{K}_{k'}$	Sous-ensembles de routes indépendantes
$R_1, \cdots, R_k$	Routes
k	Nombre de routes
	Paquets et descriptions
P	Paquet
D	Description
$\vec{P}$	N'-uplet de paquets de données
$\vec{D}$	N-uplet de descriptions générées ensemble
N'	Nombre de paquets regroupés
N	Nombre de descriptions générées
M	Nombre de descriptions suffisantes pour la reconstruction
N_i	Nombre de descriptions sur la i-ème route ($N = \sum_i N_i$)
M_i	Nombre de pseudo-descriptions sur la i-ème route ($M = \sum_i M_i$)
$\vec{N}$	$(N_1, \cdots, N_k)$
$\vec{M}$	$(M_1, \cdots, M_k)$
f_e, f_p	Fonctions d'incrémentation du coût des liens
	Fiabilité
$p_{\mathtt{e}}$	Fiabilité d'un lien
$X_{\mathtt{e}}$	v.a. de validité de $\mathtt{e}$
$p_{\mathtt{V}}$	Fiabilité d'un noeud
$X_{\mathtt{V}}$	v.a. de validité de $\mathtt{V}$
p_i	Fiabilité de la route R_i
Y_i	v.a. de validité de Y_i
$\vec{Y}$	$(Y_1, \cdots, Y_k)$
$\mathcal{R}$	Fiabilité de $\mathcal{K}$
Z	v.a. correspondant au nombre de descriptions reçues
Z'	v.a. correspondant au nombre de paquets reconstruits
s_i	Elément de $\{0, 1\}$ correspondant à un état envisagé de la route R_i
$\vec{s}$	$(s_1, \cdots, s_k)$, état envisagé pour les k routes de $\mathcal{K}$
	Flux de paquets
RTR	Couche de routage

AGT	Couche agent (au dessus du routage)
$P_{donnees}$	Paquet de données
$P_{routage}$	Paquet de contrôle
$\mathcal{P}$	Ensemble de paquets
$\mathcal{P}_{donnees}$	Ensemble des paquets de données
$\mathcal{P}_{routage}$	Ensemble des paquets de contrôle
$\mathcal{P}_y^{x\rightarrow}$	Ensemble des paquets de type y
	envoyés par la couche x (RTR ou AGT)
$\mathcal{P}_y^{-x\rightarrow}$	Ensemble des paquets de type y
	retransmis par la couche x (RTR ou AGT)
$\mathcal{P}_y^{\rightarrow x}$	Ensemble des paquets de type y
	reçus par la couche x (RTR ou AGT)
δt_P^{AGT}	Temps de parcours du paquet P entre les
	couches AGT de la source et de la destination
$P \prec P'$	P suit P' au sein d'un même flux de données
τ_P	Délai entre deux paquets de données consécutifs
τ_D	Délai entre deux descriptions consécutives
λ	Débit d'entrée des données
$\lambda' = \lambda_{glob}$	$= N\lambda_{loc} = N\lambda/M$, débit d'entrée aprés MDC
λ_{loc}	$= \lambda/M$, débit local sur le réseau (par route)

Chapitre 1

Les réseaux ad hoc

Introduction

À l'heure où la diffusion de l'information ne cesse de s'imposer comme un des plus gros besoins de notre société, les réseaux informatiques constituent depuis quelques années un outil incontournable pour le transport de celle-ci, concurrençant notamment les supports analogiques utilisés par la télévision ou la téléphonie. Si le problème est toujours le même - à savoir comment reproduire en un point donné l'information existant en un autre point - de nouvelles attentes sont apparues. Vitesse et quantité accrues, besoin de sécurité plus important, information en continu, en temps réel, l'outil de communication doit satisfaire de nouveaux besoins par nature difficilement conciliables, si bien que les schémas établis dans le passé pour les réseaux informatiques eux-même doivent constamment être repensés.

Après l'avènement d'Internet via les réseaux filaires, nous entrons aujourd'hui dans une nouvelle ère technologique où la demande de mobilité impose aux méthodes de transmission de données récentes de pouvoir s'abstraire le plus possible de l'environnement physique. Le but visé est d'assurer un échange indépendamment de la localisation géographique des participants, voire au cours de leurs déplacements. Ces réseaux - dit mobiles - sont suivant les cas plus ou moins liés à une architecture fixe pré-existante. Dans le cadre d'une indépendance maximale on parle de réseaux ad hoc. Destinés à l'origine à des fins militaires, les réseaux ad hoc ont été créés et améliorés lors de projets PRNet, SURAN ou encore GloMo [Mer05]. Particulièrement adaptés pour une mise en place rapide et peu coûteuse, ces derniers possèdent une structure caractérisée par l'absence de matériel fixe. Ces caractéristiques a priori avantageuses conduisent cependant à l'apparition de nouveaux problèmes conceptuels pour lesquels les solutions apportées par le filaire ou les réseaux sans fil standard ne sont pas pleinement réutilisables.

Un des principaux défis de ce type de structure consiste à établir de nouvelles méthodes de routage plus adaptées. Autrement dit déterminer un chemin possible - et si possible le meilleur - que peuvent parcourir les données avant de parvenir à destination ; ceci tout en tenant compte de contraintes diverses telles que les performances en terme de temps de réception, le débit maximal utilisable, les critères de sécurité habituels dont notamment l'intégrité et la disponibilité, etc ...

Nous allons ici exposer une introduction aux réseaux sans fil en général, aux principales caractéristiques et technologies associées, au mode ad hoc plus particulièrement avant de détailler différents algorithmes de routage proposés pour ce dernier.

I Les Réseaux mobiles

I.1 Terminologie et modélisation des réseaux

Un *réseau* est un ensemble de composants informatiques ayant chacun une certaine autonomie et reliés les uns aux autres par un certain média de communication permettant l'échange d'informations (pour un réseau filaire des câbles électriques ou des fibres optiques, pour un réseau mobile des ondes électromagnétiques). Le monde des réseaux possède son propre vocabulaire les termes principaux ici utilisés sont détaillé en annexe A.

I.2 Caractéristiques générales des réseaux sans fil

Un réseau sans fil (voir [Müh02]) est caractérisé par son support particulier correspondant à des signaux électromagnétiques qui permettent une connexion entre les différents nœuds sans utiliser les canaux physiques habituels du réseau filaire, mais également à une plus grande sensibilité aux perturbations et à un débit global plus lent dû aux retransmissions nécessaires pour palier aux pertes de paquet plus fréquentes.
Du point de vue physique (couche 1 du modèle OSI), la transmission d'information par ondes radio s'effectue via l'élaboration d'un signal sinusoïdal p de fréquence particulière ν appelé porteuse. L'introduction d'information consiste à combiner la porteuse avec un signal à transmettre (de format ici numérique, c'est à dire une succession de 0 et de 1) avant l'émission sur le canal, ce que l'on appelle la modulation du signal.
Un protocole d'accès intervient lors de la transmission au niveau de la couche liaison de donnée (couche 2 du modèle OSI). Son but est de permettre le partage physique du média de communication entre plusieurs nœuds. Lorsque plusieurs paquets sont émis en même temps sur le média en utilisant une même bande de fréquence, les phénomènes d'interférence résultant conduisent à une superposition inutilisable des différents signaux ; on parle alors de collision. Les technologies sans fil impliquent l'existence de problèmes inconnus du filaire qui nécessitent ainsi de nouvelles solutions techniques. Lors d'une transmission par ondes, certains nœuds peuvent par exemple être hors de portée de l'émetteur. Ce fait implique qu'un événement lorsqu'il se produit n'est plus perçu de la même façon par tous les nœuds partageant le média. Cela peut d'un côté constituer un avantage dans le sens où une limitation naturelle des collisions est ainsi créée. D'un autre côté, ces phénomènes de masquage empêchent la détection d'envois de paquets par certains nœuds, ce qui augmente le risque de collision en certains points.
Concernant le routage (couche 3 du modèle OSI), la variabilité de la topologie au cours du temps, du fait d'événements tels que :

– l'apparition ou la disparition de nœuds (et donc de liens) ;
– le déplacement de nœuds dans l'espace ;
– l'apparition ou la disparition de liens dus à l'environnement physique ;
peut compliquer la recherche et le maintien de routes entre les nœuds par rapport au cas filaire. Un de ses effets directs est la difficulté à définir les limites du réseau. Par ailleurs il n'existe plus de correspondance entre l'adresse d'un nœud et sa localisation dans le réseau.

I.3 Avec ou sans infrastructure

Dans les réseaux mobiles, on cherche à faire communiquer entre eux des composants informatiques (ordinateur muni d'un périphérique adéquat, téléphone portable, capteurs ...) appelés nœuds et dont la position est variable. Il existe une multitude de critères permettant de classer les réseaux mobiles. Du point de vue de l'étude des réseaux ad hoc, le plus intéressant est de commencer par opérer une distinction basée sur la nécessité plus ou moins grande d'une infrastructure fixe pré-existante. On peut alors distinguer :
– les réseaux avec infrastructure. C'est notamment le cas des réseaux cellulaires utilisés par la téléphonie mobile. Leur fonctionnement s'appuie sur la présence d'unités fixes communiquant avec un ensemble de nœuds mobiles via des ondes électromagnétiques, mais également entre eux par un réseau filaire.
– les réseaux sans infrastructure, dits aussi ad hoc. L'idée est ici de n'utiliser que les nœuds pour transporter l'information.
– les réseaux hybrides, constitués d'un squelette fixe autour duquel gravitent des nœuds mobiles pouvant cependant communiquer entre eux directement (comme dans [eFV03]).

I.3.a Réseaux mobiles avec infrastructure

Il s'agit du type classique de réseau sans fil, et de loin le plus courant. Le squelette de tels réseaux est constitué d'un groupe de sites fixes (point d'accès) auxquels se connectent des terminaux (nœud) mobiles (voir figure 1.1). Les points d'accès sont par ailleurs connectés entre eux par un réseau filaire et éventuellement à d'autres réseaux plus important (internet). On notera que si ce type de réseau fournit une première réponse au problème de mobilité, il ne fait cependant pas complètement abstraction du milieu physique et nécessite donc la présence a priori d'infrastructures fixes, parfois coûteuses, dans certains cas difficiles à mettre en place physiquement, et bien souvent payantes pour l'utilisateur.

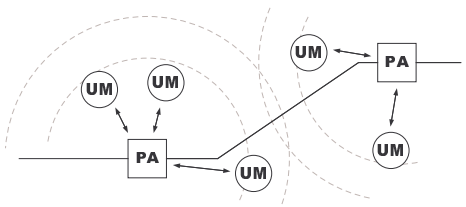


FIG. 1.1 – Réseau mobile avec infrastructure

I.3.b Réseaux mobiles sans infrastructure

Dans un réseau mobile sans infrastructure (voir figure 1.2), chaque unité mobile se comporte à la fois comme un émetteur, un récepteur et un routeur de l'information. Autrement dit, le trajet de données d'un nœud à un autre est constitué d'une succession de sauts entre deux unités mobiles voisines. On parle parfois de *multihoping* pour caractériser le passage de données par des routeurs mobiles par opposition au *single hop* du modèle avec infrastructure. Les réseaux ad hoc sont donc auto-organisés : chaque nœud doit œuvrer au bon fonctionnement général du réseau. En théorie, le terme ad hoc peut désigner des réseaux auto-organisés ne contenant que des nœuds fixes. En pratique, la mobilité des éléments est fréquement implicite, à l'exception de certaines catégories très spécifiques (comme les réseaux de capteurs).
La grande force de ce type d'architecture réside dans sa capacité à être déployé rapidement pour un coût très faible et surtout indépendamment de son environnement physique. En contre-partie il est plus vulnérable aux variations topologiques décrites dans la section I.2.

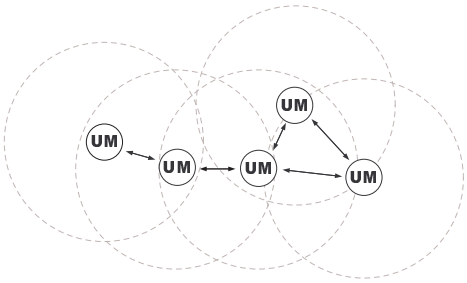


FIG. 1.2 – Réseau mobile ad hoc

Parmi les réseaux ad hoc on trouve notamment les sous-types :

MANET (Mobile Ad hoc Networks) : Ils mettent en avant la mobilité des nœuds en cours d'utilisation du réseau.

VANET (Vehicular Ad hoc Networks) : Variante des précédents où les nœuds sont intégrés à des véhicules mobiles.

Réseaux de capteurs : Les nœuds sont ici des capteurs dispersés dans une zone donnée et ayant pour but la réalisation de mesures physiques. Ils sont caractérisée par l'absence d'un utilisateur associé à chacun des nœuds, une mobilité généralement faible ou nulle et pour chaque nœud une ressource en energie faible.

I.3.c Ondes radio et modulation

Il existe trois types de modulation correspondant chacun à la variation au cours du temps d'une quantité caractéristique de la porteuse (amplitude, fréquence et phase). La modulation de phase comme par exemple PSK (Phase Shift Keying) est de loin la plus employée. Elle consiste à modifier la phase d'un

signal sinusoïdal de fréquence ν à chaque période $1/\nu$ parmi un choix de 2^n valeurs :

$$s(t) = a\cos(2\pi\nu t + \phi(t))$$

$$\phi(t) = \phi_0 + \sum_k \frac{2a_k + 1}{2^n}\pi Rect(\nu t - k)$$

avec $a_k \in [0, 2^n - 1]$ et où $Rect(t)$ correspond à la fonction rectangulaire valant 1 sur $[0, 1[$ et 0 ailleurs. À chaque période, le récepteur peut évaluer les sauts de phase introduits et en déduire un n-uplet de bits. On notera qu'il convient néanmoins d'opérer une synchronisation en début de réception.

Un canal de transmission correspond à une bande de fréquences utilisable pour cette transmission. La porteuse et la technique de modulation doivent garantir que le signal transmis varie dans cette gamme de fréquence. Les canaux autorisés pour les applications utilisées par les particuliers, les entreprises et les différents types d'organisme concernés sont réglementés par le gouvernement de chaque pays ou par des instituts spécialisés. En Europe c'est l'ETSI (European Telecommunications Standards Institute) qui s'en charge.

Le rapport entre la puissance d'un signal à l'émission et celle du même signal à la réception peut s'écrire pour une propagation en vue directe (c'est à dire en considérant le cas où les ondes peuvent se propager sans obstacle) :

$$\frac{P_r}{P_e} = \frac{\lambda^n}{(4\pi R)^n} \times cste$$

où λ est la longueur d'onde de la porteuse, R la distance entre émetteur et récepteur et n un coefficient particulier nommé facteur de décroissance, valant en pratique approximativement 2. La transmission est donc d'autant plus efficace que λ est grand (donc la fréquence ν est petite étant donné que leur produit vaut la célérité) et s'atténue rapidement avec la distance.

I.3.d Détérioration du signal

L'onde rencontre en pratique des obstacles et subit des altérations dues à la diffraction et à la réflexion [Agh04]. Ceci conduit à l'apparition de phénomènes physiques particuliers comme le fading et la dispersion de chemins.

Le "fading" correspond à un affaiblissement du signal, parfois de manière périodique (fading de Rayleigh) qui est en partie limité par les techniques dites de diversité. La diversité spatiale consiste pour le destinataire à utiliser plusieurs récepteurs (antennes) et à combiner les informations fournies par chacun d'entre eux pour déduire le signal reçu avec une probabilité d'erreur beaucoup plus faible. La diversité fréquentielle consiste à envoyer le même message avec différentes fréquences de porteuse. Ces fréquences doivent être choisies suffisamment éloignée les unes des autres afin de garantir leur indépendance mutuelle (en pratique l'écart doit être supérieur à la bande de cohérence, intervalle de fréquence maximal dans lequel le fading, s'il a lieu, altère uniformément le signal sans sélection fréquentielle). Cette technique nécessite donc de pouvoir disposer de canaux de transmission suffisamment étendus, ce qui n'est pas toujours le cas en pratique.

La dispersion de chemins (Fig. 1.3 (a)) provient du fait que l'onde peut utiliser plusieurs trajets pour parvenir au récepteur. Ces chemins étant généralement déphasés, leur superposition rend le signal reçu inexploitable si la différence de marche est trop grande. On utilise en général un étalement de spectre pour résoudre ce problème. L'étalement du spectre par séquence directe consiste à multiplier chaque bit du signal numérique d (valant +1 ou -1) par une séquence binaire pseudo-aléatoire appelée PNcode (Pseudo-Random Noise Codes) c très similaire à un bruit (Fig. 1.3 (b)). Le produit reste lui-même proche d'un bruit d'où une auto-corrélation très faible pour un retard (déphasage) non nul. Si chaque trajet conduit à un retard τ_i le produit du signal reçu avec la même séquence c retardée de façon adaptée (τ) conduit à l'élimination de tous les chemins sauf celui pour lequel $\tau_i = \tau$. L'étalement du spectre par saut de fréquence consiste simplement à changer la fréquence suivant un enchaînement connu de l'émetteur et du récepteur. De cette manière on supprime les interférences dues à certaines ondes parasites plus lentes étant donné que la fréquence de leur porteuse n'est plus écoutée à leur arrivée.

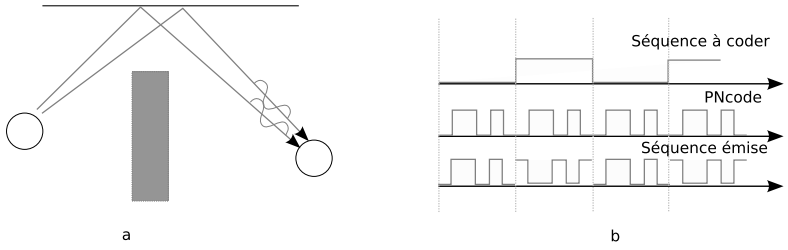


FIG. 1.3 – Dispersion de chemins (a) et étalement du spectre par séquence directe (b)

Enfin l'utilisation du codage canal permet de transformer l'information en un signal redondant dans le but de détecter, voir de corriger ces erreurs. Les perturbations lors du transport aboutissant en générale à des rafales d'erreurs plutôt qu'à des erreurs isolées, on opère généralement un entrelacement avant transmission. Autrement dit, on effectue une permutation sur l'ensemble des bits à transmettre. En cas d'erreurs en rafale, une permutation inverse permet alors d'isoler celles-ci et d'améliorer les performances du code.

I.3.e Méthode d'accès

Le but d'une méthode d'accès (voir [eBSM04]) est de permettre le partage physique du média de communication entre plusieurs nœuds. Elle correspond traditionellement à la couche 2 du modèle OSI. Cependant, dans le monde du sans fil, l'utilisation de techniques de modulation très liées au signal physique est courante.

Lorsque plusieurs signaux sont émis en même temps sur le média en utilisant une même bande de fréquence, les phénomènes d'interférence résultant conduisent, si aucun mécanisme n'est mis en place, à une superposition inutilisable des différents signaux ; on parle alors de collision. Les technologies sans fil impliquent l'existence de problèmes inconnus du filaire qui nécessitent ainsi de nouvelles solutions techniques. La première différence étant simplement que, sur la durée, on ne peut plus connaître avec certitude quels nœuds vont être atteint. Ainsi, certains peuvent par exemple passer hors de portée de

l'émetteur. Cela peut d'un côté constituer un avantage dans le sens où il existe ainsi une limitation naturelle des collisions. Néanmoins, en limitant la détection d'envois de paquets par certains nœuds, le risque de collision augmente en certains points où des ondes provenant de nœuds distants peuvent interférer. Les méthodes d'accès sont donc des mécanismes permettant de s'assurer que tout couple de nœuds voisins peut accéder suffisament souvent au média de communication afin d'échanger des données de manière satisfaisante. Parmi les méthodes proposées, certaines ont une approche basé sur la gestion des paquets (CSMA/CA) d'autres liés à l'utilisation particulière des signaux électromagnétiques (CDMA, FDMA...).

CSMA

Le protocole CSMA (Carrier Sense Multiple Access), utilisé par Ethernet, fait partie des protocoles à compétition. Il reprend l'idée de son prédécesseur Aloha selon laquelle on ne cherche pas à éviter les collisions à tout prix. Si un nœud `V` implémentant CSMA veut transmettre, il lui faut déterminer si le média est libre, auquel cas il émet son paquet. Dans le cas contraire, il attend la fin de l'émission, puis un temps d'attente fixe appelé DIFS, puis enfin un temps aléatoire généré par lui - ainsi si un nœud `W` veut lui aussi transmettre, il est probable que sa durée d'attente soit différente de celle de `V`, ce qui limite donc les risques de diffusion simultanée. Cette technique seule ne résout néanmoins pas totalement le problème des collisions. En effet, `V` peut considérer le média comme libre alors qu'un paquet est en cours de diffusion, mais sans avoir encore atteint `V`. Ethernet résout le problème par couplage avec le protocole CD (Collision Detection). La puissance transmise sur le média augmentant brusquement en cas de collision, il est possible de la détecter, de mettre fin aux transferts en cours et de réessayer par la suite. Hors, dans le monde sans fil, il n'est pas raisonnable de se fier à la puissance reçue, laquelle est également fortement liée à la distance entre les nœuds. Une solution peut alors être d'imposer au récepteur l'envoi d'un accusé de réception en retour. Si l'émetteur n'en reçoit pas il considère le paquet de départ comme perdu et donc à retransmettre. Ce système est baptisé CA (Collision Avoidance) et constitue la méthode d'accès CSMA/CA.

CDMA

CDMA (Code Division Multiple Access) est une méthode basée sur l'étalement de spectre présenté précédemment. L'utilisation de codes permet de distinguer les différents messages potentiellement envoyés par différents nœuds. Les deux variantes sont ici utilisées pour inhiber les interférences entre plusieurs couples communicants :

- L'étalement de spectre par saut de fréquence (FHSS) implique l'utilisation par chaque couple communiquant d'une séquence de fréquences connue des deux nœuds. La source utilise ainsi une fréquence de porteuse donnée pendant une courte période puis change la fréquence utilisé périodiquement. La destination, connaissant également la séquence prédéfinie, écoute successivement les différentes fréquences utilisées.
- L'étalement de spectre par séquence directe (DSSS) nécessite de multiplier le signal original par un

code de fréquence supérieure et s'apparentant à un bruit. Là encore, le code doit être connu de la source et de la destination. Ce code n'étant composé que de valeurs 1 et de -1, le produit du signal reçu par ce même code correspond alors (sous réserve de synchronisation) au signal original. À l'inverse, le produit du signal reçu par tout autre code utilisable correspond à un bruit large bzande de faible amplitude, et donc ignoré. Plusieurs codes peuvent donc être utilisés simultanément par plusieurs couples sans risque d'interférences.

Autre méthodes

FDMA (Frequency Division Multiple Access) exploite l'idée simple de découper un large spectre de fréquence en différentes sous-bandes. Un couple communiquant donné n'utilise alors plus qu'une de ces sous-bandes.

OFDM (Orthogonal Frequency Division Multiplexing) et DMT (Discrete Multi Tone) sont deux méthodes de modulations proches dans lesquelles chaque signal est réparti sur plusieurs bandes de fréquences dont les sous-porteuses sont orthogonales. Bien qu'il ne s'agisse pas à proprement parler de méthodes d'accès, elles sont utilisées dans de nombreuses technologies sans fil.

TDMA (Time Division Multiple Access) choisit quand à lui de découper le temps en périodes durant lesquelles un seul couple communiquant est autorisé à échanger des données.

Dans le cas ad hoc

D'une manière générale, les protocoles d'accès standard ont avant tout été pensés pour les réseaux mobiles avec infrastructure, comme le note [eVSReJD06]. Des variantes ont toutefois été proposées afin de mieux prendre en compte les spécificités ad hoc. L'article [eVSReJD06] répertorie de manière assez exhaustive les différentes méthodes propres aux réseaux mobiles classiques et diverses proposition d'adaptation pour le ad hoc.

I.4 Les technologies sans fil

Nous présentons dans cette section diverses technologies sans fil. Parmi elles, une vaste majorité est dédiée à des réseaux avec infrastructures (bien que certaines proposent les deux). Les différentes technologies sans fil peuvent être classées en fonction de leur taille : réseau sans fil personnels (WPAN), locaux (WLAN), métropolitain (WMAN) et étendus (WWAN).

I.4.a Le Bluetooth et les normes IEEE 802.15

La technologie Bluetooth (voir [eHA04]) a été développée à l'origine par la société Ericsson avant que celle-ci soit rejointe par d'autres au sein du "Bluetooth Special Interest Group". Elle est depuis normalisée par l'IEEE sous le nom de normes 802.15. Il s'agit par ailleurs de l'exemple type de technologie WPAN : destinée à de petits réseaux, dont les équipements mobiles disposent d'une faible portée et d'une energie limitée (souris, casques audio, PC portables, chaînes Hi-Fi...).

Les points d'accès portent ici le nom de "maîtres" par opposition aux nœuds mobiles, qualifiés d'esclaves. La mise en relation d'un maître avec un certain nombre d'esclaves forme un picoréseau. Un même esclave peut cependant appartenir à plusieurs picoréseaux.
Le signal dans un réseau Bluetooth utilise une modulation GFSK sur une bande de fréquence située autour de 2,4 GHz. Afin de permettre à chaque esclave de communiquer sans créer de collision, le Bluetooth utilise un système de multiplexage appelé TDD (Time Division Duplex) consistant à des communications successives entre le maître et chaque esclave. Il comporte par ailleurs un certains nombre de protocoles tels que LMP (configuration et gestion des liens), L2CAP (segmentation et réassemblages des paquets de données) et RFCOM (protocole de transport gérant entre autres des numéros de port).

I.4.b Les normes IEEE 802.11

Nous allons ici détailler les principaux aspects des normes 802.11 [Müh02], devenues un standard de référence pour les communications sans fil. Ces normes sont définies pour les réseaux avec ou sans infrastructure. Ces normes correspondent généralement à des réseaux locaux et ont été prévues aussi bien pour un cadre avec infrastructure que dans un cadre ad hoc. Le terme Wi-Fi a été introduit par la WECA (Wireless Ethernet Compatibility Alliance), organisation composée de fabricants de matériel sans fil créée dans un but d'intercompatibilité. Elle constitue la principale technologie disponible sur le marché.
Les normes 802.11 ne s'appliquent qu'à définir les deux premières couches du modèle OSI ; autrement dit, la couche physique (gérant les problèmes d'échange physique d'information, notamment la modulation du signal et le cablage en filaire) et la couche liaison de données (responsable de la mise en relation des machines via l'adressage physique et le protocole d'accès). Elles se déclinent principalement en 802.11, 802.11a, 802.11b, 802.11g et 802.11n .
Un ensemble de stations dans lequel chaque nœud est à portée d'un autre constitue un BSS (Basic Service Set). Dans le cas spécifique du ad hoc on parle de IBSS (Independent Basic Service Set). Si au contraire le BSS comporte un point d'accès, celui-ci est relié à ses semblables via un réseau filaire constituant le DS (Distribution System).

Le signal physique dans les normes 802.11

802.11 propose deux méthodes. La première, FHSS (Frequency Hopping Spread Spectrum) / GFSK (Gaussian Frequency Shift Keying) consiste en l'association d'un étalement de spectre par saut de fréquence avec une modulation de fréquence. Les sauts sont effectués toutes les 300 à 400 ms parmi 79 canaux d'environ 1 MHz recouvrant l'intervalle 2,4 GHz - 2,4835 GHz selon une séquence prédéfinie parmi un ensemble de 78 possibilités. Autre solution, combiner un étalement de spectre par séquence directe (la séquence aléatoire est alors celle Barker, composée de 11 bits) à une modulation de phase BPSK (Binary Phase Shift Keying) ou QPSK (Quadrature Phase Shift Keying). Le débit fourni est de 2Mbit/s.
802.11b (dite aussi 802.11 HR) s'appuie toujours sur un étalement de spectre par séquence direct utilisant cette fois un codage CCK suivi d'une modulation de phase QPSK. Le débit atteint ainsi les 11 Mbit/s.

Avec cette norme, au maximum trois réseaux peuvent fonctionner simultanément.
802.11a (baptisée Wi-Fi 5 par la WECA) utilise la bande U-NII qui regroupe trois bandes de fréquence de 100 MHz chacune : 5.15 à 5.25 GHz, 5.25 à 5.35 GHz et 5.725 à 5.825 GHz, d'où une incompatibilité avec les normes précédentes. Une bande de fréquence est divisée en 8 canaux (qui permettent l'utilisation simultanée de 8 réseaux) chacun divisé en 52 sous-canaux (48 pour la transmission, 4 pour la gestion d'erreurs). Les sous-canaux sont utilisés parallèlement pour transporter de l'information en bas débit. Cette technique est baptisée OFDM (Orthogonal Frequency Division Multiplexing). Elle est combinée avec une modulation choisie parmi 8 possibilités. Le débit final peut par conséquent varier de 6 à 54 Mbit/s. 802.11a est cependant plus gourmande en énergie que les précédentes normes.
Enfin 802.11g s'appuie sur l'OFDM, cette fois appliquée à la bande ISM avec un codage CCK (Complementary Code Keying). Elle est à compatibilité ascendante avec 802.11b et offre un débit théorique de 54 Mbit/s . Actuellement la plus utilisée, sa portée est d'un peu moins d'une centaine de mètres. Cette norme est en passe d'être supplantée par 802.11n, qui propose un débit théorique de 600 Mbit/s et une portée double grâce à l'utilisation de la technologie MIMO (utilisation de diversité spatiale grâce à plusieurs antennes).

La liaison de données dans les normes 802.11

La couche liaison de donnée reprend les sous-couches LLC, définie dans 802.2, et MAC, adaptation de 802.3 (Ethernet). Cette dernière définit deux protocoles d'accès possible : DCF (Distribution Coordination Function) et PCF (Point Coordination Function).
DCF (Fig. 1.4) reprend le modèle CSMA/CA décrit précédemment. Une fois un paquet correctement transmis sur le média par un nœud `V`, toute station voulant émettre doit obligatoirement attendre une durée DIFS (Distributed Inter-Frame Space) suivie d'un temps aléatoire. Ceci n'est cependant pas valable pour la station destinataire `U` du paquet en question qui, si elle l'a correctement reçu, a seulement besoin de patienter durant un intervalle de temps SIFS (Short Interframe Space, avec évidemment $SIFS < DIFS$) avant envoi d'un accusé de réception. Une collision avec cet accusé est en principe impossible car seul `U` est autorisé à s'approprier le média dans un temps si court. Si `V` ne reçoit rien, une collision a eu lieu avant la réception du paquet par `U` : celui-ci est à retransmettre suivant un principe similaire mais en augmentant le temps d'attente aléatoire de `V`.
Dans le cas de l'ad hoc, l'échange se fait en quatre temps au minimum, en utilisant les services RTS/CTS et NAV. Le nœud `V` voulant émettre envoie tout d'abord un RTS (ready to send), lequel contient l'adresse du voisin destinataire `U`. En cas de non collision, `U` adresse un message CTS (clear to send) à `V` après une durée SIFS. Une fois ce message reçu par `W` celui-ci délivre les données à transmettre après un temps SIFS. Enfin `U` informe de la réception correcte des données par un acquittement ACK. À la fin de l'envoi le canal est de nouveau disponible. Imaginons cependant que lorsque `V` envoie ses données, `W` (voisin de `U` mais hors de portée de `V`) veuille transmettre. A priori rien ne l'en empêche puisque même après une durée DIFS il ne détecte pas le message de `V`. Cependant l'envoi d'un message aboutirait à une collision au niveau de `U`. Pour éviter ce cas de figure, chaque RTS et chaque CTS comporte une indication NAV

correspondant à la durée totale de la transmission prévue. Ainsi, lorsque `U` envoie son CTS il informe entre autre `W` qu'une transmission est en cours avec `V` et donc que tous les autres voisins de `U` (dont `W`) ne sont pas autorisés à émettre avant expiration de cette durée.

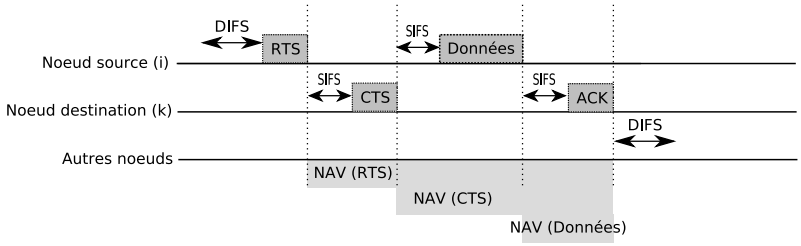


FIG. 1.4 – Le protocole DCF

Les normes 802.11 prévoient également un découpage du message. Selon le principe déjà décrit chaque morceau MESS_n est suivi d'un accusé ACK_n. La durée d'attente entre deux échanges est SIFS.
Le fonctionnement de PCF est basé sur l'existence d'un point d'accès et ne fonctionne pas en ad hoc. Il s'agit d'une méthode à réservation pour laquelle le temps est décomposé en périodes comprenant chacune deux parties nommées intervalles sans contention et avec contention. Lorsque le point d'accès (unique dans une zone donnée) estime qu'une nouvelle période doit commencer, il écoute le canal et attend qu'il se libère. Après expiration d'un délai PIFS (PcF Interframe Space) sans communication, il transmet un paquet particulier, la balise Beacon Frame. La valeur de PIFS (vérifiant SIFS $\leq$ PIFS $\leq$ DIFS) lui assure la priorité sur toute nouvelle transmission mais l'empêche d'interférer avec une transmission déjà en cours. À ce paquet d'ouverture de l'intervalle sans contention fait suite une succession de trames dites CF-Pol dont la fonction est d'interroger une à une toutes les stations couvertes (qui doivent donc être connues d'avance) et éventuellement de leur délivrer des données. Chaque station mobile répond à son CF-Pol par un acquittement ACK éventuellement accompagné de données. Si une station reste muette, le point d'accès poursuit sa liste après un délai PIFS. Ceci fait, on bascule pour une durée donnée dans le mode avec contention correspondant à DCF. À noter qu'aucun constructeur n'a encore vraiment implémenté le mode PCF.

Les services proposés

802.11 fournit un certain nombre de services au niveau de sa couche MAC tels que le mode non connecté (best effort) et le chiffrement, ainsi que certains services dépendant du mode de fonctionnement :

- en mode infrastructure :
  - association-désassociation : gestion de l'association au point d'accès ;
  - distribution : gestion du transport d'une trame vers sa destination via le point d'accès ;
  - intégration : gestion de la communication entre points d'accès via le DS ;
- en mode ad hoc :
  - authentification de la station (optionnel) ;
  - transport des données ;

– sécurité.

En ce qui concerne la confidentialité et l'authentification, les méthodes WPA (Wi-Fi Protected Access) et WPA2 ont supplanté l'ancien mode WEP, jugé peu fiable (Wired Equivalent Privacy).

I.4.c Les réseaux cellulaires

Liés au monde de la télécommunication et notamment à la téléphonie, ces réseaux sont constitués d'un groupe de sites fixes appelés stations de base (BS) réparties de façon à réaliser un pavage d'un espace géographique de taille conséquente (voir [Agh04]). Chaque station de base définit ainsi une portion de territoire appelée cellule (souvent hexagonale) qui exploite une bande de fréquences particulière. La répartition des bandes de fréquences constitue un problème de type coloriage de cartes où l'on cherche en principe à restreindre le domaine de fréquence globalement utilisé par l'ensemble du réseau. Au sein de chaque cellule, la bande de fréquence disponible est divisée entre les utilisateurs appelé terminaux mobiles (MT) qui communiquent de cette façon avec la BS correspondante.
Un des problèmes centraux de ce type de réseau est celui du "handover". Il s'agit de définir comment opérer le changement de BS pour un MT se déplaçant d'une cellule à une autre. Ce déplacement qui implique un nécessaire changement de bande de fréquences ne doit cependant pas conduire à une coupure momentanée de la communication.

1ère génération

Appelée AMPS aux Etats Unis et NMT en Europe, la première génération de réseaux cellulaires découpe une bande de fréquence autour de 800 MHz en plusieurs canaux répartis sur des cellules d'environ 10 à 20 km de diamètres. Chaque cellule peut utiliser au plus 832 cannaux répartis en quatre catégories dédiées notamment à la localisation des MP, à l'assignation des fréquences ou encore au transfert des données (principalement de la voix).

2ème génération

La technologie GSM (Global System for Mobile communication) constitue le premier standard entièrement normalisé pour la communication sans fil. Elle utilise une modulation GMSK sur chacun des canaux de 200 kHz de bande passante extraits de la bande totale 890 - 960 MHz. Par ailleurs, la méthode d'accès TDMA découpe chaque canal en 8 intervalles de temps durant lesquels un seul MT est autorisé à communiquer. Cette génération voit entre autre l'arrivé de deux choses devenus usuels pour l'utilisateur : la carte SIM (qui contient son identité et accroit donc l'indépendance vis-à-vis du terminal utilisé) et du point de vue applicatif les services de SMS (Short Message Service).

3ème génération

La troisième génération est celle de la convergence entre le monde de l'informatique et celui des télécommunications. Les réseaux 3G se proposent en effet d'offrir à l'utilisateur, en plus de la téléphonie, un accès aux services internet (navigation web, messagerie electronique) et multimédia (musique, vidéo, télévision) où qu'il

soit dans le monde. En Europe c'est principalement la téchnologie UMTS qui est utilisée. Elle utilise la méthode d'accès W-CDMA, une variante de CDMA (à séquence directe). Commercialement sa mise en place est considérée comme un semi-échec.

I.4.d Le WIMAX et les normes IEEE 802.16

Les normes 802.16 (maintenant regroupées avec la norme HiperMAN et la technologie WiBro sous l'appellation Wimax) sont prévues pour fournir des débit importants sur des réseaux de types WMAN ou WWAN via l'utilisation de points d'accès (en opposition avec le Wi-Fi qui se concentre sur les réseaux WLAN). Le WiMax (voir [eGR08]) est l'un des prétendants les plus sérieux visant à devenir le standard de référence pour la 4ème génération de réseaux de téléphonie mobile, mais également pour l'accès aux connexions haut-débit par voie hertzienne. Elle est basée sur l'utilisation de la méthode d'accès OFDMA (Orthogonal Frequency-Division Multiple Access) proche de OFDM et sur une couche MAC spécifique très orientée vers la qualité de service.

I.4.e Conclusion

On notera que parmi toutes ces technologies, seules les normes IEEE 802.11 offre a priori la possibilité d'une utilisation en mode ad hoc. Ceci a notamment pour raison que le IEEE 802.11 est la principale technologie concernant les réseaux locaux sans fil. Dans un réseau trop petit, l'intérêt du ad hoc devient en effet inexistant ; dans un réseau trop grand, il devient très difficile d'en garantir le bon fonctionnement, pour des raisons que nous allons voir.

I.5 Conclusion

Les réseaux sans fil se caractérisent par des propriétés particulières liés à la variabilité de leurs topologies physique et logique. Ces particularités ont conduit à reconsidérer en grande partie le fonctionnement des couches basses du réseau (couches physique et liaison de données). Le mode de fonctionnement ad hoc, où tout intermédiaire est lui-même mobile, est cependant resté en marge par rapport au mode avec infrastructure, lequel constitue bien souvent l'unique cadre de référence des normes et standards actuellement disponibles. Une des raisons pouvant l'expliquer est qu'en mode ad hoc, le routage nécessite à son tour d'être repensé en profondeur.

II Le routage dans les réseaux ad hoc

Cette partie expose les différentes caractéristiques du routage en contexte ad hoc. Sont ensuite exposés les principaux protocoles proposés dans la littérature ainsi qu'un tableau de comparaison récapitulatif.

II.1 Caractéristiques du routage en contexte ad hoc

Les algorithmes de routage des réseaux ad hoc sont, comme ceux développés en filaire, basés sur le maintien de tables de routage en chaque nœud d'une part, et d'autre part sur l'envoi de paquets particuliers de taille réduite, les messages de contrôle, qui comportent les informations de mise à jour de ces tables.
Un algorithme performant devrait dans l'idéal prendre en compte :

- *Des caractéristiques spécifiques aux réseaux sans fil* :
  - *La mobilité.* Lorsqu'une machine se déplace, elle est susceptible de perdre certains liens et d'en créer de nouveaux. La topologie du réseau change. Le principal problème consiste alors à déterminer le plus rapidement possible de nouvelles routes. Il faut donc que le protocole ait un temps de propagation très faible pour qu'il soit utilisable dans un environnement extrêmement mobile.
  - *Le fonctionnement distribué.* Aucun nœud ne doit être privilégié et ce en prévision de son éventuelle déconnexion. Pour certains algorithmes, il se peut néanmoins que des rôles particuliers soient attribués à certains nœuds. Ces rôles ne sont cependant que temporaires et doivent pouvoir évoluer avec les variations de topologie.
  - *La consommation d'énergie.* Chaque unité mobile fonctionnant en général à l'aide de batterie, l'envoi excessif de messages aboutit à un déchargement rapide de celle-ci. Il convient donc de surveiller les dépenses énergétiques.
- *Des caractéristiques de sécurité* :
  - *L'intégrité des données.* Ces dernières ne doivent notamment pas être perdues lors de rupture de routes. Il s'agit également de parer à des actes de malveillance comme la présence de "nœuds-puits" qui absorbent les informations sans les diffuser.
  - *La confidentialité des données.* Une information doit pouvoir transiter par un nœud sans qu'elle lui soit obligatoirement accessible.
  - *L'authentification.* Un nœud ne doit pouvoir participer que s'il y est autorisé.
- *Des caractéristiques de qualité de service* :
  - *L'accessibilité au réseau.* Les surcharges doivent être limitées afin de ne pas empêcher de nouveaux nœuds d'accéder au réseau.
  - *Le respect de contraintes de type "temps réel".* Généralement afin de garantir une réception continue et presque isochrone.

Les grandes variations de topologie des réseaux ad hoc imposent donc des contraintes particulières que les méthodes développées pour les réseaux avec infrastructure ne prennent pas nécessairement en compte. Il a donc été nécessaire de mettre au point de nouveaux algorithmes ou d'adapter ceux déjà existant afin d'offrir des solutions tenant compte de cette nouvelle architecture de réseau.

II.2 Algorithmes de routage dans le contexte filaire

Les protocoles ad hoc sont en bonne partie inspirés du filaire et reprennent en les adaptant des méthodes classiques de routage. Aussi trouve-t-on des protocoles utilisant des méthodes à *état de lien* (Link State)

ou bien encore à *vecteur de distance* (Vector Distance). Le but visé est de permettre à chaque nœud de construire un arbre source lui indiquant un chemin vers toute destination, arbre qui doit être le plus à jour possible.

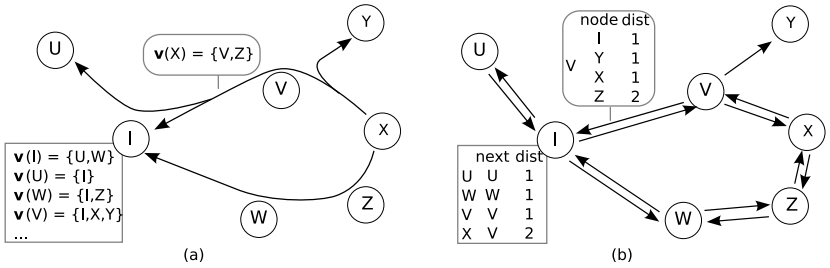


FIG. 1.5 – Algorithmes d'état de lien (a) et de vecteur de distance (b)

II.2.a Protocole de routages à état de lien

Chaque nœud commence par établir la liste de ses voisins et le coût de la communication avec chacun d'eux (Fig. 1.5 (a)). Il diffuse ensuite cette liste partout dans le réseau grâce à un mécanisme appelé *inondation*. Au cours d'une inondation, le paquet reçu pour la première fois par un nœud est automatiquement réémis sur tous les liens. De cette manière il peut rapidement atteindre tous les recoins du réseau.
Une fois cette phase effectuée, chaque nœud du réseau connaît l'intégralité de tous les liens du réseaux (par construction d'une table de topologie). Il peut donc déterminer en appliquant un algorithme particulier (généralement Dijkstra) quels sont les plus courts chemins disponibles (au sens de minimisation du coût). Il met de cette façon à jour sa table de routage en établissant pour chaque destination possible le nœud successeur dans l'arbre source ainsi formé.
Un des défauts de ce type d'algorithme est que chaque nœud doit maintenir une version de la topologie complète du réseau, de sorte que lors de l'apparition ou de la disparition d'un lien, un nouveau calcul du plus court chemin reste possible. Le mécanisme d'inondation aboutit ainsi à un très grand nombre d'échange de paquets. La convergence des tables de routage est en revanche relativement rapide (autrement dit, elles tendent rapidement vers une représentation correcte de la topologie réelle) et l'algorithme n'est pas sensible aux problèmes de type " counting to infinity " (voir II.2.c).

II.2.b Protocole de routage à vecteur de distance

Le but ici visé est non pas que chaque nœud connaisse l'intégralité de la topologie mais qu'il construise petit à petit et directement des routes vers les autres nœuds. La démarche adoptée (Fig. 1.5 (b)) consiste alors à diffuser à ses voisins (et non plus au réseau entier) des informations concernant les chemins choisis pour atteindre chacun des autres nœuds connus (et non plus seulement ses propres voisins).
Le principe est basé sur l'algorithme de Bellman-Ford mais en répartissant ici le calcul sur l'ensemble des nœuds, d'où la dénomination couramment employée DBF (Distributed Bellman Ford). Dans un premier temps, la table de routage d'un nœud comprend seulement la liste de ses voisins et des coûts associés. Il

diffuse alors l'intégralité de cette table à ces mêmes voisins et reçoit symétriquement la table de chacun d'eux, ce qui le renseigne sur de nouveaux nœuds (notamment ses voisins d'ordre 2) et lui permet peu à peu d'agrandir sa propre table. L'itération du processus, au cours duquel un nœud reçoit l'intégralité (ou une partie) des tables de routage de ses voisins permet à chaque étape d'élargir sa propre table de routage. S'il détecte plus d'un chemin possible pour une destination donnée (via des tables différentes), il calcule les coûts associés et ne conserve que le chemin de coût minimal. On finit par aboutir à une stabilisation des tables. Chaque nœud connaît alors le chemin optimal vers toute station `W` donnée, et ce en mémorisant en principe le nœud successeur sur le chemin à `W`.
Au contraire du précédent, ce genre de méthode est généralement sujet à l'apparition de boucles et aux problèmes de type " counting to infinity ".

II.2.c Comptage à l'infini

La figure 1.6 explique le principe du comptage à l'infini. On choisit ici pour distance le nombre de sauts. Dans (a) tous les nœuds peuvent atteindre `U`. Si comme dans (b) l'unique lien à `U` (par `V`) est rompu, la distance pour atteindre `U` est mise par convention à $+\infty$ dans la table de routage de `V`. Celui-ci informe alors ses voisins `W` et `X` qu'il ne peut plus atteindre `U`. Supposons néanmoins dans le même temps que `W` envoie à `X` une mise à jour qui informe ce dernier que `W` peut atteindre `U` en 2 sauts. Si cette information arrive à `X` juste après celle provenant de `V`, le nœud `X` déduit à tort qu'il existe un nouveau chemin pour atteindre `U`, en l'occurrence en passant par `W`. Il diffuse alors l'information selon laquelle il est à une distance de 3 sauts de `U` (c), ce qui permet à `V` et `W` d'actualiser leurs tables. À partir de là `W` et `X` estiment pouvoir atteindre `U` chacun en passant par l'autre (d). Les échanges d'informations à propos de `U` entre ces deux nœuds conduisent à une incrémentation illimitée de la distance à `U`, d'où le nom de comptage à l'infini.

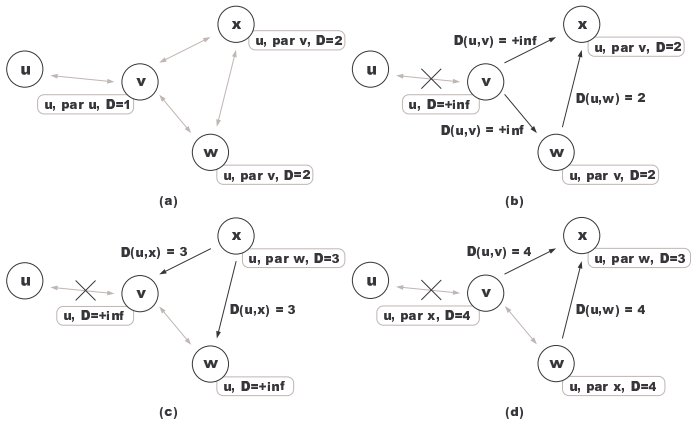


FIG. 1.6 – Comptage à l'infini

II.3 Les classifications des protocoles de routage pour le ad hoc

Les classifications courantes des protocoles ad hoc peuvent s'appuyer sur des caractéristiques variées comme l'utilisation ou non de localisation spatiale (type GPS) ou encore l'utilisation de rôles spécifiques à certains nœuds. Classiquement c'est le choix du moment où la recherche de route doit être effectuée qui est retenu comme principal critère de classification. On distingue de fait les algorithmes dit :

- *proactifs* (table driven), qui mettent continuellement à jour de nouvelles routes quelle que soit le volume de donnée à échanger ;
- *réactifs* (on demand), qui recherchent une route à la demande, autrement dit lorsqu'un paquet de données doit spécifiquement être transmis à une destination donnée ;
- *hybrides*, c'est-à-dire proactifs sur une certaine distance et réactifs au delà.

À l'heure actuelle, le groupe de travail MANET (Mobile Ad hoc Network), créé par l'IEFT en 1997 et principal organisme de normalisation des protocoles pour l'ad hoc, a retenu 4 des protocoles proposés. Il s'agit de AODV, TBRPF, OLSR (déjà normalisés) et DSR (en cours de normalisation).

Par la suite, on adoptera la notation pointée `Object.Attribute` pour la description d'algorithmes. `V` désigne un certain nœud du réseau. Chaque nœud possède un identifiant unique également noté `V`. Cet identifiant peut selon le protocole correspondre à une adresse MAC ou a une adresse IP ; des méthodes d'adressage sont proposées dans [eEMReSRD02], [eMZ02], [eRP02b] et [eRP02a]. Si `T` désigne une table, `T[z]` correspond à une ligne (ou entrée) de la table indicée par `z`, `T[z].Attribute` à la valeur d'un certain attribut dans cette même ligne et `T[#].Attribute` à la colonne qui contient les valeurs prises par cet attribut pour toutes les entrées.

On se placera dans le contexte ou les dénominations source et destination font référence à la transmission de données de niveau supérieur à celle du routage et par conséquent non décrite ici. Le nœud source est donc celui qui cherche à transmettre une information de nature indéfinie dans le cadre de ce paragraphe, tandis que le nœud destination est celui qui doit utiliser cette même information ; chacun des deux pouvant toutefois être dans cette optique tour à tour émetteur, récepteur ou même intermédiaire de messages de contrôle.

II.4 Principaux protocoles proactifs

II.4.a Le protocole DSDV (Dynamic Destination-Sequenced Distance-Vector Routing Protocol)

DSDV [ePB94] est, comme son nom l'indique, une amélioration du routage par vecteur de distance. Il suppose les liens symétriques.

Table de routage et messages de contrôle

Un nœud `V` stocke une table de routage `V.RoutingTable` dont chaque entrée comporte entre autre pour chaque destination `W` du réseau :

- `RoutingTable[W].NextHop` : l'adresse du successeur sur le plus court chemin à `W` ;

– `RoutingTable[W].Dist` : la distance à `W` choisie comme le nombre de sauts ;
– `RoutingTable[W].SN` : un numéro de séquence, défini par `W` et correspondant au numéro de la dernière information reçue par `V` ;

et un numéro de séquence personnel `V.SN` qui, utilisé dans les mises à jour, permet de distinguer la plus récente.

La mise à jour de la table de routage peut se faire soit par réception complète de chacune des tables des voisins (ce qui nécessite l'envoi de plusieurs paquets de données par table), soit en se limitant aux entrées ayant subi un changement (sous la condition qu'elles ne soient pas trop nombreuses). Ainsi, chaque nœud `U` crée un nouveau paquet de mise à jour soit après un intervalle périodique donnée, soit à la découverte d'un nouveau voisin, soit encore à la réception d'un autre paquet de mise à jour. Le paquet de mise à jour généré par `U`, noté `Pck`, n'est transmis qu'à ses voisins. Il contient l'adresse `Pck.Source` de son générateur `U`, la valeur du numéro de séquence `SN` de `U`, incrémentée avant émission, ainsi qu'une table `Pck.Updates` dont chaque entrée comporte les éléments :

– `Updates[W].Count` : la distance entre `W` et `U` ;
– `Updates[W].SN` : le dernier numéro de séquence connu de `W`.

Mise à jour de la table de routage

À la réception d'un paquet `Pck`, le nœud récepteur `V` vérifie en premier lien si le voisin émetteur est connu. Dans le cas contraire, il le rajoute à sa table. Si le nœud est déjà connu (c'est à dire si `U` est dans `RoutingTable[#]`), il détermine si le `SN` reçu est bien strictement supérieur à celui de la table afin de s'assurer que le paquet reçu n'est pas périmé (dans le cas contraire aucun traitement n'est opéré).

`V` vérifie ensuite pour chaque entrée de `Updates` si la destination `W` est également une entrée de `RoutingTable`. Si ce n'est pas le cas, le nœud est détecté pour la première fois. Il est alors inséré dans `RoutingTable` en choisissant comme successeur le nœud `U` (`RoutingTable[W].NextHop` ← `Pck.Source`) et en déterminant la distance à `W` (`RoutingTable[W].Dist` ← `Pck.Updates[W].Count` + 1).

Sinon le nœud est déjà connu :

– Soit le numéro de séquence associé `Updates[W].SN` est plus récent que celui actuellement enregistré `RoutingTable[W].SN`. Le cas échéant la nouvelle route supplante l'ancienne et on procède à un remplacement où les nouvelles informations sont enregistrées.
– Soit `Updates[W].SN` = `RoutingTable[W].SN` ; autrement dit, il s'agit d'une même information mais qui a circulé par différents chemins. Dans ce cas, il suffit de comparer `Updates[W].Count` + 1 et `RoutingTable[W].Dist` pour déterminer si le nouveau chemin est plus court que celui déjà enregistré. Selon le résultat du test, on remplace ou pas.
– Soit `Updates[W].SN` est moins récent que `RoutingTable[W].SN`, ce qui veut dire que l'information concernant `W` reçue dans ce paquet est périmée et donc à ne pas utiliser.

Par ailleurs, si aucune nouvelle information n'est reçue à propos d'un voisin `U` au bout d'une certain temps on considère que le lien n'est plus valide, auquel cas il est retiré de la table et une mise à jour est générée dans laquelle `Updates[U].Count` vaut $+\infty$.

Remarques

En pratique la retransmission d'information pour une destination donnée n'est pas instantanée. On préfère attendre un certain délai pour garantir que la nouvelle route est stable, autrement dit qu'aucune autre ne risque de la supplanter trop rapidement.

Le DSDV élimine les deux problèmes de boucle de routage et de comptage à l'infini. La diffusion d'une mise à jour reste cependant assez lente. Pour une grande mobilité les pertes sont principalement dues à l'utilisation d'entrées de table périmées.

II.4.b Le protocole OLSR (Optimized Link State Routing Protocol)

Il s'agit d'un protocole par état de lien gérant l'orientation des liens et pouvant retenir des routes multiples pour chaque destination ([ePJ03],[ePMeTCeALeAQeLV01]). Il s'avère particulièrement adapté aux réseaux larges et denses étant donné que l'optimisation qu'il opère croît avec la taille du réseau. OLSR utilise par ailleurs le concept d'interface : un nœud peut posséder plusieurs instances d'écoute et donc en quelque sorte se comporter comme plusieurs nœuds virtuels. On ne tiendra pas compte de cette particularité ci-dessous afin de simplifier la présentation.

L'idée majeure est de réduire l'inondation en affectant à certains nœuds des rôles particuliers. Chaque nœud `V` choisit ainsi parmi ses voisins ses *relais multipoints* (multipoints relays, MPR) qui sont les seuls autorisés à véhiculer l'information reçue de `V`. Les voisins qui ne sont pas des MPR reçoivent et traitent également l'information provenant de `V` mais ne la retransmette pas. Si `U` est un MPR de `V`, `V` est appelé sélecteur MPR de `U`. Le choix des MPR doit assurer que tous les voisins d'ordre 2 de `V` soient atteignables par un de ses relais (Fig. 1.7). L'ensemble des MPR n'a pas besoin d'être optimal mais doit être suffisamment petit pour présenter un intérêt. `V` mémorise également un numéros de séquence `V.SN`.

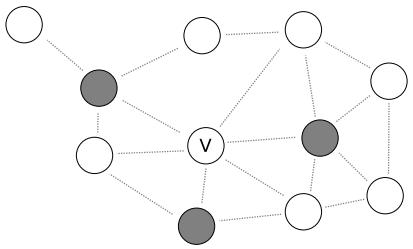


FIG. 1.7 – MPR du nœud `V` (en gris)

Gestion des voisins

Un premier mécanisme de OLSR consiste en une gestion à part de la mise à jour des voisins et la prise en compte de l'orientation éventuelle des liens avec ces derniers. Un lien avec un voisin n'est en effet considéré comme valide que s'il est symétrique. On construit la table des voisins `V.NeighborsTable` :

- `NeighborsTable[U].Status` : le statut du voisin `U` (`SYM` ou `NOT_SYM`) ;
- `NeighborsTable[U].TwoHopsNeighbors` : la liste des voisins d'ordre 2 (bidirectionnels) accessibles via `U` ;

– `NeighborsTable[U].HoldingTime` : l'instant de fin de validité de l'entrée (si trop ancienne, elle est supprimée) ;

Le statut `NOT_SYM` correspond à un voisin asymétrique, c'est à dire entendu mais pas nécessairement entendant. Lorsqu'une entrée apparaît dans la table (nouveau nœud détecté) ou disparaît (`HoldingTime` dépassé), un algorithme détermine parmi les voisins `SYM` un choix de MPR adapté et enregistre la liste `V.MPR`, sous-ensemble de `V.NeighborsTable`.

Chaque nœud `U` transmet périodiquement un message non relayé `Hello` à ses voisins contenant :

– `Hello.Source` : l'adresse du nœud `U` ;
– `Hello.NotMPR` : la liste des adresses des voisins de `U` non MPR (symétriques ou non) ;
– `Hello.MPR` : la liste des adresses des voisins de `U` choisis comme MPR (nécessairement symétriques).

Lorsque ce message est reçu par `V`, il permet de déduire que `U` est au moins un voisin asymétrique. En outre si `V` fait lui-même partie de `Hello.NotMPR` ou de `Hello.MPR` le lien est symétrique. Les modifications correspondantes sont apportées à `V.NeighborsTable` soit en mettant à jour l'entrée de `U` soit en la rajoutant si besoin est. Chaque nœud `V` détermine également une table de `V.MPRSelectorsTable` des nœuds qui l'ont choisi comme MPR.

Table de topologie et messages de contrôle

En plus des informations sur ces voisins, chaque nœud `V` tiens à jour une table de topologie `V.TopologyTable` qui contient :

– `TopologyTable[z].Node` : l'adresse d'une destination `W` ;
– `TopologyTable[z].Neighbour` : l'adresse d'un voisin de `W` ;
– `TopologyTable[z].SN` : le dernier numéro de séquence connu du voisin en question ;
– `TopologyTable[z].HoldingTime` : l'instant de fin de validité de l'entrée (si trop ancienne, elle est supprimée).

On notera que cette table ne possède pas nécessairement une seule entrée par destination `W`. Deux entrées associées à la destination `W` diffèrent par le voisin choisi. En principe il y a au minimum une entrée pour chaque voisin MPR de `W`.

`V.TopologyTable` est construite à partir de données reçues via le réseau sous la forme de messages dits de contrôle de topologie (Topology Control Message) `TC` (Fig. 1.8). Relayés de proche en proche par les MPR successifs ils comportent :

– `TC.Source` : l'adresse du nœud `X` générateur du message ;
– `TC.SN` : le numéro de séquence incrémenté de `X` ;
– `TC.AdvertisedNeighbors` : la liste de voisins de `X` contenant au moins ses sélecteurs MPR.

À sa réception par `V`, un `TC` est analysé afin de s'assurer qu'il a bien transiter par un voisin déjà connu, symétrique ou non (un `TC` ne peut en effet renseigner sur l'existence de lien).

Mise à jour de la table de topologie

Pour comprendre la mise à jour de `V.TopologyTable` grâce à `TC` on doit avoir en tête que la liste

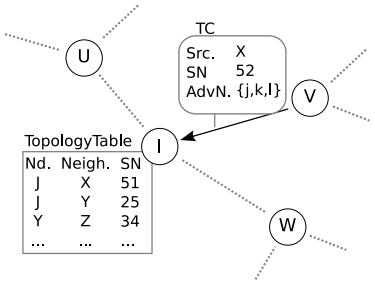


FIG. 1.8 – Transmission d'un message de contrôle de topologie

`TC.AdvertisedNeighbors` est vue par `V` comme un ensemble de destination possible alors que `TC.Source` est ici considéré seulement comme un intermédiaire vers ces mêmes destinations. L'algorithme ci-dessous décrit la mise à jour de la table de topologie :

– Si `TC.Source` est déjà présent dans la colonne `TopologyTable(#).Neighbour` mais que le numéro de séquence de `TC` est inférieur ou égal à celui de la table, rien n'est fait et le traitement du paquet s'arrête.
– Si `TC.Source` est déjà présent dans la colonne `TopologyTable(#).Neighbour` mais que cette fois le numéro de séquence `TC.SN` est supérieur à celui de la table, on commence par supprimer toutes les entrées concernés (qui sont en effet périmées) puis un traitement particulier est opéré pour chaque nœud de `AdvertisedNeighbors`.
– Si le générateur du message n'est pas présent dans la colonne considérée il n'y a pas d'entrée périmée dans la table de topologie. On passe directement au traitement des nœud de `AdvertisedNeighbors`.

En cas de traitement, on considère chaque entrée `W` de `AdvertisedNeighbors`. Chacune est une destination potentielle. Si le couple `(W,TC.Source)` correspond déjà à une entrée `z` de `TopologyTable` (c'est à dire si `TopologyTable[z].Node = W` et `TopologyTable[z].Neighbour` = `TC.Source`), la date de début de validité est rafraîchie. Sinon une nouvelle entrée est ajoutée.

Une fois la mise à jour terminée, si le voisin `U` par lequel vient de transiter `TC` est un sélecteur MPR de `V`, ce dernier retransmet le même message à tous ses voisins. Il est remarquable que la restriction de la diffusion aux seuls MPR permet malgré tout d'atteindre tous les nœuds du réseau. Les `TC` sont en outre générés périodiquement ou bien après la modification de `V.MPRSelectorsTable`. Néanmoins l'envoi consécutif de deux `TC` doit être séparé d'un délai minimum. On introduit un retard si besoin. On remarquera également que l'information véhiculée à propos d'un nœud `W` est entièrement à la charge de ses MPR et donc que si lui-même n'a pas été choisi comme MPR il est inutile qu'il génère des `TC`. Réciproquement, comme `W` possède au moins un MPR, il est assuré que l'information le concernant est effectivement diffusée.

Mise à jour de la table de routage

Enfin, on met à jour une table de routage `V.RoutingTable` à partir de la table de topologie :

– `RoutingTable[W].LastHop` : l'adresse du prédécesseur de la destination potentielle `W` ;
– `RoutingTable[W].Dist` : la distance à la destination via le chemin choisi ;

En pratique le calcul a lieu lorsque `V.TopologyTable` ou `V.NeighborsTable` sont modifiées. Durant ce

dernier, on applique un algorithme du plus court chemin.

Conclusion

En résumé, OLSR utilise des messages `Hello` non retransmis pour périodiquement mettre à jour la liste de ses voisins, ainsi qu'une inondation restreinte des messages de contrôle en élisant des intermédiaires particuliers parmi ses voisins.

II.4.c Le protocole TBRPF (Topology Broadcast Based on Reverse-Path Forwarding Protocol)

TBRPF [eEBReSRD03] est un protocole à état de lien dans lequel chaque nœud `V` établit un arbre source `V.Tree` à partir d'une table de topologie `V.TopologyTable` via une version modifiée de Dijkstra. Il prend éventuellement en compte l'existence d'interfaces multiples sur un même nœud.

Table des voisins et messages HELLO

Le protocole utilise un mécanisme appelé TND basé sur des messages `HELLO` pour maintenir à jour une table `NeighborsTable` de voisins. Contrairement à OLSR l'information correspondante est de type différentielle. Autrement dit, seuls les changements par rapport à l'état précédent sont diffusés. Pour un nœud `V`, la table `NeighborsTable` se présente sous la forme :

- `NeighborsTable[U].Status` : le statut d'un voisin `U` (`1-WAY`, `2-WAY` ou `LOST`) ;
- `NeighborsTable[U].HSEQ` : le numéro de séquence utilisé par `U` dans le dernier `HELLO` reçu par `V` ;
- `NeighborsTable[U].HoldingTime` : l'instant de fin de validité de l'entrée (si trop ancienne, elle est déclarée `LOST`).

Les `HELLO` sont envoyés au moins une fois par période `HELLO_INTERVAL` et contiennent chacun :

- `Hello.Source` : l'adresse du voisin `U` générateur ;
- `Hello.NEIGHBOR_REQUEST` : la liste des adresses des voisins de `U` qu'il peut entendre ;
- `Hello.NEIGHBOR_REPLY` : la liste des adresses des voisins de `U` avec qui le lien est bidirectionnel ;
- `Hello.NEIGHBOR_LOST` : la liste des adresses des voisins de `U` perdus ;
- `Hello.HSEQ` : un numéro de séquence incrémenté pour chaque `HELLO` généré par `U`.

Mise à jour de la table des voisins

Considérons que `V` reçoit des messages `HELLO` générés par son voisin `U`. Si `V` ne connaît pas encore `U`, ce dernier est rajouté à `NeighborsTable` et déclaré temporairement comme `LOST`. Dans tous les cas :

- Si `V` ne retrouve sa propre adresse dans aucune des trois listes et si `U` est actuellement considéré comme `LOST` dans la table des voisins, il devient `1-WAY` et est inséré dans un certain nombre des `Hello.NEIGHBOR_REQUEST` suivants produits par `V`.
- Si `V` retrouve sa propre adresse dans les listes `Hello.NEIGHBOR_REQUEST` ou `Hello.NEIGHBOR_REPLY` de plusieurs messages, `U` est reconnu comme voisin `2-WAY` et inséré dans un certain nombre des `Hello.NEIGHBOR_REPLY` suivants produits par `V`.

– Si V retrouve sa propre adresse dans la liste Hello.NEIGHBOR_LOST, U est déclaré LOST et inséré dans un certain nombre des Hello.NEIGHBOR_LOST suivants produits par V.

Si aucun HELLO n'est reçu de U pendant un certain temps ou si le Hello.HSEQ reçu est beaucoup plus grand que celui enregistré (ce qui signifie que plusieurs HELLO ont été perdus) U est déclaré comme LOST et inséré dans un certain nombre des Hello.NEIGHBOR_LOST suivants produits par V.

Nœuds rapportés et arbre rapporté

Comme il a été indiqué précédemment, chaque nœud V possède en mémoire un arbre source V.Tree dont nous n'avons pour le moment pas détaillé la construction. Le nœud V détermine à partir de ce dernier une liste ReportedNodes correspondant à l'ensemble décrit ci-après. Il commence par s'inclure lui-même dans ReportedNodes. Puis pour chaque voisin U, si U utilise V comme prochain saut pour atteindre d'autres voisins $W_1, W_2, ...$ de V, ces derniers sont rajoutés à ReportedNodes. Enfin, tous les nœuds des branches de l'arbre source V.Tree qui prolongent un voisin précédemment ajouté à ReportedNodes sont à leur tour ajoutés. Les liens (W,W') de l'arbre source tels que W est dans ReportedNodes constituent le sous arbre ReportedTree (voir Fig. 1.9). Du point de vue de V, ReportedTree ou ReportedNodes correspondent en quelque sorte à l'information sur la partie de son arbre source qu'il juge nécessaire de diffuser aux autres nœuds.

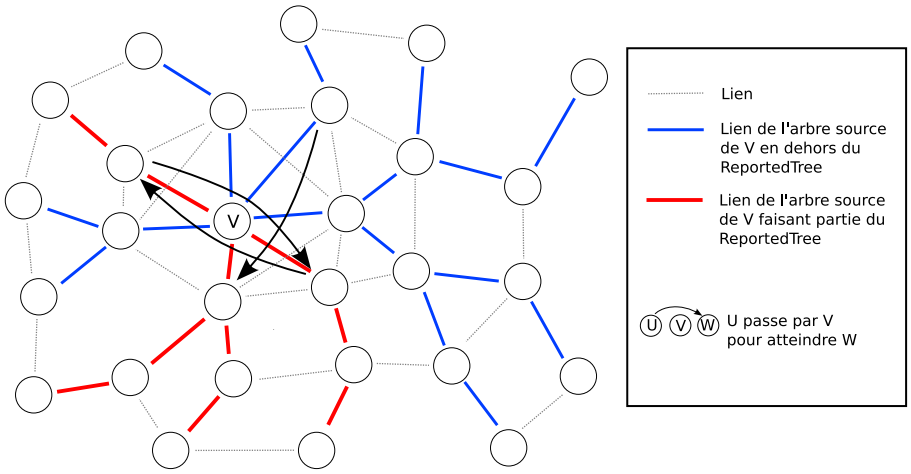


FIG. 1.9 – Arbre rapporté du nœud V

Table de topologie et messages de mise à jour

La construction de la table de topologie n'utilise pas de message à numéro de séquence. Elle est basée sur l'envoi de messages TOPOLOGY_UPDATE qui peuvent être de trois types. Soit W le nœud générateur d'un tel message qui peut être de type :

– FULL : Le message contient, pour un nœud U présent dans le sous arbre W.ReportedTree, tous les nœuds qui suivent U dans ce même sous arbre ;

– ADD : Le message contient, pour un nœud U présent dans le sous arbre W.ReportedTree, les nouveaux

nœuds qui suivent dorénavant U par rapport à la précédente version de W.ReportedTree ;
– DELETE : Le message contient, pour un nœud U présent dans le sous arbre W.ReportedTree, les nœuds qui ne suivent plus U contrairement à la précédente version de W.ReportedTree ;

Chaque nœud informe ses propres voisins des changements dans son arbre source en envoyant :
– au moins une fois par période PER_UPDATE_INTERVAL un certain nombre de paquets (si possible un seul) répertoriant des FULL_TOPOLOGY_UPDATE (un pour chaque nœud de ReportedNodes qui n'est pas une feuille de ReportedTree) ;
– au moins une fois par période DIFF_UPDATE_INTERVAL des TOPOLOGY_UPDATE faisant référence aux variations survenues dans ReportedTree.

La table de topologie V.TopologyTable de chaque nœud V est ensuite utilisée pour construire l'arbre source V.Tree extrait de celle-ci.

Conclusion

En résumé, TBRPF utilise des Hello différentiels pour la gestion des voisins. Les messages de contrôle contiennent soit un certain sous-arbre extrait de l'arbre source, soit à intervalle plus régulier les changements survenus dans ce même sous-arbre depuis les précédentes mises à jour.

II.4.d Les protocoles GSR (Global State Routing Protocol) et FSR (Fisheye State Routing Protocol)

GSR ([eMG98]) a été mis en place dans le but de profiter des avantages cumulés des techniques à état de lien et à vecteur de distance. L'algorithme reprend l'idée de la constitution en chaque nœud d'une table de topologie complète TopologyTable comportant pour chaque destination W une information provenant de W :
– TopologyTable[W].Neighbours : la liste des voisins d'un noeud W ;
– TopologyTable[W].SN : le dernier numéro de séquence de W ;

À partir de celle-ci chaque nœud V calcule des données de routage sous la forme de deux tables : V.NextHop qui mémorise le voisin à utiliser pour atteindre chaque destination W et V.Distance qui contient la distance de V à chaque destination W.

La différence majeure avec l'état de lien réside dans l'abandon du principe de diffusion. GSR reprend l'idée de DSDV où l'information concernant chaque destination W est diffusée de proche en proche et associée à un numéro de séquence fixé par W afin de déterminer la version la plus récente. Cependant là où DSDV choisit comme information la distance à W (ce qui nécessite une modification à chaque étape puisqu'il s'agit d'une quantité relative), GSR la remplace par la liste des voisins de W. La diffusion des messages de contrôle est effectuée périodiquement.

FSR ([eMGeTWC00]) est une extension de GSR dans laquelle la table TopologyTable possède une nouvelle colonne. TopologyTable[W].Distance (obtenue par calcul) indique en effet la distance de V à la destination W et permet de varier la fréquence de diffusion des données de l'entrée concernant W. Ainsi plus TopologyTable[W].Distance est petite, plus l'information concernant W est considérée comme

importante et plus souvent elle va être envoyée au reste du réseau via les messages de contrôle. Cette méthode permet de fait de limiter la multiplication des messages de contrôle.

II.4.e Le protocole HSR (Hierarchical State Routing)

L'idée proposée par HSR [eMGeXHeCCC99] consiste à classer les nœuds suivant une structure hiérarchique en arbre. Les nœuds sont tout d'abord assemblés en groupes (clusters), puis un nœud représentant est élu pour chaque groupe. Un nœud représentant un groupe de niveau n a alors pour fonction de servir d'intermédiaire à l'information destinée à n'importe quel autre membre. En raisonnant au niveau n on peut donc ne considérer que ce type de nœuds. Ils sont alors répartis dans des sur-groupes de niveau $n+1$ pour lesquels de nouveaux représentants doivent être définis et ainsi de suite. À un niveau n donné, il convient néanmoins de pouvoir distinguer les groupes entre eux. Une méthode simple consiste à utiliser pour chacun l'identifiant de son représentant. De fait chaque nœud peut être désigné par le k-uplet des identifiants des différents groupes auquel il appartient. Ce k-uplet est appelée HID (Hierarchical ID), en commençant par le niveau le plus haut jusqu'au plus bas (c'est-à-dire l'identifiant propre du nœud). À noter que le découpage en groupes n'implique pas nécessairement des intersections nulles entre ceux-ci (ce qui conduit donc à l'existence de HID multiples pour certains nœuds).
Pour que le protocole fonctionne, chaque nœud membre de niveau n doit connaitre une route vers son représentant de niveau n. Inversement chaque représentant doit connaitre les routes vers chaque membre de son niveau. HSR utilise en parallèle un adressage dit logique $< subnet, host >$ pour lequel le sous-réseau ($subnet$) correspond à un groupe précédemment défini. Chaque sous-réseau possède un nœud particulier (Home Agent) qui associe à chaque nœud local un HID. Les HID de ces mêmes "Home Agent" sont transmis aux niveaux hiérarchiques supérieurs.
Lorsqu'un message doit être transmit d'un nœud `V` à `W`, `V` utilise l'adresse logique de `W` (la seule qu'il connaisse) pour en déduire l'adresse logique du Home Agent correspondant. Il transmet alors sa requête au représentant de son groupe qui la fait remonter vers le représentant du niveau supérieur. On remonte les niveaux jusqu'à ce qu'un représentant soit capable de localiser le "Home Agent" (dont les HID sont connus des niveaux hiérarchiques supérieurs). Une réponse redescend alors vers `V` lui indiquant le HID en question. `V` délivre son message au Home Agent (par le même procédé de remonté puis de redescente dans les groupes hiérarchiques) qui est alors en mesure de le renvoyer à la destination finale `W`.
L'avantage de HSR est d'une part de permettre une découpage logique du réseau en fonction d'éventuelles relations entre utilisateurs (appartenance à un même organisme ou à une même société), d'autre part de limiter les messages de contrôle, puisque l'information est localisée en certains nœuds uniquement. Néanmoins ce choix s'avère poser certains problèmes. Ainsi si la mobilité de nœuds à faible rôle hiérarchique est somme toute facile à traiter, les représentants de haut niveau ainsi que les "Home Agent" apparaissent comme des points critiques au sens ou leur disparition entraîne une désorganisation importante.

II.4.f Autres protocoles proactifs

On peut citer :

- Zone Based Hierarchical Link State Routing Protocol (ZHLS) [eITL99] : Protocole à état de lien basé sur le regroupement en zone. Le routage se fait sur deux niveaux : de zone en zone, et de nœud en nœud à l'intérieur de chaque zone.
- Distance Routing Effect Algorithm for Mobility (DREAM) [eICeVSeBW98] : Protocole basé sur la localisation des nœuds. La source, lorsqu'elle sait dans quelle direction se situe la destination, envoie ses données aux voisins situés dans la direction en question. Dans le cas contraire il y a inondation. La distance entre deux nœuds détermine la fréquence avec laquelle les mises à jour doivent être générées. Cette dernière croît également avec la mobilité.
- Clusterhead Gateway Switch Routing Protocol (CGSR) [eHKWeWLeMG97] : Protocole reprenant l'algorithme de DSDV appliqué à une structuration du réseau en groupes pour lesquels des représentants sont élus.
- Landmark Ad Hoc Routing (LANMAR) [eXHeLMeGP01] : Protocole basé sur GSR/FSR et utilisant également groupes et représentants.

II.5 Principaux protocoles réactifs

II.5.a Le protocole AODV (Ad hoc On-Demand Distance-Vector Routing Protocol)

Dans AODV ([eEBReSRD03],[eEMBR04]), les tables de routage de chaque nœud sont mises à jour lorsque ceux-ci désirent connaître le chemin vers une destination non répertoriée (ou pour laquelle l'information correspondante est périmée) ou lorsqu'ils participent à une recherche de route lancée par un autre nœud. Ces recherches sont effectuées par l'envoi de requête par la source et l'attente d'une réponse provenant de la destination.

Table de routage et HELLO

Chaque nœud maintient un numéro de séquence `V.SN` et une table de routage `V.RoutingTable`, qui contient les destinations intéressantes pour `V`. Le contenu d'une entrée est le suivant :

- `RoutingTable[W].SN` : le dernier numéro de séquence connu de `W` ;
- `RoutingTable[W].NextHop` : le successeur de `V` (voisin de `V` sur le chemin choisi menant à `W`) ;
- `RoutingTable[W].Dist` : la distance entre `V` et `W` en nombre de saut ;
- `RoutingTable[W].Lifetime` : le temps de vie pour lequel la route est considérée correcte ;
- `RoutingTable[W].Precursors` : la liste des voisins qui utilisent cette route, (c'est-à-dire les voisins pour qui `V` est le successeur dans le chemin menant à `W`).

Chaque nœud conserve à jour une liste de voisins grâce à des messages `HELLO` envoyés périodiquement. Si ces messages ne sont plus reçus, le lien est considéré comme invalide.

Requêtes

Quand un nœud V veut connaître un chemin vers une destination W et que V.RoutingTable ne contient pas d'information suffisante, il incrémente son SN puis lance une recherche de route en diffusant un message "route request" RREQ à ses voisins contenant :

- RREQ.Identifier : l'identifiant du message ;
- RREQ.Dest : la destination à atteindre W ;
- RREQ.DestSN : le dernier numéro de séquence de la destination connu (éventuellement inconnu) ;
- RREQ.Source : la source du message V ;
- RREQ.SourceSN : le numéro de séquence de la source ;
- RREQ.TTL : la portée du message, soit le nombre maximum de sauts pouvant être franchis depuis la source ;
- RREQ.Count : la distance parcourue.

En attente d'une réponse, une copie de chaque RREQ est enregistrée par chaque intermédiaire.

À la réception d'un tel message un nœud U commence par mettre à jour le lien avec dernier saut. Puis il vérifie s'il a déjà reçu le message (mêmes RREQ.Identifier et RREQ.Source) auquel cas cette dernière version n'est pas prise en compte. Sinon, soit :

- U est la destination (U=W) ou connaît une route vers la destination, auquel cas il envoie une réponse à la source ;
- U ne sait pas atteindre la destination mais la distance déjà parcourue a atteint la portée maximale, auquel cas on stoppe la diffusion ;
- la distance est encore inférieure à la portée maximale du message, auquel cas il est diffusé après incrémentation de la distance.

Réponses

En cas de réponse, le nœud U commence par incrémenter son propre SN et diffuse RREP :

- RREP.Dest : la destination atteinte ou désormais accessible (W dans l'exemple) ;
- RREP.DestSN : le numéro de séquence de la destination ;
- RREP.Source : la source du message de requête (V dans l'exemple) ;
- RREP.Lifetime : la durée de vie, période pendant laquelle la route créée est valide ;
- RREP.Count : la distance parcourue.

en utilisant le chemin inverse de celui utilisé par RREQ. En effet chaque intermédiaire connaît alors une route au nœud source, à supposer que les liens soit bien bidirectionnels (Fig. 1.10).

Si aucun message n'est reçu par V au bout d'un certain temps il renvoie un RREQ avec un nouveau NS, un nouvel identifiant et une portée plus grande. Après un certain nombre d'envois sans succés la recherche est abandonnée.

Routage

Grâce aux mécanismes de requêtes/réponses, chaque nœud a pu choisir un voisin privilégié à utiliser pour atteindre la source d'une part, un autre pour atteindre la destination d'autre part (ceux-ci sont stockés

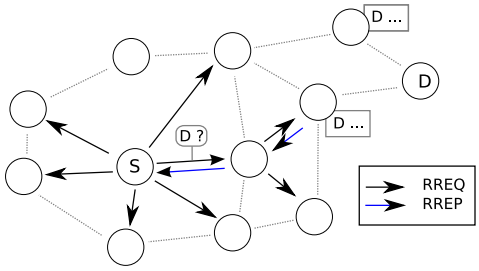


FIG. 1.10 – Circulation de RREP et RREQ dans AODV

dans sa table de routage). Tout paquet de donnée peut alors être envoyé directement sur le réseau : chaque intermédiaire se charge alors de le faire se rapprocher de sa destination.

Message d'erreur

Si une perte de lien est detectée, un `RERR` est envoyé avec :

– `RERR.UDest` : la destination inatteignable ;

– `RERR.UDestSN` : le numéro de séquence de la destination ;

– `RERR.Count` : la distance parcourue ;

et met à jour chaque intermédiaire jusqu'à la source.

Conclusion

En résumé, AODV diffuse en inondation restreinte et pour chaque destination inconnue une demande de route à laquelle répond toute station qui sait atteindre ladite destination. La réponse la plus rapide permet à chaque intermédiaire de définir par quel voisin passer pour atteindre cette destination.

II.5.b Le protocole DSR (Dynamic Source Routing)

DSR ([eDAM96]) fonctionne d'un manière assez comparable à AODV (méchanisme requêtes/réponses) mais inclut cependant certaines spécificités (absence de HELLO, ajout de la route dans les données).

Requêtes

Lorsque la source `V` veut envoyer un message à une destination `W` elle recherche une route complète enregistrée dans son cache de route `RoutingTable`. Si cette route existe, elle est placée dans l'en-tête du message. Sinon le nœud lance une procédure de découverte via une requête de route `RREQ` contenant :

– `RREQ.Source` : l'identifiant de la source `V` ;

– `RREQ.MiddleNodes` : la liste des identifiants des intermédiaires par lesquels le message a transité (vide au début) ;

– `RREQ.Dest` : l'identifiant de la destination `W` ;

– `RREQ.SN` : un nouveau numéro de séquence fourni par `V` ;

envoyé à tout nœud susceptible de la recevoir (autrement dit tout voisin symétrique ou non).

Si le nœud `U` qui reçoit ce message est la destination il répond par un `RREP` envoyé à la source. Sinon il doit vérifier s'il a déjà reçu le message en question (avec la même source et le même numéro de séquence) ou s'il fait déjà partie de la liste d'intermédiaire (présence d'une boucle). Dans ces cas-là, le message est supprimé. Dans le cas contraire, le nœud en question ajoute son identifiant à la liste des intermédiaires avant de diffuser le message. Par ailleurs, il conserve en mémoire l'information relative à la réception de cette requête.

Réponses

Le parcours de la réponse `RREP` dépend des caractéristiques du réseau. Si les liens sont considérés comme symétriques il suffit simplement de suivre le chemin utilisé par `RREQ` en sens inverse. Sinon `W` commence par vérifier dans son propre cache de route `RoutingTable` s'il connaît un chemin à `W`. Si oui ce chemin est utilisé, sinon `W` doit lui-même lancer une requête `RREQ`. Afin d'éviter la multiplication des échanges DSR opte pour une solution de portage (piggybacking) dans laquelle la requête de route générée par `W` est accompagnée de la réponse à `V` (Fig. 1.11).

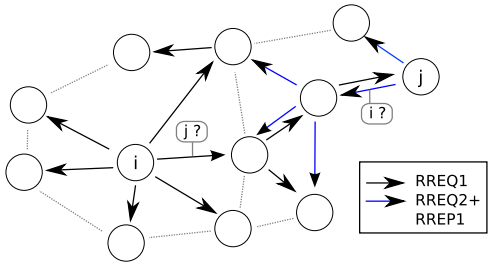


FIG. 1.11 – Portage (piggybacking) de la réponse RREP1 par une nouvelle requête RREQ2

Tout nœud recevant une telle réponse peut l'utiliser pour remplir son cache de route `RoutingTable` même s'il n'est ni source ni destinataire.

Si un nœud `U` différent de la destination `W` recevant une requête connaît déjà un chemin jusqu'à celle-ci, il est autorisé à répondre lui-même à la source `V` en ajoutant le tronçon de route manquant (de `U` à `W`) à la liste `RREQ.MiddleNodes` . Néanmoins il doit avant tout s'assurer que la concaténation des deux morceaux ne forme pas de boucle. De plus, afin d'empêcher qu'un grand nombre de nœuds dans le même cas ne répondent tous en même temps, un temps d'attente aléatoire est observé. Si aucun des paquets reçus par `U` durant cette période ne lui permet de déduire que sa réponse est inutile, un `RREP` est effectivement généré et envoyé à `V`.

Routage

On notera qu'après réception de la réponse `RREQ`, `V` connaît une nouvelle route vers `W`, mais également vers chaque intermédiaire entre lui et `W`. Il envoie alors ses données sur la route en y incluant la liste des nœuds à parcourir. Ceci constitue une particularité peu fréquente dans le monde des protocoles de rouatge : les paquets de données sont modifiés.

En outre, un mécanisme de raccourcissement des chemins est prévu. Lorsqu'un nœud `U` reçoit un paquet

de données n'ayant pas transité par le prédécesseur l prévu dans l'en tête (donc provenant directement du nœud précédant `X` sur la route prévue) il peut déduire que `X` est dès à présent inutile. Il envoie alors un `RREP` à la source `V` contenant une nouvelle route où `X` a été supprimé.

Messages d'erreurs

Comme pour beaucoup de protocoles, des messages d'erreurs permettent d'assurer la maintenance des routes. Elle consiste simplement, lors du transport des données via une route préalablement déterminée à s'assurer qu'à chaque saut le nœud suivant a effectivement reçus les données. Ce mécanisme peut utiliser la couche liaison de donnée si cette dernière requiert systématiquement l'envoi d'acquittement pour chaque message reçu (cas du WiFi). Dans le cas contraire il peut s'appuyer sur une "confirmation passive" qui consiste à vérifier si le nœud suivant cherche à son tour à véhiculer les données. En cas de rupture, un message d'erreur est délivré à la source via le chemin connu.

Conclusion

En résumé, DSR propose une approche comparable à AODV en faisant toutefois circuler l'information concernant chaque intermédiaire entre la source et la destination. L'algorithme pouvant être déployé même en cas de lien unidirectionnel, la réponse peut emprunter un chemin de retour différent de celui utilisé par la requête.

II.5.c Autres protocoles réactifs

On peut citer :

- Cluster Based Routing Protocol (CBRP) [eJLeYCT99] : Protocole basé sur un découpage en groupe disjoints où les représentants seuls font transiter les requêtes de route. Les liens unidirectionnels sont ici utilisables.
- Location-Aided Routing Protocol (LAR) [eNHV98] : Protocole semblable à DSR, utilisant conjointement la localisation géographique afin de limiter l'inondation pour les requêtes de route.
- Temporally-Ordered Routing Algorithm protocole (TORA) [eJLeYCTeVPeSC01] : Protocole pouvant répertorier en chaque nœuds plusieurs chemins (parmi lesquels ne figure pas forcément l'optimal) vers une même destination. Il fait appel à une méthode dite d'inversion des liens qui se différencie de l'état de lien ou du vecteur de distance.
- Associativity Based Routing Protocol (ABR) [Toh96] : Protocole basé sur une métrique particulière liée à la stabilité des liens.
- Relative Distance Micro-discovery Ad hoc Routing Protocol (RDMAR) [eRT99] : Protocole visant à augmenter la réactivité du réseau en limitant la portée des requêtes. Il prend en considération les déplacements des nœuds pour estimer la distance actuelle d'une destination à partir de l'ancienne position connue et du temps écoulé.

II.6 ZRP, un protocole hybride

Peu de protocoles tentent une approche hybride. ZRP [eMRPePS02] constitue pour le moment l'algorithme de ce type le plus connu. ZRP définit pour chaque nœud `V` un groupe appelé zone de routage. Il s'agit de l'ensemble des nœuds dont la distance à `V` est inférieure à un seuil donné. Les nœuds les plus distants de `V` à l'intérieur de sa zone de routage constituent l'ensemble des nœuds périphériques de `V`.
Lorsqu'une source `V` cherche à atteindre une destination `W` elle commence par vérifier si cette dernière appartient à sa zone de routage. Dans le cas contraire, une requête est envoyée à chaque nœud périphérique de `V`. Chacun de ces derniers vérifient à son tour si la destination fait partie de leur propre zone de routage. Si ce n'est pas le cas ils diffusent à leur tour la requête à leurs nœuds périphériques et ainsi de suite jusqu'à ce que la destination soit trouvée. À chaque passage par un nœud périphérique, l'adresse du nœud courant est ajoutée. Par ailleurs une portée est attribuée à chaque requête afin de limiter la recherche dans un certain périmètre autour de la source.
La mise à jour des données relative à la partie proactive du protocole peut se faire en adaptant n'importe quel protocole proactif existant de telle sorte que la diffusion des informations de routage d'un nœud reste limitée à sa zone de routage.

II.7 Protocoles à routes multiples

Les protocoles précédement présentés cherchent à créer, pour chaque couple de nœud souhaitant communiquer, une route les joignant. De nouveaux protocoles étendent les mécanismes connus afin de définir non pas une seule mais plusieurs routes entre chaque couple. On parle alors de protocoles multiroutes. Les protocoles présentés ici sont tous réactifs. Leur spécifité et l'intérêt particulier que l'on porte pour les transferts multiroutes nous incite cependant à les présenter à part.

II.7.a Le protocoles SMR (Split Multi-path Routing)

Basé sur DSR, le but de SMR (voir [eMG01]) est de définir deux routes pour chaque couple communiquant, non nécessairement disjointes. L'idée de base est de ne pas limiter, comme dans DSR, la réponse de la destination à une seule route.
À cette fin, deux modifications sont introduites sur les requêtes RREQ. Primo, elles accumulent les adresses des nœuds traversées (procédé reservé aux réponses dans DSR). Secundo, la réception de plusieurs copies d'une même RREQ par un nœud intermédiaire n'aboutit pas systématiquement à la retransmission d'une seule d'entre elle. En pratique le nœud intermédaire vérifie que les chemins empruntés par ces copies sont à la fois : suffisamment différents les uns des autres (une seule version est conservée entre routes trop similaire) et pas plus long que la route de la première RREQ. De cette façon plusieurs routes concervant une certaine diversité peuvent parvenir à la destination, qui sélectionne alors les k routes jugées les plus disjointes possibles (avec $k = 2$ dans [eMG01]). Une réponse RREP est envoyée sur chacune des routes sélectionnées pour signifier ce choix à la source.

II.7.b Le protocoles AODV Multipath

Le protocole AODV Multipath (voir [eSVKeSKT04]) est une variante de AODV dans laquelle plusieurs routes disjointes sont recherchées.

À la différence d'AODV, AODV Multipath n'autorise pas un nœud intermédiaire à répondre à une requête à la place de la destination. Par ailleurs, les nœuds intermédiaires, même s'ils ne doivent rediffuser qu'une seule copie de chaque RREQ (en utilisant comme habituellement un numéro de séquence) enregistrent néanmoins pour chaque copie reçu dans une table `RREQTable` :

– `RREQTable[z].Dest` : l'identifiant de la destination ;
– `RREQTable[z].Source` : l'identifiant de la source ;
– `RREQTable[z].Dist` : la distance depuis la source ;
– `RREQTable[z].LastHop` : le dernier saut effectué (c'est-à-dire le voisin transmettant la RREQ).

Ainsi, si les nœuds intermédiaires n'augmentent pas le nombre de copies des requêtes par rapport à AODV, ils conservent cependant en mémoire plusieurs façon de retourner à la source.

La destination `D` répond par ailleurs à chaque RREQ reçue, et non plus uniquement à la première. Les réponses sont renvoyés à chaque voisin de `D` ayant fait parvenir une destination. Ceux-ci, puis chaque nœud intermédiaire recevant à son tour une copie de la réponse, enregistre le chemin par lequel cette copie provient (afin de savoir retourner à la destination) choisit dans sa table `RREQTable` le noeud `U` le plus court pour retourner à la source, supprime l'entrée relative à `U` de `RREQTable` et retransmet la réponse à `U`.

Par ailleurs AODV Multipath utilise un mécanisme d'écoute passive des réponses. Si un nœud `W` est témoin du passage d'une réponse par l'un de ses voisins - réponse qui ne lui est cependant pas destinée - il peut toutefois supprimer l'entrée correspondant audit voisin (et au couple communiquant `S/D`) dans sa table `RREQTable`. Ainsi, s'il a par la suite à gérer lui-même une autre copie de cette même réponse, il n'utilisera plus le voisin en question. Grâce à ce mécanisme, si divers réponses parviennent à destination, elle sont a priori disjointes.

II.7.c Le protocoles AOMDV (Ad hoc On demand Multi-path Distance Vector)

Le protocoles AOMDV (décrit dans [eSRD01]), également basé sur AODV se consacre à la recherche de routes multiples disjointes par les liens (et non par les nœuds comme dans AODV Multipath, ceci étant jugé trop restrictif).

Les requêtes porte une information supplémentaire : celle du premier nœud atteint à partir de la source (`RREQ.FirstHop`). Comme habituellement, une seule version de chaque RREQ est rediffusée par chaque nœud intermédiaire. Cependant, chaque intermédaire `V` conserve en mémoire plusieurs voisins ayant transmis une copie de RREQ dans la mesure où les `RREQ.FirstHop` de ces copies sont distincts. Ceci garantit que `V` connait plusieurs routes disjoints par les nœuds pour retourner à `S`. En effet, deux routes distinctes dès le début ne peuvent converger dans la mesure ou chaque intermédaire ne retransmet qu'une seule copie de RREQ.

Lorsque la destination reçoit des copies de la requête, elle répond à k de ses voisins en leur adressant une réponse (k correspondant au nombre de réponses recherchées) et ce indépendamment du `RREQ.FirstHop`. Chaque réponse est ensuite retransmise par chaque intermédiaire à l'un des voisins enregistrés dans sa table de routage. Si plusieurs requêtes arrivent à un même nœud, ce dernier prend soin de les diriger vers des voisins distincts. De fait, les chemins suivis par les RREP sont disjoints par les liens.

II.8 Synthèse sur les algorithmes existants

Un grand nombre d'algorithmes a donc été proposé. Si des similitudes d'approche apparaissent entre certains, les méthodes mises au point sont très diversifiées selon les optimisations recherchées. Le tableau 1.1 fournit un comparatif des protocoles mentionnés.

	Réactivité	Stratégie d'échange d'information	Liens bidirectionnels	Hiérarchie	multiroutes
DSDV	Réactif	Vecteur de distance	Requis	Non	Non
OLSR	Réactif	Etat de lien	Sélectionnés	Relative	Non
TBRPF	Réactif	Etat de lien	Sélectionnés	Non	Non
FSR	Réactif	Hybride	Requis	Relative	Non
HSR	Réactif	Etat de lien		Oui	Non
ZHLS	Réactif	Etat de lien		Oui	Non
DREAM	Réactif			Non	Non
CGSR	Réactif			Oui	Non
LANMAR	Réactif	Hybride		Oui	Non
DSR	Proactif	Etat de lien	Non requis	Non	Non
AODV	Proactif	Vecteur de distance	Requis	Non	Non
CBRP	Proactif		Sélectionnés	Oui	Non
LAR	Proactif			Non	Non
TORA	Proactif	Retournement des liens	Requis	Non	Non
ABR	Proactif	Vecteur de distance		Non	Non
RDMAR	Proactif	Vecteur de distance		Non	Non
ZRP	Hybride	Indéfini	Indéfini	Relative	Non
SMR	Proactif	Etat de lien	Non requis	Non	Oui
AODV Multipath	Proactif	Vecteur de distance	Requis	Non	Oui
AOMDV	Proactif	Vecteur de distance	Requis	Non	Oui

Tab. 1.1 – Tableau comparatif des protocoles

On peut supposer que chaque protocole comporte ses forces et ses faiblesses. Certains resisteront par exemple mieux à l'augmentation de taille du réseau. D'autres favoriseront un échange rapide des données. D'autres encore supporteront plus facilement une grande mobilité des nœuds. Il est ainsi probable que les

propriétés de l'environnement et des prérequis conditionnés par le type d'utilisation soient déterminants dans le choix du protocole de routage le plus adapté.

II.9 Services supplémentaires

II.9.a Sécurité du routage

Plusieurs mécanismes pour sécuriser le routage ont été proposés. La notion de sécurité est particulièrement vaste, sachant qu'un attaquant est susceptible de supprimer, détériorer, dupliquer ou désordonnancer les paquets reçus, voire même de forger des paquets corrompus. Les protocoles proposés se concentrent pour la plupart sur l'authentification des nœuds source et destination afin de garantir qu'aucun attaquant ne peux usurper l'identité d'un des deux. Les problèmes de plus bas niveau (la transmission physique des données et en particulier le protocole d'accès ne sont pas étudiés) et de plus haut niveau (confidentialité et intégrité des données transmises après élaboration des routes) ne sont pas considérés ici.
Les attaques visant les données de routage d'un réseau ad hoc sont de divers types. On mentionne fréquemment :

- usurpation d'identité de la source ou de la destination ;
- falsification des caractéristiques d'une route (notamment utilisation d'un numéro de séquence trop grand pour invalider les autres, ou diminution de la longueur réelle) ;
- suppression de messages ;
- réutilisation d'anciens messages ;
- utilisation de tunnel entre plusieurs nœuds corrompus.

Afin de prévenir de ce genre d'action plusieurs algorithmes utilisent des systèmes de suite de haché, autrement dit, des quantités $(h_0, h_1, ..., h_n)$ où $h_{i+1} = H(h_i)$ et H est une fonction de hachage. Sont également employées des signatures (chiffrement du haché d'une information), de certificat et les méthodes classiques de chiffrement symétrique ou à clée public.

ARAN

ARAN [eBDeBLeEBR02] fait intervenir un nœud particulier T qui joue le rôle de serveur. T délivre à chaque nœud entrant X un certificat $cert_X$ contenant entre autre l'identifiant de X et K_X^+, une clé publique que X a choisi. Chaque nœuds du réseau peut vérifier la validité d'un message signé par X grâce à ce certificat (délivré conjointement) et à la clé publique de T, K_T^+ (connue de tous les nœuds). Avant transmission d'une requête, la source X la signe en utilisant la clé secrète K_X^-. Le résultat et le certificat $cert_X$ sont ensuite transmis. Chaque intermédiaire signe lui-même une seconde fois en prenant soin de tester tout d'abord la signature du précédent intermédiaire puis de la retirer. De plus chaque intermédiaire conserve en mémoire l'adresse du voisin qui lui a fourni le message. À l'arrivée la destination s'assure de l'identité de la source puis effectue le même procédé en sens inverse, chaque intermédiaire sachant alors quel chemin utiliser. Avec cette méthode, chaque nœud qui transmet de l'information a nécessairement reçu l'aval du serveur. Par ailleurs source et destination peuvent s'authentifier mutuellement.

ARIADNE

Le but du protocole ARIADNE [eAPeDBJ02], basé sur DSR, consiste à garantir moyennant un faible coût de calcul que :

– si un nœud destination D reçoit un message de requête de la source S il peut garantir que cette requête a bien été émise par S ;
– si S reçoit une réponse il peut déterminer si la route qu'elle contient est valide (autrement dit, chaque nœud intermédiaire est correcte).

Le bon fonctionnement d'ARIADNE est assuré par l'utilisation de méthode de chiffrement symétrique (chaque couple de nœud possède deux clés, une pour chaque sens de communication) avec un mécanisme d'authentification particulier. Les auteurs mettent en particulier l'accent sur la méthode d'authentification TESLA. TESLA consiste à envoyer à chaque requête de route le `V`-ième élément d'une chaîne de hachés générée à partir d'un élément aléatoire K_0. En supposant que le K_n de chaque nœud est connu de tous et en diminuant `V` avec le temps on assure qu'il existe un test permettant de déterminer si un message comportant K_i a bien été formé par le nœud supposé. Il suffit pour cela de vérifier si $K_n = H^{n-i}(K_i)$. `V` est une fonction de l'instant t_i où le test est effectué. Cette particularité nécessite donc que chaque nœud comporte une horloge synchronisée avec celles des autres. Le fonctionnement général d'ARIADNE s'effectue comme il suit :

1. La source S calcule, grâce à un algorithme de certificat MAC et de la clé secrète $K_{S,D}$ partagée par lui et la destination D, une signature des données de routage à envoyer $h_S = MAC_{K_{S,D}}(requete)$. La requête contient entre autre la quantité t_i correspondant à la date supposée d'arrivée à la destination.
2. Chaque intermédiaire X calcule le haché de son adresse concaténée à cette quantité $h_X = H(ID_X, h_Y)$, ainsi que sa propre signature $M_X = MAC_{K_{X,t_i}}(requete||route\ parcourue)$ en utilisant comme clé K_{X,t_i}, l'élément relatif à l'instant t_i dans la chaîne fournit par TESLA.
3. La destination vérifie si $h_Y = H(ID_Y, H(ID_X, ...MAC_{K_{S,D}}(requete)...))$. Elle élabore alors une réponse, la signe avec $M_D = MAC_{K_{S,D}}(reponse||route\ complete)$ et l'envoie à la source par le chemin contenu dans le message.
4. Chaque intermédiaire ajoute au message la clé K_{X,t_i} qu'il a précédemment utilisé.
5. La source vérifie que chaque clé K_{X,t_i} est valide, si oui que la signature de la destination M_D est valide, puis chaque signature M_X est valide.

ARIADNE garantit qu'une information de routage modifiée par un nœud malveillant est rejetée. Un nœud ne peut en particulier pas détourner un trafic vers lui s'il n'est pas sur le chemin optimal, ni créer de fausses routes. Par ailleurs une sécurisation des messages d'erreur est également effectuée.

SEAD

SEAD [eDJeAP02] est un protocole proactif à vecteur de distance inspiré par DSDV. Le mécanisme de protection est très similaire à celui de TESLA mais ne fait cependant pas intervenir de calcul fonction

du temps. Chaque entrée diffusée par X est associée avec un haché $K_{X,i}$ correspondant :

– soit à un certain élément $K_{X,i}$ de sa propre séquence $K_{X,n}, ..., K_{X,1}, K_{X,0}$ pour l'entrée concernant X lui-même ;

– soit au haché de $K_{Z,i'}$, quantité précédemment associée à cette entrée pour tout autre nœud Z.

L'élément $K_{X,i}$ est choisi en fonction du nouveau numéro de séquence associé à la mise à jour - plus le numéro est grand, plus `V` est petit. Etant donné qu'à chaque saut on opère $K_{X,i+1} = H(K_{X,i})$, il dépend également de la distance parcourue depuis la source. Un tel mécanisme garantit alors qu'il n'est pas possible pour un nœud malveillant ni d'augmenter le numéro de séquence pour faire croire à une route plus récente, ni de diminuer la distance parcourue.

SAODV

SAODV [Zap05] est une version sécurisée de AODV. À l'envoi d'une requête ou d'une réponse chaque nœud émetteur X génère une nouvelle chaîne de hachés $K_n, ..., K_1, K_0$ et transmet K_n et $Hash = K_0$ en plus des données. Les données du message sont par ailleurs signées, à l'exception de $Hash$ et de la distance parcourue $Dist$ (nulle au début). Chaque intermédiaire vérifie si $K_n = H^{n-D}(Hash)$. Si oui, il incrémente D et calcule $Hash \leftarrow H(Hash)$. Les données étant signées, un nœud malveillant ne peut fabriquer de fausse route.

SRP

Dans SRP [eZJH02], autre protocole réactif, chaque couple de nœud partage d'une clé secrète $K_{S,D}$. Celle-ci est notamment utilisée pour signer la requête envoyée par S. À chaque saut, un nœud intermédiaire ajoute son adresse au message. La destination D peut seule lire la signature. Elle forme alors sa réponse en incluant le chemin parcouru et en signant le tout avec $K_{S,D}$. La réponse progresse en suivant le chemin inverse jusqu'à S qui peut vérifier à son tour de la validité du message reçu.

II.9.b Multicast

Un certain nombre de protocoles a été développé à partir des protocoles standards afin d'intégrer des fonctions multicast dans lesquels un paquet peut être délivré à plusieurs destinations. L'objectif de ce genre d'approche est d'éviter que certaines données particulièrement demandées ne nécessitent autant de transmissions que de récepteurs. Dans une approche multicast, un seul envoi est théoriquement suffisant pour atteindre plusieurs nœuds et ainsi limiter la charge du réseau. Là encore, les variations topologiques fréquentes constituent le principal obstacle à l'adaptation des méthodes élaborées pour le filaire.

Les applications susceptibles d'utiliser l'envoi multicast sont multiples tels que les services de chat (messagerie instantanée), la vidéoconférence, le calcul réparti, le travail collaboratif, la réplication de base de données, les jeux en ligne ...

Bien que l'approche proactif/réactif soit encore utilisable ici, d'autre critères sont également à prendre en compte. Il s'agit notamment de la représentation en arbre ou en maillage du réseau (respectivement tree-based et mesh-based protocols). Dans l'approche en arbre, une source voulant envoyer un message en

multicast utilise un arbre source (un seul chemin vers n'importe quelle destination). Dans un maillage il existe au contraire plusieurs chemins. Cette approche est moins efficace en terme de calcul mais néanmoins plus adaptée face à la mobilité des nœuds. En outre, le nœud à l'origine de l'envoi multicast (la source ou les destination) est un troisième aspect à prendre en compte.

Ci dessous les principaux protocoles :

- MAODV (Multicast Ad-hoc On-Demand Distance Vector routing) [eCEP99] basé sur AODV, réactif, initié par les destinations, utilisant une topologie en arbre ;
- ODMRP (On-Demand Multicast Routing Protocol) [eMGeCCC99] réactif, initié par la source, utilisant une topologie en maillage ;
- ADMR (On-Demand Associativity-Based Multicast) [eDBJ01] réactif, initié par la source, utilisant une topologie en arbre ;
- ABAM (On-Demand Associativity-Based Multicast) [eGGeSB00] réactif, initié par la source, utilisant une topologie en arbre ;
- MZR (Multicast Zone Routing) [eAASeDS01] initié par la source, utilisant une topologie en arbre ;
- SRMP (Source Routing-based Multicast Protocol) [eHL02] réactif, utilisant une topologie en maillage ;

II.9.c Qualité de service et réseau ad Hoc

La qualité de service (QoS) [eRNeBReHS98], [Mer05] peut se définir comme un ensemble de besoins et de performances attendues vis-à-vis de l'utilisation d'un type particulier de service sur un réseau. Assurer une QoS c'est assurer que certaines propriétés particulières nécessaires au bon fonctionnement du service en question sont conservées pour tous les modes de fonctionnement du réseau envisagés. Certains protocoles de routage ad hoc comme CEDAR [eRSeVB99] ont été spécifiquement créés pour prendre cette dimension en compte. Néanmoins les techniques mises en œuvre pour garantir cette QoS peuvent concerner tous les aspects de la transmission d'information, notamment l'accès au média, la réservation de route, la gestion des files d'attente, etc ... L'apparition de cet axe de recherche coïncide avec une nouvelle utilisation des réseaux informatiques. L'ancienne approche, dite *best effort* ou "au mieux" (notamment employée par IP) ne permet pas de garantir les contraintes attendues par les nouveaux services multimédias - dont la voix sur Internet constitue l'exemple typique - et offre par conséquent une efficacité décroissante à mesure que ce genre de services se développent.

Les systèmes à qualité de service participent généralement à lutter contre le caractère erratique du flot des paquets échangés. Une des causes est la congestion du réseau (saturation de la bande passante par une quantité d'information trop importante au même moment) qui aboutit au ralentissemnt ou à l'arrêt de certains flux en des nœuds intermédiaires. Ceci entraine de fait une perte de qualité à la réception. Le principal mécanisme utilisé consiste à adapter le fonctionnement du réseau à la nature des flux qu'il doit traiter. Cela implique donc d'une part qu'il soit possible de caractériser les flux de données en transite, d'autre part que les informations relatives à cette classification soit accessibles par les nœuds routeurs. On procède par exemple, via la mise en place de contrats, à la réservation de débit et de mémoire dans

les fils d'attente des routeurs pour privilégier certains flux ayant des contraintes temporelles importantes. Dans [eWSeALeKC00], [eSHSeKN02] et [eGSAeXZeATC00] des modèles particuliers de qualité de service pour les réseaux ad hoc ont été mis au point.

FQMM

Le modèle FQMM [eWSeALeKC00] s'appuie sur les modèles existant en filaire IntServ et Diff-Serv. Il fonctionne sur une gestion de réservation des ressources hybride entre les méthodes "par flux" (où la priorité d'un flux et le traitement associé lui est propre) pour les trafics prioritaires et "par classe" (où les flux sont regroupés en classes aux comportements différents) pour les autres. Reposant sur la couche IP, il attribue aux nœuds d'entrée le rôle de classer les flux et aux suivants celui de gérer les priorités correspondantes. Le routage doit garantir des routes adaptées à la quantité d'information. Par ailleurs, les protocoles utilisés autorisent une communication simple entre le réseau et Internet.

iMAQ

Le modèle iMAQ [eSHSeKN02] répartie ses fonctionnalités en trois couches :

– la couche application qui partage ses données multimédias ;
– la couche "middleware" qui assure la localisation, l'accès et la reproduction de données ;
– la couche de routage ad hoc.

Chaque nœud diffuse périodiquement des informations sur les données qu'il possède. La couche centrale assure par ailleurs la réplication de données en plusieurs points afin de lutter préventivement contre le partitionnement du réseau.

INSIGNIA

INSIGNIA [eGSAeXZeATC00] est un protocole de réservation de ressource utilisant le champ options des paquets IP pour limiter les messages envoyés. Distinguant les trafics "best effort" des flux temps-réel, il permet une adaptation continue de la réservation des flux (notamment grâce à des rapports de QoS fournis périodiquement par la destination), une gestion de l'ordonnancement des paquets, de l'acceptation ou du rejet de nouveau flux et un contrôle d'accès MAC adapté à la QoS.

II.10 Simulation et comparaisons des protocoles

La majeure partie des simulations et évaluations des protocoles de routage sont actuellement réalisés en utilisant une des versions de l'application *Network Simulator* (ns). Ce logiciel libre, développé conjointement par plusieurs centres de recherche, permet la simulation de routage dans diverse configuration.
Même si l'utilisation de communication sans fil est possible, notamment grâce à l'implémentation de la couche MAC 802.11 et de certains algorithmes de routage, il est à noter que le nombre de ces derniers reste assez faible. De fait les études comparatives jusqu'à présent menées délaissent une partie importante des protocoles disponibles. Par ailleurs, les simulations de réseaux ad hoc sur NS impliquant un nombre important de nœuds et d'envoi buttent généralement sur les limites des machines standard en terme de

capacités de calcul. [eTG03] propose pour ce problème des modifications du code afin d'améliorer les performances du simulateur.

La plupart des simulations menées ([eRQ00],[eDMeDJeYCHeJJ98] et [eZYeBQeJH04]) utilise un modèle de déplacement appelé "Random way point" proposé dans [eDAM96]. Dans ce dernier, un certain nombre de nœuds est réparti sur une surface donnée. Chaque nœud choisit une nouvelle position et se déplace vers celle-ci à vitesse constante choisie uniformément dans $[V_{min}, V_{max}]$. Une fois arrivée il fait une pause à cet endroit pour une durée caractéristique de la simulation appelée "pause time" avant de définir une nouvelle destination à une nouvelle vitesse. [eRQ00], [eDMeDJeYCHeJJ98] et [eZYeBQeJH04] définissent tous un espace de 1200×300 m, et pour chaque nœud une bande passante de 2 Mb/s ainsi qu'une portée de 250 m.

[eDMeDJeYCHeJJ98] se concentre sur la capacité à obtenir une nouvelle route en cas de perte. Il compare pour cela AODV-LL, DSDV-SQ, DSR et TORA sur des simulations de 50 nœuds et $900s$ avec des débits de type CBR. 210 scénarios sont générés qui recouvrent les valeurs de "pause time" suivantes : 0, 30, 60, 120, 300, 600 et 900s. Les auteurs montrent que la proportion de paquets perdus reste faible pour AODV-LL et DSR, même en cas de haute mobilité. Concernant les paquets de contrôle, leur nombre est constant dans DSDV-SQ quel que soit la mobilité et le nombre de sources. Il est faible pour AODV-LL et DSR en cas de faible mobilité. DSDV-SQ et DSR utilisent plus facilement des chemins optimaux.

[eRQ00] compare les protocoles DSDV, DSR, AODV, SSA pour l'envoi de paquets TCP. 50 scénarios de 25 nœuds sont étudiés pour 5 vitesses possibles et 2 valeurs de "pause time" pour une durée de simulation de 200 s. Les auteurs concluent que DSR offre de meilleures performances en cas de mobilité moyenne alors que SSA obtient de bon résultat à haute vitesse étant donné qu'il sélectionne les routes stables. AODV possède quant à lui de bons résultats généraux.

Dans [eZYeBQeJH04] les protocoles DSDV, DSR, AODV, TORA sont comparés sur un ensemble de 280 scénarios de fonctionnement d'une durée de 600 s chacun. La simulation utilise 25 nœuds se déplaçant à une vitesse maximum de 1,5 ou 20 m/s suivant les cas. Sont comparés le taux de paquets reçus, le temps moyen de trajet, le nombre total de paquets de contrôle générés et le temps d'acquisition d'une route. Les auteurs concluent que DSDV est principalement adapté pour les petits réseaux à faible capacité. TORA quant à lui fonctionne mieux dans un environnement dense et très mobile. DSR utilise peu de paquets de contrôle, ce qui est avantageux dans des réseaux à faible bande passante. AODV est le plus performant de tous.

III Conclusion

Les perspectives de liberté offertes par les réseaux ad hoc sont à l'échelle des difficultées conceptuelles qu'ils posent. Afin de prendre en compte la mobilité extrême des nœuds et l'absence d'éléments stables, des propositions très variées ont été faites, en particulier à propos du routage. Parallèlement les exigences en terme de sécurité ou de qualité de service sont peu à peu prises en compte. Les simulations de réseaux ad hoc se sont jusqu'à présent concentrées sur des comparaisons entre les protocoles de routage les plus

connus. Elles restent cependant encore peu représentatives de ce que pourrait être un cas d'utilisation réelle.

En outre, le postulat d'intégrité des données véhiculées via les mécanismes de routage est fondé sur le franchissement de liens que l'on espère suffisamment stables. En cas de disparition de ceux-ci, le protocole doit être à même de garantir qu'une part minimum des informations de routage est malgré tout diffusée, et ce, le plus rapidement possible. L'utilisation simultanée de plusieurs routes pour une même transmission (autrement dit un même couple source destination) est une autre solution proposée qui permet d'être moins exigeant sur leur validité respective ; la durabilité des chemins dans un réseau ad hoc n'étant de toute façon de toute façon jamais garantie. La représentation multiple de l'information fournit dans cette optique des méthodes de répartition de l'information que l'on peut supposer avantageuses, puisqu'elles permettent de répartir les données sur le réseau tout en limitant la surcharge.

Chapitre 2

Représentation multiple de l'information

Le codage regroupe l'ensemble des techniques permettant de modifier la représentation de l'information. Classiquement, on sépare en théorie de l'information le codage source du codage canal. Le premier vise à diminuer le volume total de l'information en s'appuyant, d'une part sur l'a priori que le signal à coder possède certaines propriétés statistiques particulières, d'autre sur la possibilité de supprimer de l'information jugée non indispensable. Le second tente au contraire d'accroître ce même volume en introduisant de la redondance, et ce afin d'augmenter la résistance du signal à d'éventuelles dégradations. Ces deux opérations sont en principe effectuées séparément. Néanmoins, de plus en plus de méthodes conjointes source-canal voient le jour. Les méthodes dites de descriptions multiples peuvent notamment être utilisée dans cette optique.
On présente donc dans ce chapitre une introduction générale sur le codage de l'information, ce qui nous permet d'aborder dans un second temps le principe du codage par descriptions multiples. Enfin, une section est dédiée à la présenatation en détail d'une transformation mathématique permettant notamment de réaliser ce type de codage : la transformation Mojette.

I Méthode de codage source standard

On peut distinguer les codes avec ou sans perte d'information selon qu'ils permettent ou non de reconstituer le signal original. Nous nous proposons d'expliquer le principe des codes avec perte. On peut en effet considérer qu'ils généralisent l'ensemble des codes, au sens où un code avec perte comprend généralement des étapes sans perte. Chaque méthode de codage consiste en la description d'un codeur $\mathfrak{C}$ et d'un décodeur $\mathfrak{D}$ tels que pour un certain type de signal x d'entrée $\mathfrak{C}(x)$ représente une quantité d'information éventuellement moindre (égale si le codeur est sans perte), et que $\mathfrak{D}(\mathfrak{C}(x))$ fournit une bonne approximation de x (vaut x si le codeur est sans perte).

I.1 Format de la source

L'idée de "quantité d'information" dépend bien entendu de la nature des signaux traités. Du point de vue de l'informatique, seuls les signaux numériques (c'est-à-dire discrets) sont manipulables. On considère donc qu'à l'exception éventuelle de l'entrée X, les transformations effectuées ont toutes valeurs dans des ensembles finis.

Etant donné un espace probabilisé Ω et une mesure de probabilité $\mathbb{P}$, on considère $X \in \left(\mathcal{A}^{\mathcal{N}}\right)^{\Omega}$ un vecteur aléatoire de dimension $\mathcal{N}$ à valeurs dans un ensemble $\mathcal{A}$ (ce cas est considéré comme suffisament général dans notre présentation). Dans le cas où $\mathcal{A} = \mathbb{R}$, on peut associer à X une densité de probabilité f_X. X doit rendre compte de l'ensemble des formes possibles du signal à coder et surtout de leurs probabilités d'apparition. Chaque coordonnée aléatoire $X_k \in \mathcal{A}^{\Omega}$ modèlise un élément particulier de la source appelé symbole aléatoire (par exemple un pixel d'une image, l'amplitude d'un signal sonore à un instant donné, une lettre). On notera que le vecteur aléatoire X ne prend pas nécessairement en compte l'intégralité de l'objet de départ. On peut ainsi imaginer que ce dernier est découpé en une suite (finie) de vecteurs $(X^{(1)}, X^{(2)}, \cdots)$. Chaque $X^{(k)}$ est alors codé indépendamment des autres. Les $X^{(k)}$ peuvent ou non constituer des quantités identiquement distribuées et/ou indépendantes. Ces différentes situations justifient l'existence de techniques distinctes, plus ou moins adaptées dans chaque cas.

I.2 Les étapes du codage

Un codeur $\mathfrak{C}$ est toujours décomposable en deux opérations successives [Goy00], (même si l'une peut être absente) :

- un codeur par quantification $\alpha \ : \mathcal{A}^{\mathcal{N}} \to \mathcal{I}$ où $\mathcal{I}$ est un ensemble fini d'éléments. Pour fixer les idées, on peut considérer chaque élément comme une chaîne de bits ;
- un codeur entropique $\gamma \ : \mathcal{I} \to \mathcal{B}$ avec $\mathcal{B} \subset \{0,1\}^*$ et $\{0,1\}^*$ l'ensemble des chaînes de bits de longueur finie.

De même un décodeur $\mathfrak{D}$ associé est constitué de :

- le décodeur entropique inverse $\gamma^{-1} \ : \mathcal{B} \to \mathcal{I}$;
- un décodeur de reproduction $\beta \ : \mathcal{I} \to \mathcal{A}^{\mathcal{N}}$.

Un couple codeur / décodeur adapté à la source X doit fournir une estimation $\hat{X} = \mathfrak{D}(\mathfrak{C}(X))$ de X suffisamment proche. Dans une approche sans perte on souhaite même obtenir $\hat{X} = X$. En pratique on définit une distance d sur $\mathcal{A}^{\mathcal{N}}$ permettant de juger la proximité entre deux données déterministes. Cette métrique permet alors de définir la distorsion introduite par le couple codeur / décodeur pour la source X par la valeur : $\Delta = \mathbb{E}[d(X, \hat{X})]$. Une distorsion couramment utilisée pour des données numériques ($\mathcal{A} = \mathbb{R}$) est la distorsion sur l'erreur quadratique (MSE) définie pour la distance $d(x, \hat{x}) = \frac{1}{\mathcal{N}} \sum_{k=1}^{\mathcal{N}} (x_k - \hat{x}_k)^2$. Il est dans notre intérêt de garantir une petite valeur de Δ qui correspond à une bonne estimation. Néanmoins il convient de s'assurer que parallèlement les chaînes produites en sortie de $\mathfrak{C}$ ont une longueur petite afin de ne pas perdre la principale propriété recherchée, à savoir la compression des données. Dans cette optique on définit le débit du code comme le rapport entre le nombre moyen de bit produits en sortie sur

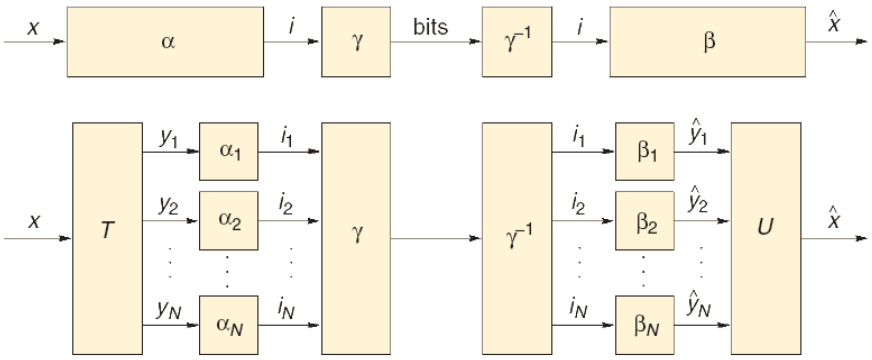


FIG. 2.1 – Décomposition d'un codeur / décodeur

le nombre d'éléments en entrée :

$$R = \mathbb{E}[\text{len}(\mathfrak{C}(X))]/\mathcal{N} \tag{2.1}$$

où len $: \{0,1\}^* \to \mathbb{N}$ est la fonction associant à une chaîne de bits sa longueur. Un code pour une source X de dimension $\mathcal{N}$ est dit optimal à un débit R s'il n'existe pas d'autre code qui au même débit réalise une distorsion inférieure.

Etant donné que le choix de $\mathcal{N}$ est libre, il est possible d'ajuster ce paramètre pour obtenir des codeurs plus efficaces (en pratique en augmentant $\mathcal{N}$). Néanmoins le gain ainsi opéré est faible en comparaison de l'augmentation de la complexité du calcul. Ce constat est d'autant plus vrai lorsque le code se rapproche d'un code optimal. Dans un souci d'efficacité on cherche donc en pratique à réaliser des codes suboptimaux pour lesquels la calculabilité reste abordable.

I.2.a La quantification

La quantification (réalisée par α et β) est une opération avec perte dans laquelle $\mathcal{A}^{\mathcal{N}}$ est partitionné en un ensemble fini $(\mathcal{S}_i)_{i\in\mathcal{I}}$ de sous-ensembles généralement connexes indicés par les éléments de $\mathcal{I}$. $i = \alpha(x)$ correspond alors à l'indice de l'ensemble contenant x. Cette étape (non injective) peut être utilisée s'il est nécessaire de réduire la quantité d'information. L'idée est que les valeurs contenues dans un même sous-ensemble sont plus ou moins assimilables les unes aux autres. Aussi, la mise en place d'un "bon" découpage est fortement liée aux propriétés probabilistes de X. $\hat{x} = \beta(i)$ est un vecteur de $\mathcal{A}^N$ généralement choisi dans $\mathcal{S}_i$.

Dans beaucoup d'applications pratiques où $\mathcal{A} = \mathbb{R}$ on privilégie certains types de quantification. Tout d'abord une transformation $T : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^{\mathcal{N}}$ bijective est appliquée à la source dans le but de décorréler ses composantes. Autrement dit, on souhaite que la matrice d'autocovariance de $Y = T(X)$ soit diagonale (il s'agit alors de la transformation dite de Karhunen-Loève). Il s'agit en fait de pouvoir quantifier chaque coordonnée indépendamment plutôt que d'appliquer une quantification vectorielle généralement coûteuse en calculs. On opère donc pour chaque coordonnée y_k d'une réalisation y une quantification scalaire

$i_k = \alpha_k(y_k)$. Le découpage de $\mathbb{R}$ en intervalles pour chaque coordonnée y_k peut néanmoins être différent. Pour faire le lien avec le cas général, chaque $\mathcal{N}$-uplet de symboles en sortie $(i_1, ..., i_\mathcal{N})$ est ici assimilé à un symbole i de $\mathcal{I}$. Le découpage $(\mathcal{S}_i)_{i\in\mathcal{I}}$ de l'espace $\mathbb{R}^\mathcal{N}$ des valeurs de y correspond dans ce cas particulier à un découpage en parallélépipèdes de dimension $\mathcal{N}$ (c'est-à-dire en produit cartésien de $\mathcal{N}$ intervalles). L'opération β est dans ce cadre formé sur un schéma symétrique. Elle est composée de $\mathcal{N}$ reconstructions scalaires β_k, tels que $\hat{y}_k = \beta_k(i_k)$ est une estimation de y_k, suivies d'une transformation $U : \mathbb{R}^\mathcal{N} \to \mathbb{R}^\mathcal{N}$. On a éventuellement $U = T^{-1}$. Les transformées T et U sont fréquemment choisies comme linéaires. Ce nouveau choix permet de simplifier les calculs, notamment pour des valeurs importantes de $\mathcal{N}$. Les deux précédentes restrictions (linéarité et quantifications scalaires) empêchent évidemment l'obtention de codes optimaux pour la plupart des lois de probabilités de X.

I.2.b Le codage entropique

Les codeurs et décodeurs entropiques γ et γ^{-1} réalisent deux opérations inverses l'une de l'autre. Il s'agit d'une pure réduction de la taille des données effectuée sans perte. Le but recherché est d'associer à $i \in \mathcal{I}$ une chaîne de bits de longueur la plus petite possible. Si $I = \alpha(X)$ a une répartition uniforme (aucun symbole i de l'alphabet $\mathcal{I}$ n'a plus de chance de se produire qu'un autre), il n'existe pas de méthode pour raccourcir statistiquement la longueur du résultat. Dans le cas contraire le codage entropique (comme par exemple les codages de Huffman et arithmétique) associe à chaque élément i de $\mathcal{I}$ une chaîne de caractère donnée issu d'un dictionnaire $\mathcal{B}$. On passe en réalité d'un alphabet de symboles à taille fixe à un nouvel alphabet de symboles à taille variable. Plus la probabilité d'apparition d'un symbole i est grande, plus la chaîne associée est courte. Deux approches sont en fait envisageables :

1. Dans un code non adaptatif, on utilise un dictionnaire prédéfini, connu à la fois du codeur et du décodeur, et déterminé à partir de l'ensemble des réalisations possible de I.
2. Dans un code adaptatif, on utilise un dictionnaire variable, initialement défini, et tel que lors du traitement de chaque partie $x^{(k)}$ d'un message total, le dictionnaire soit ré-adapté simultanément par le codeur et le décodeur en fonction du nombre d'occurrences déjà rencontrées pour chaque $i^{(k)} = \gamma(x^{(k)})$.

Le code adaptatif ne présente bien entendu d'intérêt que lorsque les segments $i^{(k)}$ ont des lois de probabilités variables et sont non indépendantes. Par ailleurs, le flux d'éléments $\cdots, \gamma(i^{(k)}), \gamma(i^{(k+1)}), \gamma(i^{(k+2)}), \cdots$ se présentant sous la forme d'une chaîne de bits, la possibilité de pouvoir en extraire les éléments $\gamma(i^{(k)})$ et donc d'appliquer γ^{-1} correctement implique certaines conditions sur la forme des mots de $\mathcal{B} \subset \{0,1\}^*$. Il faut en l'occurence que $\mathcal{B}$ soit un code préfixe, autrement dit qu'aucune chaîne $b \in \mathcal{B}$ ne constitue le préfixe d'une autre $b' \in \mathcal{B}$.

I.3 Optimisation

L'optimisation de la quantification et du codage entropique peut s'effectuer séparément. Considérons en premier lieu l'étape de quantification $\alpha : \mathcal{A}^\mathcal{N} \to \mathcal{I}$. Le débit en sortie de α est noté R_{qut}. Partant de

l'a priori que tous les éléments de $\mathcal{I}$ sont exprimables comme des mots binaires de longueur fixe, le débit correspond à :

$$R_{\text{qut}} = \mathbb{E}[\text{len}(\alpha(X))]/\mathcal{N} = \log_2 |\mathcal{I}|/\mathcal{N} \tag{2.2}$$

Plus ce débit est faible, plus la quantification est grande. Ceci garantit un nombre limité de symboles dans $\mathcal{I}$ pour représenter les valeurs possibles de la source $\mathcal{A}^{\mathcal{N}}$, mais diminue en revanche la précision possible des données reconstruites.

D'un point de vue mathématique la problématique du codage optimal consistent en la détermination pour un $\mathcal{N}$ fixé de l'ensemble des couples (R, Δ) pour lesquels il existe un couple codeur décodeur réalisant un codage à débit R et tels que la distorsion entre le signal de départ et son estimation vaille Δ. On parle alors d'une paire (R, Δ) atteignable. La fermeture de l'ensemble précédemment décrit constitue la région $R\Delta$. Il s'agit d'une partie convexe de $\mathbb{R}^{+2}$ dont la frontière est décrite par une courbe fonctionnelle $\Delta_{min}(R)$ (ou de manière équivalente $R_{max}(\Delta)$).

Si $X \in \left(\mathbb{R}^{\mathcal{N}}\right)^{\Omega}$ a pour densité de probabilité la fonction f_X et pour variance σ^2, on montre que :

$$\frac{1}{2\pi e} \cdot 2^{2h} \cdot 2^{-2R} \leq \Delta_{min}(R) \leq \sigma^2 \cdot 2^{-2R} \tag{2.3}$$

avec l'entropie différentielle $h = -\int f(x) \log_2 f(x) dx$. Dans le cas d'une source X gaussienne, on a par ailleurs $\Delta_{min}(R) = \sigma^2 2^{-2R}$.

La performance d'un code entropique $\gamma : \mathcal{I} \to \{0,1\}^*$ vis à vis d'une source $I \in \mathcal{I}^{\Omega}$ est quant à elle quantifiable par la "longueur du code" :

$$L(\gamma) = \mathbb{E}[\text{len}(\gamma(I))] = \sum_{i \in \mathcal{I}} \mathbb{P}(I = i) \cdot \text{len}(\gamma(i)) \tag{2.4}$$

Claude Shannon a montré que, théoriquement, si γ est un code entropique préfixe optimal, cette longueur vaut l'entropie de la source :

$$H(I) = \sum_{i \in \mathcal{I}} \mathbb{P}(I = i) \cdot \log_2 \mathbb{P}(I = i) \tag{2.5}$$

II La description multiple

Les méthodes précédemment développées s'appuient sur un codage à description unique. Autrement dit, pour une réalisation x de la source X, un code $\mathfrak{C}(x)$ est calculé puis éventuellement découpé en N paquets $P_1, \cdots, P_N$ consécutivement envoyés sur un même canal de transmission avant décodage. La réception de tous les paquets est alors nécessaire pour reconstituer x (ou dans le cas général son estimation $\hat{x}$).

Les codes progressifs constituent une amélioration du cas précédent dans laquelle la reconstitution de la source peut débuter avant réception de l'intégralité des paquets. Les premiers paquets permettent donc d'établir une première estimation de la source. À mesure que l'on en reçoit de nouveaux, la qualité de la reconstitution tend à augmenter au sens ou la distorsion entre données originales x et estimation $\hat{x}$ diminue.

Un inconvénient majeur des codes à description unique (progressifs ou non) reste néanmoins que la perte d'un paquet entraîne un arrêt temporaire du processus de reconstitution du message original x. Il est alors nécessaire de procéder à un renvoi, ce qui peut grandement augmenter le temps de réception de x dans sa totalité. Même si les codes progressifs sont utilisables après réception d'un nombre restreint de paquets, tout déséquencement dans l'ordre de ces derniers occasionne un retard dans la reconstitution puisqu'un paquet est généralement inexploitable si les précédents n'ont pas été reçus. Le codage à description multiple permet de résoudre ce problème.

II.1 Le principe et contexte d'utilisation

Le codage à description multiple (codage MD) [Goy01] prend le parti de répartir l'information différemment. En effet le codeur fournit non pas une seule mais plusieurs chaîne de bits $D_1, D_2, ...D_N$ appelées descriptions. Chacune d'entre elle constitue une unité indépendamment transmissible. Qui plus est, différents canaux peuvent être utilisés dans ce but. Chaque description ne contient qu'une partie de X, mais de manière "équivalente" à toutes les autres descriptions. Autrement dit la reconstruction peut débuter quelles que soient la ou les descriptions reçues en premier.

Alors que dans le cas à description unique, la perte d'un paquet bloque la reconstruction et nécessite sa retransmission, dans le cas description multiple la perte de paquet est moins grave. La réception d'un nombre suffisant de paquets (quels qu'ils soient) permet ici de reconstituer une partie du message initiale (voire l'intégralité). Il s'agit donc d'une forme de codage conjoint. D'une part, la possibilité de reconstruire partiellement et progressivement l'information implique que le couple codeur/décodeur a une connaissance de la sémantique de l'information (caractéristique du codage source) ; d'autre part, les descriptions sont indépendamment manipulables (transmissibles) et peuvent naturellement comporter de la redondance (caractéristique du codage canal).

Le codage MD a été introduit dans un contexte d'envoi ininterrompu de données à travers un réseau. La fiabilité de liens n'étant jamais parfaitement garantie, la méthode alors utilisée consistait à rediriger l'information dès la détection de la perte d'un lien via une autre route prévue à cet effet. Outre des temps de détection de perte et de redirection indésirables, ce système nécessitait donc l'emploi d'un ou plusieurs canaux parallèles inutiles en temps normal, mais pouvant potentiellement être mis à contributions à tout moment.

Actuellement le codage multiple présente un intérêt dans un contexte où s'applique au moins l'une des conditions suivantes :

- la perte de données en cours de transmission est fréquente ;
- l'ordre de réception des données transmises peut différer de l'ordre de l'émission ;
- différents niveaux de qualité de l'information sont acceptables.

II.2 Modélisation

Le cas à deux descriptions est particulièrement détaillé dans la littérature (voir par exemple [eSC81], [Vai93] ou [eJKeJc02]). Considérons une source $X = (X_k)_{k\in[1,\mathcal{N}]}$ à l'entrée d'un codeur. Le résultat est réparti en deux descriptions corrélées D_1 et D_2, de débit respectif R_1 et R_2. Après transmission sur deux canaux différents, un décodeur central récupère les deux signaux pour former son estimation $\hat{X}^0$. La distorsion associée est $\Delta_0 = \mathbb{E}[d(X, \hat{X}^0)]$. On lui adjoint deux décodeurs supplémentaires tels que chacun n'a accès qu'à l'une des descriptions. Les estimations associées $\hat{X}^1$ et $\hat{X}^2$ permettent le calcul des distorsions Δ_1 et Δ_2.

Mathématiquement parlant, le problème revient ici à se placer dans un espace $\mathbb{R}^5$ et similairement au cas précédent à considérer les quintuplés $(R_1, R_2, \Delta_0, \Delta_1, \Delta_2)$ atteignables. D'un point de vue plus pratique, on s'arrange généralement pour obtenir des descriptions aux propriétés semblables. Autrement dit, on se limite à $\delta = \Delta_1 = \Delta_2$ et $r = R_1 = R_2$. Le but généralement visé est alors de déterminer un codeur $\mathfrak{C}$ minimisant Δ_0 avec δ et r contraints :

$$\arg\min_{\mathfrak{C}} \Delta_0(\mathfrak{C}) \tag{2.6}$$

$$r = R_1 = R_2 \leq R_{budget} \tag{2.7}$$

$$\delta = \Delta_1 = \Delta_2 \leq \Delta_{budget} \tag{2.8}$$

El Gamal et Cover ont notamment montré dans [eTC82] que dans le cas d'une source X de coordonnées iid et de loi f_X, pour la métrique d décrite précédemment, le quintuplé $(R_1, R_2, \Delta_0, \Delta_1, \Delta_2)$ est atteignable si et seulement si il existe une loi de distribution $f_{\hat{X}_0,\hat{X}_1,\hat{X}_2|X}$ définissant des variables aléatoires $\hat{X}_0, \hat{X}_1, \hat{X}_2$ telles que :

$$\forall i \in \{0,1,2\}, d(X, \hat{X}_i) \leq \Delta_m \tag{2.9}$$

$$R_1 > I(X, \hat{X}_1) \tag{2.10}$$

$$R_2 > I(X, \hat{X}_2) \tag{2.11}$$

$$R_1 + R_2 > I(X, (\hat{X}_0, \hat{X}_1, \hat{X}_2)) \tag{2.12}$$

où $I(X, Y)$ définie l'information mutuelle de Shannon. Dans [Ahl86], Ahlswede s'intéresse à un cas similaire où la source est à valeur discrète.

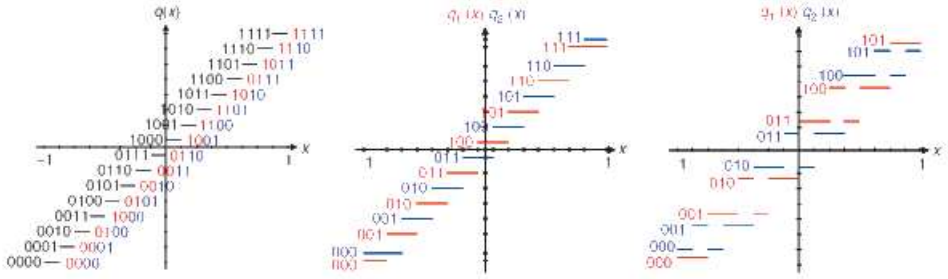
II.3 Quelques méthodes proposées dans la littérature

Diverses méthodes de codage par description multiple sont ici décrite. On notera que la source peut être vue comme la réunion de diverses informations distinctes. Il peut être alors préférable de protéger certaines d'entre elles plus que d'autres. La qualité des approximations des informations plus sensibles croit alors plus rapidement avec le nombre de descriptions reçues. On parle dans ces cas-là de protection inégale. La protection égale considère pour sa part que l'information n'est pas fragmentable, ou que les

parties qui la compose sont d'égale importance, et donc qu'il n'y a pas lieu d'établir une variation dans la protection.

II.3.a Protection égale

Une première méthode [eSC81] consiste à découper la source en plusieurs ensembles supposés équivalents. Dans le cas d'un signal audio, cela peut consister à séparer les échantillons de rang pair et impair avant d'appliquer une compression à chaque sous-flux. L'idée est ici d'utiliser directement la redondance inhérente à la source (deux images successives d'une même vidéo se ressemblent), puis d'interpoler l'information manquante. Une méthode similaire pour la vidéo est détaillée dans [Wan05] où les images sont découpées en lignes de "macroblocs" (GOB), eux même répartis en deux ensembles.
On peut également appliquer différentes quantifications à la même source (définies par différents partitionnements $(\mathcal{S}_i^k)_{i\in\mathcal{I}^k}$) telles que la connaissance conjointe des sous-ensembles $\mathcal{S}^k$ contenant le message x permette d'en déduire une approximation $\hat{x}$ plus précise (elle doit en effet appartenir à $\bigcup_k \mathcal{S}^k$) comme dant l'exemple de la quantifiacation scalaire de la figure II.3.a. Le pavage de $\mathbb{R}^{\mathcal{N}}$ peut être fait de manière fine, puis des regroupements distinct entre les pavets permettent d'obtenir différentes quantifications. Cette approche est notamment celle retenue par les méthodes de quantification par réseau (sous-groupe discret de $\mathbb{R}^{\mathcal{N}}$ ([eJKeJc02]) et générateur au sens des espaces vectoriels). On commence par élaborer un réseau principal Λ correspondant au niveau de reconstruction le plus précis. Des sous réseaux $\Lambda_1, \Lambda_2, ..., \Lambda_N$ généralement optenus par homothétie et rotation de Λ fournissent alors autant de descriptions.



Vaishampayan utilise une assignation d'index (index assignment) dans [Vai93]. Il s'agit d'une fonction $A : \mathcal{I} \to \mathcal{J} \times \mathcal{J}$ appliquée après la quantification qui associe à l'entier i indexant la partition de départ un couple d'index $I(i) = (j, k)$. L'ensemble $\mathcal{J}$ étant généralement inclue dans $\mathcal{I}$ on représente souvent l'assignation d'index par une matrice $(a_{jk})_{jk\in\mathcal{J}\times\mathcal{J}}$ telle que $a_{jk} = i$ si jk est produit par l'entrée i et a_{jk} n'a pas de valeur sinon. Ce procédé est très largement repris ([eSSH04]). Dans [eEMR02], les auteurs étudient les assignations d'index permettant les distorsions les plus faibles en cas de perte.
Les méthodes par transformation linéaire cherchent à répartir l'information des composantes d'une source $X \in \mathbb{R}^{\mathcal{N}}$ sur celles de $Y = \mathfrak{M}\cdot X$ où $\mathfrak{M}$ est une matrice de taille $\mathcal{M}\times\mathcal{N}$. Certaines approches pour lesquelles $\mathcal{M} = \mathcal{N}$ cherchent à augmenter la corrélation entre composantes. De cette manière la perte de certaines coordonnées Y_k autorise malgré tout l'estimation de toutes les composantes de X. Ces méthodes, dites par transformées corrélantes (Correlating Transforms), sont exposées dans [eJK01]. Dans d'autre méthodes

par transformation linéaire (avec lesquels $\mathcal{M} > \mathcal{N}$) on souhaite de manière similaire au codage par bloc standard augmenter la quantité d'information envoyée au total afin de permettre une "pseudo-inversion" de M avec certaines composantes de Y.
À noter que certains articles s'intéressent au cas X_k variable discrète et $\Delta_0 = 0$. Autrement dit, on souhaite pouvoir reconstituer parfaitement une source discrète donnée si l'ensemble des descriptions sont reçues.

II.3.b Protection inégale et encodage prioritaire

Les techniques de codage par protection inégale (Unequal Error Protection ou UEP) utilisent généralement un code progressif. On découpe ainsi l'information en fragments d'importance décroissante. Ces fragments sont ensuite assignés aux descriptions avec une fréquence d'autant plus grande qu'ils sont importants. Dans le cas de deux descriptions de R bits chacunes, on commence par coder les données à taux $(2-\zeta)R$ avec $\zeta \in [0,1]$ tel que les $\zeta \cdot R$ premier bits concentre l'information la plus importante. Les deux autres fragments de $(1-\zeta)R$ bits sont alors couplés avec des copies du premier. Les deux paquets obtenus constituent des descriptions avec protection inégale.
Les méthodes dites de transmission par encodage prioritaire (PET) (détaillé dans [eJBeJEeMLeMS94] et utilisé notamment dans [eHWeVP03]), bien que non complétement équivalentes au codage MD avec protection inégale, partagent néanmoins un point commun avec celui-ci : la possibilité d'ajouter une redondance plus ou moins grande à divers données à transmettre en fonction de leur importance sémantique. Certains modèles de PET combinent cette propriété avec une approche description multiple.

II.4 Conclusion

Nous venons ici d'introduire le concept de description multiple. Ce dernier s'oppose aux autres méthodes de codage en ceci qu'il permet une amélioration progressive de la qualité de l'information reconstruite indépendamment de l'ordre des descriptions reçues. Quelques méthodes ont été par ailleurs esquissées. On se propose maintenant d'expliquer en détail le fonctionnement d'une transformation mathématique (dite Mojette) pouvant être utilisés comme méthode de description multiple avec protection égale ou inégale.

III Transformation Mojette

III.1 Introduction

La transformation Mojette est un outil mathématique proposé par l'équipe IVC du laboratoire IRCCyN (voir les références [eNN05], [eMSeJPG05] et [Gué09]) et inspiré des traveaux de Katz (voir [Kat78]). Il permet de réaliser un codage d'information au moyen d'opérations géométriques simples, correspondant à une discrétisation de la transformée de Radon. Nous allons ci-dessous détailler son formalisme et son utilité dans le contexte de la description multiple.

III.2 Morphologie mathématique

La transformation Mojette fait appel à certains concepts de morphologie mathématique que nous décrivons ci-dessous. On considère ici un groupe commutatif $(E, +, O)$.

III.2.a Dilatation et érosion

Pour toute partie B de E on définit $(B)_y$ le translaté de B par le vecteur y et $\check{B}$ le symétrique de B respectivement par :

$$(B)_y = \{x + y, x \in B\} \tag{2.13}$$

$$\check{B} = \{-x, x \in B\} \tag{2.14}$$

$$\tag{2.15}$$

On définit les opérateurs $\oplus$ (l'addition de Minkowski) et $\ominus$ respectivement par :

$$A \oplus B = \{x + y, x \in A \vee y \in B\} \tag{2.16}$$

$$A \ominus B = E \setminus ((E \setminus A) \oplus B) \tag{2.17}$$

L'opérateur $\oplus$ est commutatif, associatif et d'élément neutre $\{O\}$. L'opérateur $\ominus$ admet $\{O\}$ comme neutre à droite.
La dilatation de A par B est définie comme l'ensemble $A \oplus \check{B}$ qui vérifie :

$$A \oplus \check{B} = \{y, (B)_y \cap A \neq \emptyset\} \tag{2.18}$$

L'érosion de A par B est définie comme l'ensemble $A \ominus \check{B}$ qui vérifie :

$$A \ominus \check{B} = \{y, (B)_y \subseteq A\} \tag{2.19}$$

On notera que :

$$O \in B \Rightarrow A \ominus \check{B} \subseteq A \subseteq A \oplus \check{B} \tag{2.20}$$

Erosion et dilatation peuvent en effet introduire une translation inutile. Comme on ne regarde que la forme générale du résultat, considérer un élément érodant ou dilatant contenant l'origine garantit que le résultat d'une érosion ou d'une dilatation reste confiné dans un certain espace.

III.2.b Ouverture et fermeture

On définit $A \circ B$ l'ouverture de A par B comme le résultat d'une érosion par B suivi d'une dilatation par $\check{B}$:

$$A \circ B = (A \ominus \check{B}) \oplus B \tag{2.21}$$

On définit $A \bullet B$ la fermeture de A par B comme le résultat d'une dilatation par B suivi d'une érosion par $\check{B}$:

$$A \bullet B = (A \oplus \check{B}) \ominus B \tag{2.22}$$

Les opérateurs $\circ$ et $\bullet$ sont idempotents à droite :

$$(A \circ B) \circ B = A \circ B \tag{2.23}$$

$$(A \bullet B) \bullet B = A \bullet B \tag{2.24}$$

$$\tag{2.25}$$

Quelle que soit la position de B par rapport à l'origine O on a toujours :

$$A \circ B \subseteq A \subseteq A \bullet B \tag{2.26}$$

On note nB l'ensemble défini par :

$$nB = \begin{cases} \{O\} & \text{si } n = 0 \\ (n-1)B \oplus B & \text{si } n > 0 \end{cases} \tag{2.27}$$

On a alors pour $n \geq m \geq 0$:

$$nB \ominus m\check{B} = (n-m)B \tag{2.28}$$

III.2.c Élément structurant à deux pixels

Les ensembles les plus simples utilisés pour les opérations de morphologie mathématique sont (si l'on omet les singletons qui n'introduisent qu'une translation ainsi que l'ensemble vide, élément absorbant) les paires de points distincts, appelées éléments structurants à deux pixels (ÉS2P). En pratique, puisque l'on considère que les changements de position des figures nous indiffèrent et que seule compte leur forme, on se restreint aux cas $\{O, x\}, x \in E \backslash \{O\}$.

III.2.d Notion de connexité dans $\mathbb{Z}^n$

Soit S un sous-ensemble de $\mathbb{Z}^n$. Le couple de vecteurs (y_A, y_B) de $\mathbb{Z}^n \times \mathbb{Z}^n$ est dit S-connexe s'il existe un vecteur $x \in S$ tel que $y_B = y_A + x$ ou $y_B = y_A - x$. Un sous-ensemble G de $\mathbb{Z}^n$ est dit S-connexe si pour tout couple $(y_A, y_B) \in G^2$ il existe $(y_1, ...y_k)$ un k-uplet de points de G tel que $y_1 = y_A$ et $y_k = y_B$ et tel que pour tout $i \in [1, k-1]$ le couple (y_i, y_{i+1}) est S-connexe.
La notion de S-connexité de G renvoie donc à l'idée intuitive qu'il est possible dans G de relier n'importe quel couple de points sans jamais sortir de cet ensemble en utilisant uniquement des translations par les vecteurs de S ou par leur opposé (Fig. 2.2).

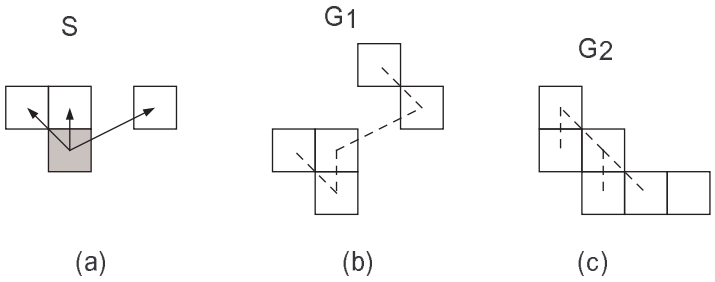


FIG. 2.2 – S-connexité : l'ensemble G_1 est S-connexe, mais pas G_2

Dans le cas bidimensionnel on parle d'ensembles 4-connexes (respectivement 8-connexes) pour désigner les sous-ensembles S_4-connexes (respectivement S_8-connexe) de $\mathbb{Z}^2$ où :

$$S_4 = \{(1,0),(0,1),(-1,0),(0,-1)\} \tag{2.29}$$

$$S_8 = \{(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1)\} \tag{2.30}$$

III.3 Définitions de la transformation Mojette

III.3.a Transformée de Radon

La transformation Mojette correspond à une discrétisation de la transformation de Radon. Cette dernière est définie pour une fonction du plan $f : \mathbb{R}^2 \to \mathbb{R}$ et lorsqu'elle existe comme l'intégrale de f sur la droite perpendiculaire à $\vec{e_r}$ passant par le point $O + \rho\vec{e_r}$ où $\vec{e_r}$ vecteur unitaire formant un angle θ par rapport à l'axe (Ox) (Fig. III.3.b) :

$$\mathcal{R}f(\rho,\theta) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\delta(\rho - x\sin\theta - y\cos\theta)dxdy \tag{2.31}$$

$$\mathcal{R}f(\rho,\theta) = \int_{-\infty}^{+\infty} f(\rho\cos\theta - s\sin\theta, \rho\sin\theta + s\cos\theta)ds \tag{2.32}$$

Si l'on considère les transformées de Fourier à une et deux variables, définies respectivement par :

$$\mathcal{F}g(\nu) = \int g(t)\exp(-2\pi\mathrm{i}\nu t)dt \tag{2.33}$$

$$\mathcal{F}f(u,v) = \int\int f(x,y)\exp(-2\pi\mathrm{i}(ux+vy))dxdy \tag{2.34}$$

on a le théorème suivant (dit de la tranche centrale) :

$$\mathcal{F}f(\rho\cos\theta, \rho\sin\theta) = \mathcal{F}\left\{\mathcal{R}f(\cdot,\theta)\right\}(\rho) \tag{2.35}$$

III.3.b Définition générale de la transformation Mojette

Une transformation Mojette (voir l'article [eJPG98]) consiste à appliquer Radon non plus sur f mais sur la fonction interpolée f_ϕ définie en considérant que ϕ est $\mathcal{L}^2$ par :

$$f_\phi(x,y) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(k,l)\phi(x-k)\phi(y-l) \tag{2.36}$$

On a alors

$$\mathcal{R}f_\phi(\rho,\theta) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(k,l)\mathcal{K}_\phi(k,l,\rho,\theta) \tag{2.37}$$

où $\mathcal{K}_\phi$ est un noyau discret défini par :

$$\mathcal{K}_\phi(k,l,\rho,\theta) = \int\int \phi(x-k)\phi(y-l)\delta(\rho - x\cos\theta - y\sin\theta)dxdy \tag{2.38}$$

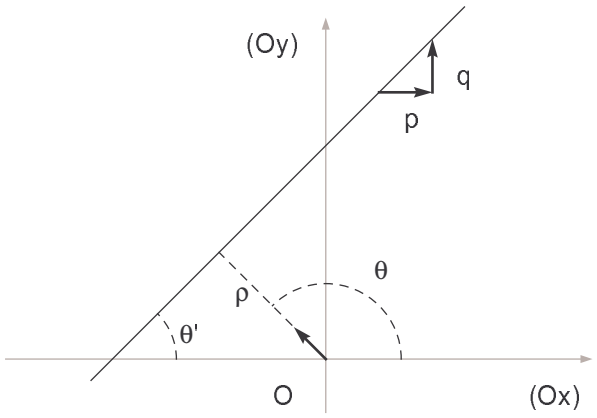


FIG. 2.3 – Droite du plan support de l'intégration pour la transformée de Radon

Par ailleurs on opère en général une rotation de $\pi/2$ en ne considérant les angles $\theta' = \theta - \pi/2$. Pour θ' fixé le résultat de la transformation est appelé projection suivant l'angle θ'. En pratique seuls nous intéressent les angles de projection assurant la superposition d'au moins deux points de $\mathbb{Z}^2$. De fait on se restreint aux angles vérifiant $\tan\theta' = \frac{q}{p}$ où $(p,q) \in \mathbb{Z}^2 \backslash (0,0)$.
Le vecteur (p,q) définit de manière unique la direction de projection. On peut le choisir sans perte de généralité dans le sous-ensemble :

$$\mathcal{D} = \{(1,0)\} \cup \{(p,q) \in \mathbb{Z} \times \mathbb{N}^*, \text{pgcd}(p,q) = 1\} \tag{2.39}$$

La transformée Mojette de noyau $\mathcal{K}_\phi$ de la fonction f est alors :

$$\mathcal{M}_\phi f(\rho,p,q) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(k,l)\mathcal{K}_\phi(k,l,\rho,p,q) \tag{2.40}$$

avec

$$\mathcal{K}_\phi(k,l,\rho,p,q) = \int\int \phi(x-k)\phi(y-l)\delta\left(\rho + x\frac{q}{\epsilon} - y\frac{p}{\epsilon}\right)dxdy \tag{2.41}$$

et $\epsilon = \sqrt{p^2+q^2}$.

III.3.c Transformation Mojette Dirac

Elle correspond au cas particulier où ϕ est une impulsion de Dirac δ. Le noyau devient alors :

$$\mathcal{K}_\delta(k,l,\rho,p,q) = \delta\left(\rho + k\frac{q}{\epsilon} - l\frac{p}{\epsilon}\right) \tag{2.42}$$

et la projection selon (p,q) :

$$\mathcal{M}_\delta f(\rho,p,q) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} f(k,l)\delta\left(\rho + k\frac{q}{\epsilon} - l\frac{p}{\epsilon}\right) \tag{2.43}$$

Les fonctions f considérées ici sont à support borné (Autrement dit, elles sont nulles partout sauf sur un ensemble borné de points de $\mathbb{R}^2$). Comme seules sont ici prises en compte les valeurs de f sur le pavage $\mathbb{Z}^2$ du plan, on peut ne considérer que les fonctions définies sur G sous-ensemble borné de $\mathbb{Z}^2$.

III.3.d Transformation Mojette discrète

Une projection est définie par un nombre fini de pics de Dirac répartis sur une droite selon un pas constant (qui vaut en pratique $1/\epsilon$). L'information de chaque projection peut donc se résumer à un I-uplet d'éléments de $\mathbb{R}$ appelés bins. On peut donc se ramener dans le cas de le transformation Mojette Dirac à une transformée (dite Mojette discrète [Nor97]) où la variable i appartient à un certain segment de $\mathbb{Z}$. Chaque projection correspond donc à un I-uplet de bin (Fig. 2.4). Le bin indicé par i de la projection de vecteur (p,q) vaut :

$$\mathcal{M}f(i,p,q) = \sum_{k\in\mathbb{Z}}\sum_{l\in\mathbb{Z}} f(k,l)\Delta(i+qk-pl) \tag{2.44}$$

$$\tag{2.45}$$

avec le symbole de Kronecker $\Delta(i) = \begin{cases} 1 & \text{si } i = 0 \\ 0 & \text{sinon} \end{cases}$

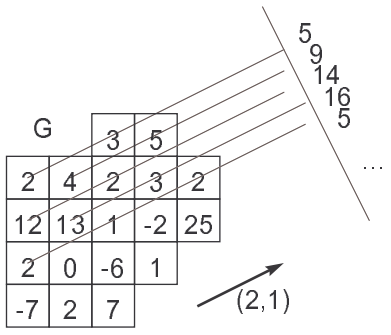


FIG. 2.4 – Projection suivant $(p,q) = (2,1)$ d'une fonction définie sur un ensemble G discret et borné

C'est cette version de la transformation Mojette qui nous interesse le plus, sa nature discrète en faisant un candidat idéal pour le traitement informatique.

III.3.e Transformation Mojette Spline

On peut également définir la transformation Mojette Spline d'ordre n comme la transformation Mojette où la fonction ϕ_n est un certain spline d'ordre n positif, d'intégrale 1, pair et défini comme il suit :

$$\phi_0(x) = \begin{cases} 1 & \text{si } |x| < 1/2 \\ 0 & \text{si } |x| > 1/2 \end{cases} \tag{2.46}$$

$$\phi_n = \phi_{n-1} * \phi_0 \text{ pour } n > 0 \tag{2.47}$$

Pour $n = 0$ le noyau vaut alors :

$$\mathcal{K}_0(k, l, \rho, p, q) = \int_{x=k-1/2}^{k+1/2} \int_{y=l-1/2}^{l+1/2} \delta\left(\rho + x\frac{q}{\epsilon} - y\frac{p}{\epsilon}\right) dxdy \tag{2.48}$$

$$\tag{2.49}$$

On peut prouver qu'à (p, q) fixé la transformée Mojette Spline d'ordre n se calcule simplement comme une convolution de la transformée Mojette Spline d'ordre $n-1$ par le noyau $\mathcal{K}_0$:

$$\mathcal{M}_n f(\cdot, p, q) = \mathcal{M}_{n-1} f(\cdot, p, q) * \mathcal{K}_0(0, 0, \cdot, p, q) \tag{2.50}$$

Comme précédemment on ne s'intéresse qu'à un nombre fini de valeurs de la transformée, correspondant à un échantillonnage de $\mathcal{M}_n f$. On peut donc trouver un équivalent discret de la transformation en redéfinissant le noyau pour i à valeur dans un certain segment de $\mathbb{Z}$:

$$\mathcal{K}_n(k, l, i, p, q) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \phi_n(x-k)\phi_n(y-l)\delta(i + xq - yp)dxdy \tag{2.51}$$

On notera que la formule 2.50 reste valable par discrétisation (c'est à dire en considérant un produit de convolution de $\mathbb{Z}^2$ et non plus de $\mathbb{R}^2$, les noyaux étant définis par 2.51). Dans le cas $n = 0$ on peut alors calculer de manière simple la valeur de $\mathcal{K}_0(0, 0, \cdot, p, q)$:

$$\mathcal{K}_0(0, 0, \cdot, p, q) = \begin{cases} \underbrace{(1, 1, ..., 1)}_{p} * \underbrace{(1, 1, ..., 1)}_{q} & \text{si } pq \text{ est impaire} \\ 1/2 \underbrace{(1, 1, ..., 1)}_{p} * \underbrace{(1, 1, ..., 1)}_{q} * (1, 1) & \text{si } pq \text{ est paire} \end{cases} \tag{2.52}$$

III.3.f Représentation matricielle

Philippé montre dans [Phi98] que la transformée Mojette discrète peut s'écrire comme le produit matriciel $B = \mathfrak{M}P$ où B et P sont respectivement le vecteur des pixels d'une fonction à support fini (par exemple une image) et celui obtenus par concaténation de toutes les projections (en nombre fini si l'on considère que certains angles produisent les mêmes séries de bins). Il existe par ailleurs une matrice $\hat{\mathfrak{M}}$ telle que $P = \hat{\mathfrak{M}}B$. La matrice $\mathfrak{M}$ peut se réorganiser de telle façon que :

$$\begin{pmatrix} B_N \\ B_C \end{pmatrix} = \begin{pmatrix} M_{DN} \\ M_{DC} \end{pmatrix} P$$

avec B_N le vecteur de bins non contraints (qui permettent une reconstruction exacte) et B_C les bins contraints restant (qui fournissent une information redondante par rapport à B_N). La matrice M_{DN} est carré et inversible.

III.4 Inversion de la transformation Mojette

À partir de maintenant, on ne considère que la transformée Mojette discrète appliquée à une fonction $f : \mathbb{Z}^2 \to \mathbb{R}$. Un nombre important de projections de f peut être calculé. Néanmoins la reconstructibilité (critère qui nous intéresse principalement) ne nécessite pas de disposer de toutes ces dernières. De fait, la transformée Mojette peut être utilisée en tant que méthode de description multiple au sens où l'utilisation d'un certains nombre de projections parmi toutes celles générées permet la reconstruction des valeurs de départ (l'ensemble de définition ayant une forme connue).

III.4.a Cas d'un ensemble de définition convexe

On considère qu'une partie G du plan discret $\mathbb{Z}^2$ est convexe s'il existe $\mathfrak{G}$ sous-ensemble convexe de $\mathbb{R}^2$ tel que $G = \mathfrak{G} \cap \mathbb{Z}^2$.
Étant donné un M-uplet de vecteurs de projection $(p_m, q_m), m \in [1, M]$ distincts deux à deux et appartenant à $\mathcal{D}$ (défini dans 2.39), considérons l'ensemble R obtenu par dilatation successive de l'origine O par les ÉS2P $\{O, (p_m, q_m)\}$:

$$R = \bigoplus_{m=1}^{M} \{O, (p_m, q_m)\} \tag{2.53}$$

alors pour toute fonction f définie sur un convexe G les trois propositions ci-dessous sont équivalentes :

i. f est reconstructible à partir des M projections de vecteurs $(p_m, q_m), m \in [1, M]$;

ii. $\forall x \in \mathbb{Z}^2, (R)_x \not\subseteq G$

iii. $G \ominus \check{R} = \emptyset$

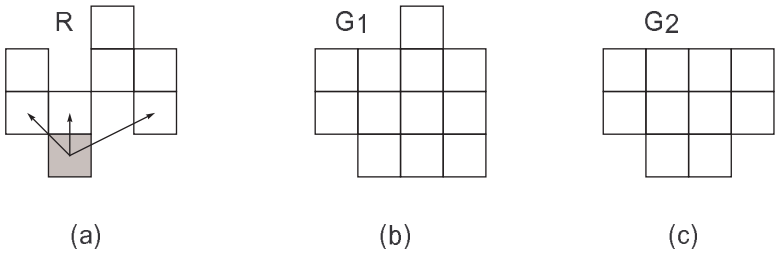


FIG. 2.5 – Ensemble R pour les directions de projection $(-1, 1)$, $(0, 1)$ et $(2, 1)$ (a) et exemples d'ensemble G non-reconstructible (b) et reconstructible (c)

III.4.b Cas d'un ensemble de définition rectangulaire

Dans le cas où G, l'ensemble de définition de f est rectangulaire (Autrement dit, s'il est le produit cartésien de deux intervalles de $\mathbb{Z}$), on peut déduire un critère de reconstructibilité plus simple à partir du précédent. Sans perte de généralité, on considère $G = [0, P-1] \times [0, Q-1]$ comportant $P \times Q$ points.

En imposant encore ici que les vecteurs de projections soient distincts deux à deux et appartiennent à $\mathcal{D}$, on peut montrer que :

i. f est reconstructible à partir des M projections de vecteurs $(p_m, q_m), m \in [1, M]$;

ii. $P \leq \sum_{m=1}^{M} |p_m| \vee Q \leq \sum_{m=1}^{M} |q_m|$

III.4.c Algorithme de reconstruction

La transformation Mojette inverse suit un principe assez simple et relativement rapide (Fig. 2.6). Pour une fonction inconnue f définie sur G (connu) et un ensemble de projections $\mathcal{M}_{p_m,q_m} f, m \in [1, M]$ permettant la reconstruction, on procède de la façon suivante (voir [Nor97] et [eAKePE06]) :

1. On calcule les projections $\mathcal{M}_{p_m,q_m} \mathbf{1}, m \in [1, M]$ de la fonction caractéristique de G (c'est à dire égale à 1 sur G).
2. Pour chaque projection (p_m, q_m), on initialise un vecteur N_m à la valeur $\mathcal{M}_{p_m,q_m} \mathbf{1}$ et un vecteur V_m à $\mathcal{M}_{p_m,q_m} f$. À tout moment, pour la direction (p_m, q_m) et le bin numéro i, $N_m(i)$ correspond au nombre de pixels (points de G) non reconstruits situés sur la droite de projection associée et $V_m(i)$ à la somme cumulée des valeurs de f en ces points.
3. À chaque étape, si f est effectivement reconstructible, il existe pour au moins une projection m un bin i de N_m égal à 1. Il indique que sur la droite de direction (p_m, q_m) correspondant au bin i un seul point de G est encore inconnu. Sa valeur est donc $V_m(i)$.
4. Pour toutes les projections, on décrémente de 1 la valeur $N_m(i_m)$ et de $V_m(i)$ la valeur $V_m(i_m)$ où i_m est le bin correspondant dans chaque projection au pixel précédemment reconstruit.
5. On itère le procédé jusqu'à ce que les vecteurs N_m soit nuls.

III.4.d Reconstructibilité partielle

On sait qu'il n'y a pas reconstructibilité si l'érodé $G \ominus \check{R}$ est non vide. On peut néanmoins démontrer qu'alors il suffit de choisir les valeurs de f sur $G \ominus \check{R}$ pour garantir l'unicité de reconstruction. Notons f_{SC} la fonction reconstruite en supposant qu'elle s'annule sur l'érodé.
Il est ici nécessaire d'introduire la notion de fantôme Mojette. On appelle fantôme de base élémentaire un ÉS2P $\{O, (p, q)\}$ pondéré par 1 en O et par -1 en (p, q). De manière plus correct, il correspond à une fonction $\mathfrak{e}$ définie sur $\mathbb{Z}^2$ nulle partout sauf en O et (p_m, q_m) où elle vaut donc respectivement 1 et -1. Étant donnés les vecteurs de projections $(p_m, q_m), m \in [1, M]$, le fantôme de base associé est la fonction :

$$\mathfrak{f} = \mathfrak{e}_1 * \mathfrak{e}_2 * ... * \mathfrak{e}_M \tag{2.54}$$

où $\mathfrak{e}_m$ est le fantôme de base élémentaire correspondant à $\{O, (p_m, q_m)\}$. On remarquera d'une part que le support de $\mathfrak{f}$ est inclu dans R, d'autre part que les projections de $\mathfrak{f}$ selon chacun des vecteurs (p_m, q_m) sont des vecteurs nuls (Fig. 2.7).

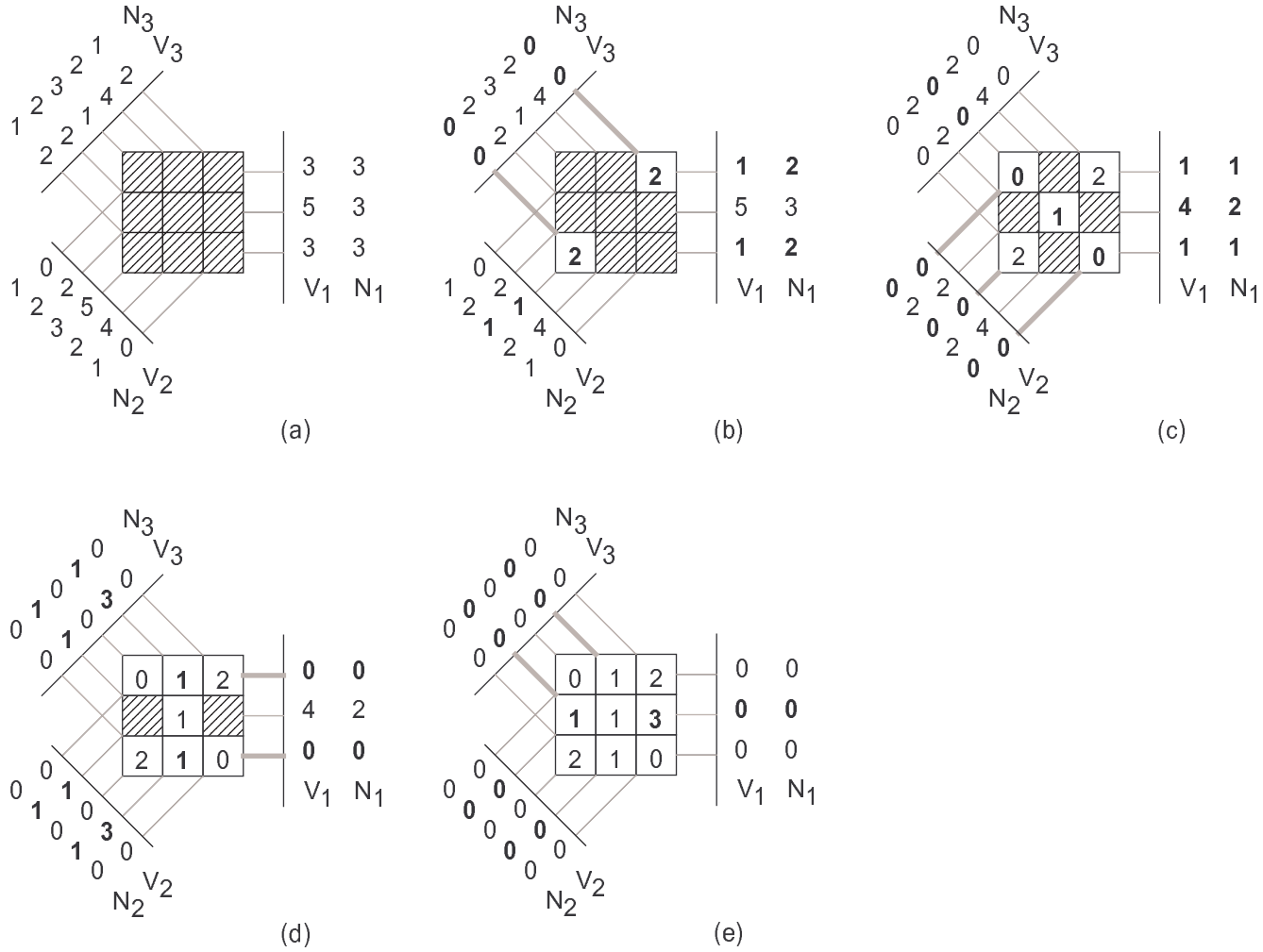


FIG. 2.6 – Exemple de reconstruction avec les projections de vecteur $(-1,1)$, $(1,1)$ et $(1,0)$

Pour $x \in G \ominus \check{R}$ on note par ailleurs $\mathfrak{f}_x$ le fantôme de base translaté par le vecteur x (autrement dit $\mathfrak{f}_x(y) = \mathfrak{f}(y+x)$). On peut alors montrer que f admet une unique décomposition de type :

$$f = f_{SC} + \sum_{x \in G \ominus \check{R}} a_x \mathfrak{f}_x \tag{2.55}$$

avec $a_x \in \mathbb{R}$ pour tout $x \in G \ominus \check{R}$. La reconstrucibilité partielle appliquée à des images a été notamment étudiée par Philippé dans [eJPG97].

III.5 Choix des projections et du support

III.5.a Redondance

Soit une fonction f et pour $M > 0$ entier, un ensemble de vecteurs de projection $\{(p_m, q_m), m \in [1, M]\}$ tel que l'ensemble des projections de f associées permet la reconstruction de cette dernière, alors on peut toujours trouver un vecteur (p_{M+1}, q_{M+1}) tel que les projections associées à n'importe quel sous-ensemble

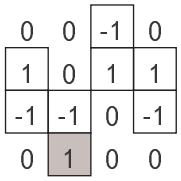


FIG. 2.7 – Fantôme associé aux directions de projection $(-1,1)$, $(0,1)$ et $(2,1)$

à M éléments de $\{(p_m, q_m), m \in [1, M+1]\}$ permettent la reconstruction de f.

Ayant déterminé M projections suffisantes pour la reconstruction, il est donc possible de construire un ensemble de N projections ($N \geq M$) tel que l'utilisation de n'importe quel sous ensemble à M éléments garantit la reconstruction ; soit C_N^M combinaisons possibles. On garantit donc qu'une perte d'au plus $M - N$ projections n'entraîne pas de perte d'information.

On appelle taux de redondance la quantité :

$$Red = \frac{\mathfrak{N}_{bins}}{\mathfrak{N}_{pixels}} - 1 = \frac{\sum_{m=1}^{N} \mathfrak{N}_{bins}(p_m, q_m)}{\mathfrak{N}_{pixels}} - 1 \tag{2.56}$$

où $\mathfrak{N}_{bins}$ et $\mathfrak{N}_{pixels}$ représentent respectivement le nombre total de bins sur toutes les projections et le nombre de pixels (cardinal de G). Le taux de redondance renseigne donc sur la multiplication de l'information présente dans un ensemble de projections donné.

La taille de l'information $\mathfrak{N}_{pixels}$, le nombre de projection minimal M et le niveau de protection $N - M$ étant donnés, le choix du support et des projections doit optimiser le taux de redondance (le plus bas possible).

III.5.b Cas du support rectangulaire

Le nombre de bins pour chaque projection peut être calculé de manière simple dans le cas rectangulaire. Pour une projection selon $(p, q) \in \mathcal{D}$ d'une fonction définie sur le rectangle $G = [0, P-1] \times [0, Q-1]$ il vaut :

$$\mathfrak{N}_{bins}(p, q) = |p|(Q-1) + |q|(P-1) + 1 \tag{2.57}$$

On peut choisir un nombre M de projections de vecteur $(p_m, q_m) \in \mathcal{D}$ suffisamment grand pour garantir $Q \leq \sum_{m=1}^{M} |q_m|$ et donc une reconstruction possible (raisonner sur P est une démarche équivalente).

Ajouter $N - M$ projections supplémentaires n'assure cependant pas a priori que n'importe quel sous-ensemble $A \subset [1, N]$ à M éléments soit suffisant (puisque rien ne permet alors de dire que $Q \leq \sum_{m \in A} |q_m|$). C'est par contre le cas si l'on impose dès le départ que pour tout $m \in [1, N]$ $q_m = q$ constant. En effet on a alors indépendamment du choix de A : $\sum_{m \in A} |q_m| = Mq$. La condition $Q \leq Mq$ étant garantie pour $A = [1, M]$, elle l'est pour n'importe quel A.

Les projections associés à $\{(p_m, q) \in \mathcal{D} \ : m \in [1, N]\}$ suffisent donc en particulier à reconstruire n'importe quel support rectangulaire de dimension $P \times Mq$. Si l'on cherche à minimiser $\mathfrak{N}_{bins}$ à q donné, il convient

de minimiser $\sum_m |p_m|$. On montre que lorsque ce minimum est atteint $\sum_m |p_m| = N^2/4 - \epsilon(N)$ avec $\epsilon(N) = (1-(-1)^N)/8$.

Parrein montre par ailleurs dans [Par01] que la redondance est minimale avec $q = 1$. Elle vaut alors :

$$Red = \frac{N(q(P-1)+1) + (Q-1)\sum_{m=1}^{N} |p_m|}{MP} - 1 = \frac{NP + (M-1)(N^2/4 - \epsilon(N))}{MP} - 1 \tag{2.58}$$

La valeur de $Q = M$ étant fixé par le nombre minimal de projections, le support rectangulaire possède en général une longueur P assez grande. On peut donc considérer :

$$\lim_{P \to +\infty} Red = \frac{N}{M} - 1 = \frac{N-M}{M} \tag{2.59}$$

La redondance est donc presque optimale au sens où elle est aussi proche que voulu de la valeur limite $\frac{N-M}{M}$.

III.5.c Cas du support hexagonal

Un support hexagonal peut être utilisé afin d'obtenir des projections de taille constante (nombre de bins fixe). La figure 2.8 montre un exemple de support (cas $M = 5$ et $N = 7$). Les droites discrètes qui délimitent ce support sont telles que :

$$D_1 = \{(k,l) \in \mathbb{Z}^2 \ : -qk + p_{max} l = cste_1\}$$

$$D_2 = \{(k,l) \in \mathbb{Z}^2 \ : -qk + p_{min} l = cste_2\}$$

Géométriquement il est évident qu'un tel choix fournit bien des projections de taille constante. Néanmoins la redondance pour ce choix de support est légèrement supérieure à celle d'un support rectangulaire. En effet pour certains angles, certains bins situés en début ou à la fin d'une projection correspondent alors à un pixel original. Cette correspondance univoque augmente la redondance sans augmenter le pouvoir de reconstruction par paquet. Parrein montre cependant que la différence devient négligeable lorsque le nombre de pixels croît.

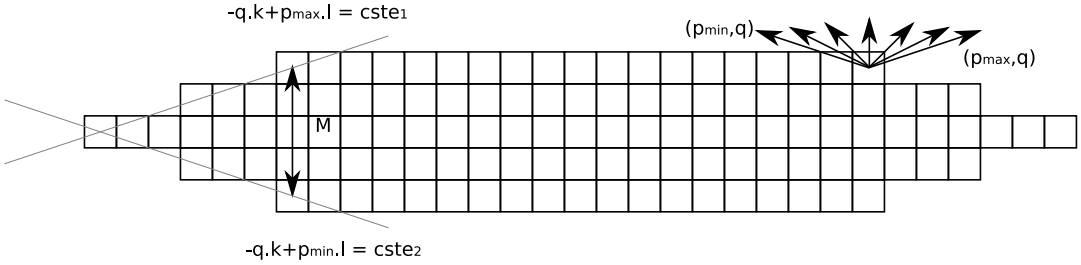


FIG. 2.8 – Exemple de support hexagonal et de projections associées

Ainsi, la transformation Mojette peut être utilisée de manière simple comme mécanisme de description multiple. Parmi les N paquets (descriptions) générées, M sont suffisantes pour reconstituer le message.

III.6 Mojette et codage MD

La transformation Mojette peut ainsi être utilisée pour établir un codage par description muliple. Elle permet en effet de répartir une information en N composantes redondantes tout en garantissant différents niveaux de reconstructions partielles à la réception de M_1, M_2 ... ou M_i de ces composants (avec $N > M_1 > ... > M_i$). Son avantage principal par rapport à des méthodes basées sur des codes optimaux (de Reed Solomon) comme dans [eHWeVP03] réside alors dans sa faible complexité calculatoire.

III.6.a Concaténation de projections Mojette

Nous avons vu que la hauteur Q du support détermine de manière simple le niveau de protection de l'information à coder. En attribuant à chaque calque d'une l'information scalable un support de hauteur correspondant à son importance (plus un calque est important plus Q est faible) puis en projetant chaque support suivant un jeu de projection compatible avec l'ensemble, on peut facilement créer un système PET en concaténant les projections de même angle obtenues pour chaque calque (Fig. 2.9).

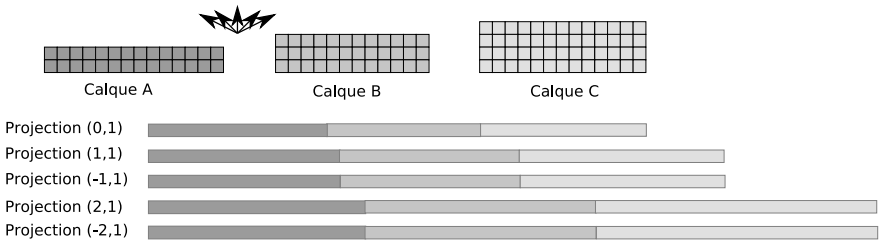


FIG. 2.9 – Modèle PET obtenu par concaténation de projections Mojette calculées pour différents calques d'une source scalable

III.6.b Concaténation de support Mojette

Afin d'obtenir des projections de longueur fixe on peut concaténer les supports correspondant à chaque calque de l'information d'origine. L'utilisation d'extrémités de type hexagonal permet d'obtenir des paquets de longueur constante (Fig. 2.10). Le décodage s'effectue alors à partir de l'extrémité la plus fine selon le procédé décrit précédemment.
Cette opération permet par ailleurs de diminuer le nombre de projections unitaires (un pixel projeté dans un seul bin).

III.7 Conclusion sur la transformation Mojette

La transformation Mojette est donc une transformation géométrique permettant d'introduire de la redondance dans des données. Il est par ailleurs possible de faire varier la redondance apportée sur les différentes parties de l'information transmise. Ces propriétés, ainsi qu'une faible complexité algorithmique, en font un bon candidat pour une utilisation en tant que méthode de codage à description multiple.

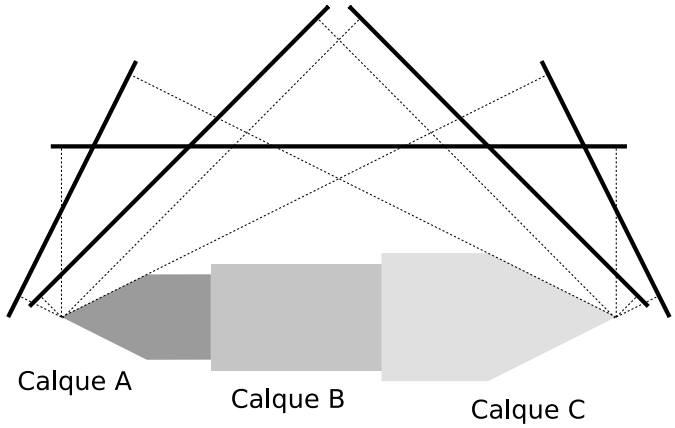


FIG. 2.10 – Modèle PET obtenu par concaténation de différents calques d'une source scalable projetés suivant un ensemble d'angles donné

IV Description multiple et routage

Plusieurs applications du codage à description multiple ont été proposées dans le cadre de la circulation de l'information sur les réseaux ou du stockage. Dans [eKReKLeVB01] la description multiple permet de lutter contre les phénomènes de congestion. En fonction du nombre de descriptions reçues par la destination, la source adapte son codeur. Dans [eHWePCeKS02] les nœuds clients d'un service donné sont susceptibles de fournir eux-mêmes des descriptions des fichiers déjà reçus dans le cas où le serveur deviendrait temporairement inaccessible.

Autre type d'utilisation : séparer les projections pour utiliser différent chemin. Ce mécanisme est connu sous le nom de "path diversity". [eYAeOEeMA05] élabore un modèle probabiliste de transmission de description par deux chemins disjoints et propose une méthode de sélection de chemin optimisée pour les application de type "streaming". Les tests (sur des réseaux filaires) montre une amélioration notable de la qualité moyenne reçue en terme de distorsion. [Apo01] s'intéresse à une problématique similaire. Dans [eBBLeAVePDeMV03], les auteurs comparent pour des réseaux denses l'envoie d'information dans les cas de description unique à chemin unique, description unique à chemins multiples, description multiple UEP et description multiple par quantification scalaire. Ils démontrent d'une part que les performances (distorsion à l'arrivée) sont meilleures en utilisant plusieurs chemins, d'autre part que pour un réseau faible débit ou à contrainte temporelle importante, la description multiple apporte un avantage notable. La transformation Mojette mojette a été utilisé dans un but d'amélioration de qualité de la transmission de l'information (et également le stockage). Voir à ce propos les références [Phi98] et [eBPeNN01].

V Conclusion

Parmi les techniques courantes de codage de l'information, les méthodes à description multiple fournissent une approche nouvelle et intéressante de représentation. Plus particulièrement, les modèles de transmis-

sion par encodage prioritaire (PET) opèrent un découpage équilibré (toutes les descriptions ont le même poids) permettant une reconstruction progressives des sources scalables. La transformation Mojette en est un exemple quasi optimal en terme de compression qui bénéficie par rapport aux codes MDS d'un algorithme de reconstruction simple et performant.

Faire bénéficier les réseaux ad hoc de techniques de description multiple est une optique prometteuse en terme de sécurité du routage. En effet diversifier le parcours de l'information de routage permet de s'assurer que celle-ci, et en particulier parvienne malgré tout à destination, même dans un réseau mobile où les pertes sont importantes.

Chapitre 3

Simulations de protocoles standard sur NS2

Ce chapitre se consacre à l'évaluation des protocoles ad hoc les plus courants. Notre but final étant d'intégrer de nouvelles caractéristiques au routage ad hoc, il convient en effet de comprendre comment ces protocoles fonctionnent et quelles sont leurs forces et leurs faiblesses afin d'être en mesure de définir une amélioration efficace.
Un simulateur, à savoir le logiciel NS2, est utilisé à cette fin. Sont de fait détaillés le principe des simulations réalisées avec NS2, les outils de paramétrage et les méthodes d'analyse des résultats. Il nous faut en outre définir sur quoi portent précisément ces simulations. De très nombreux scénarios peuvent être envisagés dès lors qu'on fait varier les paramètres proposés par NS2. Une fois l'algorithme de routage choisi, ces paramètres peuvent toutefois être classés en trois groupes logiques : le déplacement de noeuds dans l'espace de simulation, le transfert des données et les propriétés de transmission liés aux couches basses des noeuds. La production de résultats nécessite également de définir un certain nombre de critères permettant de juger et de comparer les performances des divers protocoles. Ces critères définis, on procède finalement aux simulations. Les résultats sont analysés afin de déterminer dans quelles situations et sur quels critères chacun des protocoles testés obtient les meilleurs performances.

I Le logiciel NS-2

I.1 De la nécessité de la simulation

Tester et comparer le fonctionnement de protocoles de routage ad hoc pose un problème pratique. Cela nécessiterait de posséder un grand nombre d'unités mobiles (ordinateurs portables). En deçà d'une vingtaine d'unités, le routage est une opération simple et il y a fort à parier que les différents protocoles étudiés ne présenterons que des différences de fonctionnement minimes dans ce contexte. Par ailleurs, il conviendrait dans l'idéal que des utilisateurs manipulent et déplacent ces ordinateurs, sous peine de

devoir se restreindre à n'étudier que des réseaux statiques.

Une manière plus pratique et moins coûteuse consiste à utiliser un simulateur ; autrement dit un logiciel capable de reproduire les principales caractéristiques d'un réseau ad hoc et le faire évoluer virtuellement. Un tel logiciel doit donc être capable de simuler :

- le mouvement d'unités mobiles dans un environnement physique (comportant éventuellement des obstacles) ;
- la capacité à recevoir et émettre des ondes, à créer des interférences ;
- les diverses protocoles liées à la communication en réseau, notamment au niveau :
  - physique (forme des ondes, modulation),
  - liaison de données (protocole de politesse, de gestion de collision, type 802.11, 802.16, etc),
  - routage (la partie qui nous intéresse),
  - transport (gestion des échanges du début à la fin, notamment par utilisation de message d'acquittement et par modulation du trafic),
  - applicatif (création et consommation de flux de données).

Un simulateur adapté doit par ailleurs pré-intégrer les protocoles de routage ad hoc les plus courants, à savoir DSDV, OLSR, DSR, AODV ...

I.2 Les choix de NS2

Le simulateur NS2 ("Network Simulator 2", voir [br09]) a été retenu. Bien qu'a priori non spécialisé pour le contexte ad hoc, il dispose de modules spécifiques pour ce type de réseaux et ses possibilités recouvrent la quasi-totalité des besoins formulés ; il est par ailleurs gratuit et bénéficie d'une large utilisation dans le monde de la recherche en réseau ad hoc. Issu du projet VINT (pour "Virtual Inter Network Testbed") le logiciel NS est, à ses origines, le fruit de la collaboration de quatres laboratoires de l'université de Californie (à savoir USC/ISI, Xerox PARC, LBNL et UCB).

I.3 Le fonctionnement de NS2

NS2 est un simulateur à événements discrets : l'ensemble des changements d'état du système simulé se produise en des instants (sans durée) répartis sur un axe temporel. Par ailleurs, les simulations ne se font pas en temps réel. Le simulateur gère un temps propre sur lequel sont placés les différents événements. Le traitement de ces derniers se fait dans l'ordre de placement sur l'axe temporel de la simulation et permet de séparer le passé (les événements déjà traités), le présent (l'événement en cours de traitement) et le futur (les événements restant à traiter). Les événements traités provoquent l'apparition de nouveaux événements dans le futur, ce qui reproduit les relations de causalité. La dissociation entre temps réel et temps de simulation, si elle a l'inconvénient d'empêcher une visualisation "directe" du fonctionnement du réseau, garantit néanmoins l'indépendance de la simulation vis-à-vis de la rapidité et de la puissance de calcul de l'ordinateur sur lequel elle est effectuée.

L'utilisation de NS2 pour un scénario donné nécessite schématiquement trois étapes :

La création de *fichier de paramétrage.* Ce fichier se présente sous la forme d'un script écrit dans le langage interprété TCL. Il décrit les différents aspects du scénario :

- le nombre et le déplacement des unités mobiles (appellées nœuds),
- le choix des différents protocoles pour chaque couche de chaque nœud,
- le nombre, le type et la durée de divers transferts de données entre ces nœuds,
- etc...

La simulation à proprement parler. Elle reproduit en interne le fonctionnement du scénario décrit précédemment et génère un *fichier de traces*. Ce dernier contient l'information jugée utile, écrite dans un format standardisé.

L'analyse du fichier de trace. Cette analyse peut être effectuée au moyen d'analyseurs syntaxiques ou d'outil de visualisation (comme le logiciel NAM, fournit avec NS2). Le contenu du fichier de trace consiste en une liste d'événements datés se produisant chronologiquement, à raison d'un événement par ligne. Les événements repertoriés dans le fichier de trace sont, à quelques exceptions près, de deux types : d'une part ceux concernant le déplacement des noeuds, d'autre part ceux concernant le parcours des différents types de paquets. C'est ce dernier type d'information qui s'avère le plus intéressant du point de vue de l'analyse du routage. Il permet de connaître, à chaque fois qu'un paquet passe d'un noeud à un autre ou passe - au sein d'un même noeud - d'une couche à une autre, le contenu de ses en-têtes et des informations caractéristiques (identifiants, taille, etc...).

I.4 Le paramétrage

I.4.a L'espace de simulation et la mobilité

Les noeuds se déplacent sur une surface rectangulaire dont on peut régler les dimensions. Le déplacement d'un noeud se fait en précisant par une ligne du type :

```
$simulator at 5.04 "$node-15 setdest 852.264 927.335 7.69478"
```

précisant le temps à partir duquel commence le déplacement, les coordonnées de la destination à atteindre et la vitesse de déplacement. Tout déplacement complexe ne peut donc être qu'une succession de déplacements linéaires.
Par ailleurs, il n'est pas possible dans NS2 de faire disparaître ou apparaître des noeuds de manière simple. On peut bien entendu essayer de simuler ce comportement en déplaçant les noeuds concernés de manière extrêmement rapide vers des zones vides, puis les réinsérer le moment venu. Nous nous sommes cependant abstenu d'utiliser ce procédé.

I.4.b Les transferts de données

Dans NS2, on trouve deux types de flux prédéfinis. Les flux FTP reproduisent le comportement d'un échange de fichier suivant le protocole FTP. Ces flux sont prévus pour utiliser la couche transport TCP.

```
set udp-connection-4 [new Agent/UDP]
set sink-4 [new Agent/Null]
set cbr-flow-4 [new Application/Traffic/CBR]

$simulator attach-agent $node-99 $udp-connection-4
$simulator attach-agent $node-198 $sink-4
$simulator connect $udp-connection-4 $sink-4

$cbr-flow-4 set packetSize_ 512
$cbr-flow-4 set interval_ 0.05
$cbr-flow-4 set random_ 1
$cbr-flow-4 set maxpkts_ 10000
$cbr-flow-4 attach-agent $udp-connection-4

$simulator at 55.5995 "$cbr-flow-4 start"
$simulator at 65.8091 "$cbr-flow-4 stop"
```

TAB. 3.1 – Paramétrage en Tcl d'un flux CBR sur UDP

Les flux CBR (constant bit rate) envoient comme leur nom l'indique des données à un débit constant. Ils utilisent quand à eux la couche transport UDP.
Le paramétrage d'une connexion CBR (voir l'exemple de la table 3.1) se fait par création d'un objet `$udp-connection-4` décrivant la couche UDP de la source (ici le nœud `$node-99`) et relié à un puit `$sink-4` sur la destination (ici `$node-198`). Le flux CBR `$cbr-flow-4` est ensuite attaché à l'agent `$udp-connection-4` et paramétré via un certain nombre d'attributs dont les plus importants sont `packetSize_` (la taille des paquets) et `interval_` (l'intervalle de temps entre deux paquets). Une commande indique au simulateur à quel instant commencer le transfert, une autre à quand y mettre fin.
Le paramétrage d'une connexion FTP (voir l'exemple de code 3.2) se fait par création d'un objet `$tcp-connection-3` décrivant la couche TCP de la source (ici le nœud `$node-0`) et relié à un puit TCP `$tcp-sink-3` sur la destination (ici `$node-3`). Le flux CBR `$ftp-flow-3` est ensuite attaché à l'agent `$tcp-connection-3`. Des commandes similaires au cas CBR indiquent au simulateur le début et la fin du transfert.

I.4.c Les paramètres physique

Les paramètres physique permettent de choisir le type d'antenne et les caractéristiques physiques des ondes électromagnétiques utilisées par les nœuds pour communiquer. Parmi l'ensemble des paramètres on retiendra surtout le paramètre `RXThresh_` correspondant à la puissance minimale devant être reçue

```
set tcp-connection-3 [new Agent/TCP]
set tcp-sink-3 [new Agent/TCPSink]
set ftp-flow-3 [new Application/FTP]

$simulator attach-agent $node-0 $tcp-connection-3
$simulator attach-agent $node-3 $tcp-sink-3
$simulator connect $tcp-connection-3 $tcp-sink-3

$ftp-flow-3 attach-agent $tcp-connection-3

$simulator at 14.3633 "$ftp-flow-3 start"
$simulator at 23.4950 "$ftp-flow-3 stop"
```

Tab. 3.2 – Paramétrage en Tcl d'un flux FTP sur TCP

par un nœud pour qu'il soit capable d'extraire l'information du signal correspondant.
Le modèle de propagation correspond à la façon dont les ondes sont censées se propager physiquement d'un nœud à un autre. Ici le modèle `Propagation/TwoRayGround` prend en compte de l'interférence entre une première onde correspondant au signal reçu de manière directe et une seconde correspondant au signal reçu après réflexion sur le sol.
Modèle de propagation et seuil de puissance de réception permettent de moduler le principal critère qui nous intéresse du point de vue du routage, à savoir la portée des nœuds ; autrement dit, la distance maximale (considérée comme égale pour tous les noeuds) à laquelle les messages sont encore physiquement perçus.

I.5 L'analyse des résultats

Les simulations NS2 produisent des fichiers particuliers, dits de trace, contenant un ensemble d'informations sur le déroulement de la simulation. C'est par l'analyse de ces fichiers que l'on peut comparer les performances des protocoles. Après la description de la syntaxe de ces fichiers de traces, on détaille les critères retenus pour évaluer les protocoles. Enfin, sont présentés les logiciels utilisés pour l'analyse des résultats et le calcul pratique desdits critères.

I.5.a Les fichiers traces

Les fichiers en ".nam" constituent un premier type de fichier de trace. Leur usage requiert le logiciel NAM, détaillé dans le paragraphe I.5.c. Les fichiers en ".tr" constituent à proprement parler les fichiers de trace utiles à l'analyse de la simulation. Chaque ligne d'un de ces fichiers correspond à un événement daté concernant soit un nœud soit un paquet (voir le texte 3.3).

```
d -t 150.000000000 -Hs 20 -Hd 170 -Ni 20 -Nx 603.21 -Ny 35.50 -Nz 0.00 ...
    -Ne -1.000000 -Nl IFQ -Nw END -Ma 13a -Md 14 -Ms 14 -Mt 800 ...
    -Is 58.255 -Id 66.255 -It AODV -Il 44 -If 0 -Ii 0 -Iv 25 ...
    -P aodv -Pt 0x4 -Ph 7 -Pd 90 -Pds 12 -Pl 9.000000 -Pc REPLY
```

TAB. 3.3 – Extrait d'un fichier trace : événement lié à un paquet

Une telle ligne indique :

- l'action correspondant à l'événement : `s` pour l'envoi, `r` pour la réception, `f` pour la retransmission et `d` pour la suppression ;
- sa date (champ `-t`) ;
- des informations sur le(s) nœud(s) réalisant l'événement (champs de type `-H` et `-N`) ;
- des informations sur l'en-tête MAC du paquet (champs de type `-M`) ;
- des informations sur l'en-tête IP du paquet (champs de type `-I`) ;
- des informations sur l'en-tête concernant les couches supérieures (champs de type `-P`).

Le format des lignes est décrit plus en détail en annexe B. C'est l'analyse de ces informations (en particulier les couches IP et supérieures) qui permettent d'évaluer la qualité du routage. Pour cela, on se doit, bien entendu, de définir des critères explicites pouvant synthétiser les points forts et les points faibles des protocoles.

I.5.b Les critères d'évaluation

Afin d'évaluer le comportement des protocoles de routage, 5 critères d'évaluation ont été définis : le taux de paquets délivrés, le délai de transmission, la gigue, le coût du routage et la concentration de l'activité. Ces divers critères ont pour d'évaluer les différents aspects du fonctionnement des protocoles tels que l'efficacité des transferts, leur rapidité et leur stabilité, la quantité d'information supplémentaire nécessaire au bon fonctionnement, ou bien encore la capacité des protocoles à répartir plus équitablement le routage entre les nœuds. Ces 5 critères sont alors calculés comme suit :

le taux de paquet délivré : il correspond au rapport entre le nombre de paquets de données reçus par les destinations et le nombre de ceux qui ont été émis par les sources (les messages d'acquittement dans le cadre FTP ne sont pas pris en comptes) ;

$$\texttt{TauxDePaquetsDelivres} = \frac{\left|\mathcal{P}^{\rightarrow AGT}_{donnees}\right|}{\left|\mathcal{P}^{AGT\rightarrow}_{donnes}\right|}$$

le délai de transmission moyen : il correspond au temps moyen mis par les paquets reçus pour passer des couches applicatives de la source à celles de la destination ;

$$\texttt{Delai} = \frac{1}{\left|\mathcal{P}^{\rightarrow AGT}_{donnees}\right|} \sum_{P \in \mathcal{P}^{\rightarrow AGT}_{donnees}} \delta t_P^{AGT}$$

la gigue : elle correspond à la stabilité du délai ; en pratique on la calcul comme la moyenne de la différence dans les délais de transmission de deux paquets successivement reçus appartenant à un même flux de données ;

$$\texttt{Gigue} = \frac{1}{\left|\mathcal{P}^{\rightarrow AGT}_{donnees}\right|} \sum_{\substack{\{P_i, P_{i+1}\} \subset \mathcal{P}^{\rightarrow AGT}_{donnees} \\ P_i \prec P_{i+1}}} \left| \delta t^{AGT}_{P_{i+1}} - \delta t^{AGT}_{P_i} \right|$$

Ainsi une gigue nulle signifierait que dans chaque flux de donnée tous les paquets mettent exactement autant de temps pour parvenir à destination.

le coût de routage : il mesure la quantité moyenne d'information de routage nécessaire pour chaque paquet de données reçu ; il est égal au rapport entre le nombre d'envois de paquets de routage et le nombre de paquets de données reçus à destination ;

$$\texttt{CoutDuRoutage} = \frac{\left|\mathcal{P}^{RTR\rightarrow}_{routage} \cup \mathcal{P}^{-RTR\rightarrow}_{routage}\right|}{\left|\mathcal{P}^{\rightarrow AGT}_{donnees}\right|}$$

la concentration de l'activité : elle sert à évaluer si la contribution des noeuds au bon fonctionnement du réseau ad hoc est bien répartie, ou si à l'inverse, seul un nombre limité de noeuds contribue à garantir ce fonctionnement (ceux-ci dépensent alors plus d'énergie que les autres). Soit $\Phi_{act}(V) = |\mathcal{P}^{V.RTR\rightarrow}| + |\mathcal{P}^{-V.RTR\rightarrow}|$ le nombre de paquet retransmis ou envoyé par V et soit $\mu_{act} = \frac{1}{|\mathcal{V}|} \sum_{V \in \mathcal{V}} \Phi_{act}(V)$ la moyenne de Φ_{act}. La concentration de l'activité est :

$$\texttt{Concentration} = \frac{1}{\mu_{act}} \left(\frac{1}{|\mathcal{V}|} \sum_{V \in \mathcal{V}} |\Phi_{act}(V) - \mu_{act}| \right)$$

I.5.c Outils d'analyse des résultats

Deux logiciels ont été utilisés afin d'analyser les fichiers traces :

Le logiciel NAM permet de visualiser le déplacement des nœud et le parcours des paquets dans le réseau.

Un analyseur syntaxique a été développé en java afin d'extraire des fichier de traces les 5 critères de performances précédemment décrites.

II Principe des tests

Afin d'évaluer le comportement général de plusieurs protocoles de routage, des simulations ont été effectuées en faisant varier divers paramètres de simulation (dont les résultats ont été publiés dans [eECeHIeJPG06]). Un grand nombre de scénario a ainsi été généré. Ceux-ci ont notamment pour but d'analyser comment se comporte chacun des protocoles pour des transfert CBR ou FTP lorsque l'on fait varier la densité du réseau, le nombre de transfert ou la vitesse des noeuds. Ces tests sont par ailleurs réalisés suivant deux modèles de mobilité afin d'évaluer si le mode de déplacement des noeuds - en dehors de leur vitesse moyenne - a un impact réel sur les résultats.

II.1 Les Modèles de mobilité

Pour chaque scénario de test, un ensemble de noeuds se déplacent aléatoirement sur la surface de simulation. Il convient toutefois de préciser ce qu'on entend par déplacement aléatoire. Autrement dit, il faut décrire précisément quel processus aléatoire décrit le déplacement de chaque noeud. Dans les scénarios couramment utilisés chaque noeud suit une même loi. Dans [eNSeAH03], les auteurs indiquent que le modèle de mobilité peut grandement influencer les performances des protocoles.
Un grand nombre de modèles a été proposé. Chacun a des spécificités propres, liées à un comportement et des caractéristiques recherchés. Il peut s'agir de :

- *dépendance spatiale* : un noeud se déplace en groupe ou influence le déplacement des autres noeuds à proximité ;
- *dépendance temporelle* : la vitesse d'un noeud à un moment donné est corrélée avec sa vitesse dans le passé ;
- *restriction géographique* : un noeud ne peut franchir une zone ou doit suivre un axe.

La plupart des modèles de déplacement utilise à la fois plusieurs de ces principes. L'article [eJBeVD02] et le livre [eAH04] répertorient les plus courants. Nous en donnons ici une courte description.

II.1.a Modèle Random Waypoint

Dans le *Random Waypoint* un noeud alterne les périodes de mouvement et d'immobilité. Il commence par attendre une certaine période T_{pause} (le temps de pause), suite à laquelle il choisit une nouvelle position $\vec{X}'$ sur la surface de simulation (la sélection étant uniforme). Une vitesse V est tirée uniformément dans un intervalle $[V_{min}, V_{max}]$. Le noeud se déplace alors à vitesse V de sa position actuelle $\vec{X}$ jusqu'à la nouvelle position $\vec{X}'$. À l'arrivée, il effectue à nouveau une pause, et ainsi de suite. Le gros inconvénient d'un tel modèle est qu'au fil du temps, il a tendance à concentrer les paquets vers le centre de l'aire de simulation.

II.1.b Modèle Random Direction

Les modèles dits de *Random Direction* fonctionnent à contrario sur l'idée que c'est la direction θ (modélisée par un angle) et non la position de destination qui constitue la caractéristique principale du déplacement. Une des variantes propose de ne mettre fin à ce déplacement qu'après avoir atteint le bord de la surface de simulation. À partir de là le noeud marque une pause pour une durée T_{pause} puis tire de manière uniforme une nouvelle direction θ' dans l'ensemble des angles acceptables (ceux dirigés vers l'intérieur de la surface). Il repart alors en ligne droite à une vitesse $V \in [V_{min}, V_{max}]$ dans l'angle choisi jusqu'à atteindre à nouveau le bord de la surface. Dans la variante dite *Modified Random Direction*, le noeud ne parcourt pas nécessairement le terrain jusqu'à en atteindre le bord. En même temps que l'angle et la vitesse, une distance D à parcourir est tirée uniformément dans $[D_{min}, D_{max}]$. Une fois la distance D parcourue à vitesse V dans la direction θ, le noeud effectue un temps d'arrêt puis repart avec de nouvelle valeurs de D, V et θ. En cas de collision avec le bord, le noeud rebondit et ne termine son parcours que

lorsque D est effectivement parcourue.

II.1.c Modèle Proba Walk

Dans le *Proba Walk* (proposé dans [Chi98]), le temps est divisé en courtes périodes T durant lesquelles les coordonées X_1 et X_2 vont, indépendamment l'un de l'autre, subir un incrément positif, négatif ou nul. À cette fin, chaque noeud possède deux variables d'état s_1 et s_2 à valeur dans $\{0, 1, 2\}$. Si à l'instant t la valeur de s_1 est :

- 0, l'abscisse X_1 du noeud à $t + T$ demeure inchangée ;
- 1, l'abscisse X_1 du noeud à $t + T$ varie d'une quantité δ dans le sens négatif (autrement dit le noeud voit son abscisse diminuer) ;
- 2, l'abscisse X_1 du noeud à $t + T$ varie d'une quantité δ dans le sens positif (autrement dit le noeud voit son abscisse augmenter).

Il en va de même pour l'ordonnée X_2, liée à la variable s_2. Une matrice de probabilité P de dimension 3×3 décide à chaque étape si les variables s_1 et s_2 changent de valeur :

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} \\ P_{10} & P_{11} & P_{12} \\ P_{20} & P_{21} & P_{22} \end{pmatrix}$$

Ainsi $P_{i,j}$ indique la probabilité pour s_k, sachant qu'elle se trouve actuellement dans l'état i ($\in \{0, 1, 2\}$), de passer dans l'état de j ($\in \{0, 1, 2\}$).

II.1.d Modèle Gauss-Markov

Le modèle de *Gauss-Markov* découpe également le temps en périodes égales T durant lesquelles la vitesse est constante. La vitesse V et la direction θ sur une période donnée dépendent des vitesse et direction V' et θ' de la période précédente suivant les relations :

$$V = \max(0, \min(V_{max}, V' + W_V))$$

$$\theta = \theta' + W_\theta$$

où W_V et W_θ sont deux variables aléatoires gaussiennes centrées indépendantes (d'écarts type respectifs σ_V et σ_θ) et V_{max} est une vitesse maximale. Le couple (V, θ) est donc effectivement un processus de type Gauss-Markov :

- La réalisation du processus à l'instant précédent apporte au moins autant d'information que l'ensemble du passé (principe des processus markoviens).
- Les variables aléatoires correspondant au processus en des instants donnés sont gaussiennes et toute combinaison linéaire de celles-ci est également gaussienne (principe des processus gaussiens).

Plus σ_V est grand plus la vitesse peut subir des variations rapides ; plus σ_θ est grand plus la trajectoire est susceptible de tourner rapidement.

II.1.e Modèle Random Walk

Dans ce modèle, inspirée du mouvement brownien, le nœud sélectionne une vitesse dans $[V_{min}, V_{max}]$ et un angle dans $[0, 2\pi]$ indépendamment des valeurs précédentes. La distance parcourue peut être fixe ou aléatoirement tirée dans $[D_{min}, D_{max}]$ ou définie par une durée déterminée.

II.1.f Modèle Manhattan

Le modèle *Manhattan* propose un déplacement des noeuds mimant celui de véhicules dans les rues d'une ville. Il implique généralement l'existence d'une grille carré telle que les noeuds ne sont autorisés à se déplacer que sur les axes définis par celle-ci. Arrivé à un carrefour, chaque mobile décide aléatoirement quel doit être le prochain segment suivi. Il tourne avec une probabilité p_θ (indifférement à droite ou à gauche) et maintient sa direction avec une probabilité $(1 - p_\theta)$. À chaque fois qu'une distance D est parcourue, sa vitesse est susceptible de changer avec une probabilité p_V. Dans ce cas la nouvelle vitesse est sélectionnée selon une loi normale de paramètres V_{moy} et σ_V.

II.1.g Conclusion sur les modèles

Les différents modèles décrivent diverses processus aléatoires représentant des façon distincte d'envisager le déplacement sur un plan. On trouve en annexe C le résultat de tests effectués afin d'évaluer si certains de ces modèles favorisent ou non certains protocoles de routage. Par la suite, seuls les modèles *Modified Random Direction* et *Proba Walk* sont utilisés pour confronter les performances des protocoles.

II.2 Les protocoles et les paramètres des tests

De tous les protocoles de routage ad hoc, les plus connus et couramment utilisés sont DSR [eDAM96], AODV [eEMBR04] et OLSR [ePMeTCeALeAQeLV01]. Nous nous proposons donc d'effectuer de les mettre en application dans un certain nombre de scénarios. Les diverses scénarios doivent permettre de dégager l'influence sur chacun de paramètres tels que la densité du réseau, le nombre de connexion et la vitesse des nœuds. Pour chaque variantes, 3 scénarios sont aléatoirement générés et leur résultats moyennés. La liste des paramètres constants et variables utilisés pour ces simulations est donnée dans le tableau 3.4.
Nous nous sommes restreint dans les présente simulations à l'usage de deux modéles de mobilités. À savoir le *Modified Random Direction* (dont les paramètres correspondants sont donnés dans le tableau 3.5) et le *Proba Walk* (voir le tableau 3.6).

III Résultat des tests

Trois séries de tests sont effectuées pour chaque protocole. Elles ont pour but de mettre en évidence le comportement de ces derniers vis à vis des variations de :
– charge (par augmentation du nombre de transfert) ;

Paramètres du scénario	
Nombre de nœuds n_{tot}	125,150,175,200,225,250,275
Taille de l'aire de simulation	$1000m \times 1000m$
Durée de simulation	180 s
Nombre de transferts	10,20,30,40,50,60
Durée des transferts	entre 40 et 80 s pour CBR, entre 5 et 20 s pour FTP
Débit de chaque transfert λ	16 paquets de $512o$ par secondes $= 8ko/s$ pour CBR,
Nombre de scénarios moyennées	3
Paramètres physiques	
Protocol MAC	IEEE 802.11
Modèle de reflexion	Two-ray ground
Portée des nœuds r	100 m

Tab. 3.4 – Paramètres utilisés afin de comparer les protocoles OLSR, DSR et AODV

Distance parcourue D	tirage entre 50 et 100 m
Vitesse de déplacement V	tirage entre 2 et 15 m/s
Temps de pause T_{pause}	0, 5, 10, 15, 20 ou 25 s

Tab. 3.5 – Paramètres utilisés pour le modèle Modified Random Direction

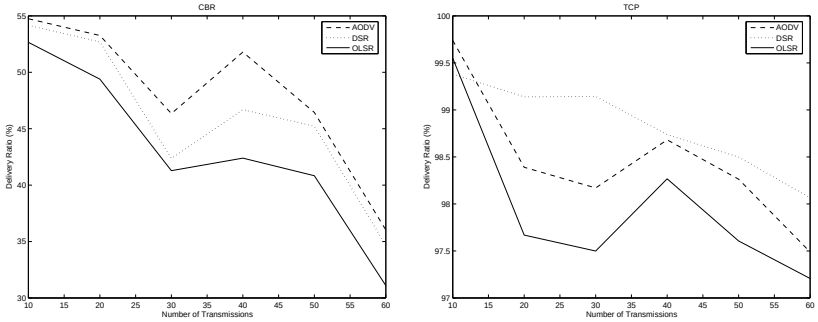
Fréquence de mise à jour T	50 s
Distance parcouru δ	50 m
Matrice de probabilité P	$P = \begin{pmatrix} 0.0 & 0.5 & 0.5 \\ 0.3 & 0.7 & 0.0 \\ 0.3 & 0.0 & 0.7 \end{pmatrix}$

Tab. 3.6 – Paramètres utilisés pour le modèle Proba Walk

– mobilité (par diminution du temps de pause) ;
– densité (par augmentation du nombre de nœud sur une surface constante).
Chaque série est par ailleurs découpée en deux sous-séries correspondant à des envois FTP sur TCP d'une part et CBR sur UDP d'autre part.

III.1 Impact de la charge

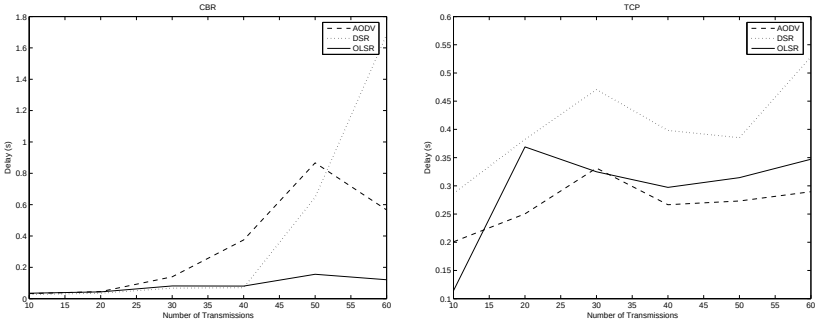
Le Proba Walk a ici été utilisé sur 200 nœuds. On voit dans les figures 3.1 (a) et (b) qu'en terme de paquets délivrés AODV et DSR semblent se maintenir au dessus d'OLSR quelque soit la charge du réseau (de 3 à 5% d'écart avec OLSR). Si AODV sort vainqueur pour les transfert CBR, c'est en revanche DSR qui l'emporte pour le FTP.



(a) Flux CBR (b) Flux FTP

FIG. 3.1 – Taux de paquet délivré, en fonction de la charge

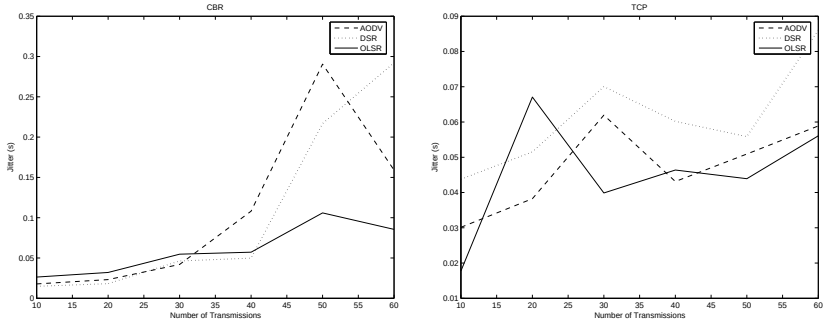
Les figures 3.2 (a) et 3.3 (a) montrent qu'en terme de délai et de gigue, OLSR est plus efficace que ses deux concurrents à transmettre des flux CBR lorsque la charge dépasse 30 connections. Dans la version FTP (figures 3.2 (b) et 3.3 (b)), la charge semble cependant avoir moins d'impact et l'on observe par ailleurs une nette amélioration des performance d'AODV.



(a) Flux CBR (b) Flux FTP

FIG. 3.2 – Délai, en fonction de la charge

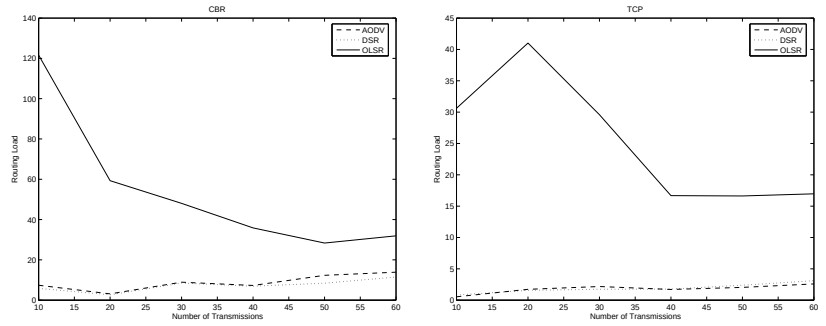
Les figures 3.4 (a) et (b) présente à l'inverse peu de différence : le coût du routage, très important pour OLSR vers les petites valeurs de charge, tend à diminuer avec son augmentation. Il reste cependant bien



(a) Flux CBR (b) Flux FTP

FIG. 3.3 – Gigue, en fonction de la charge

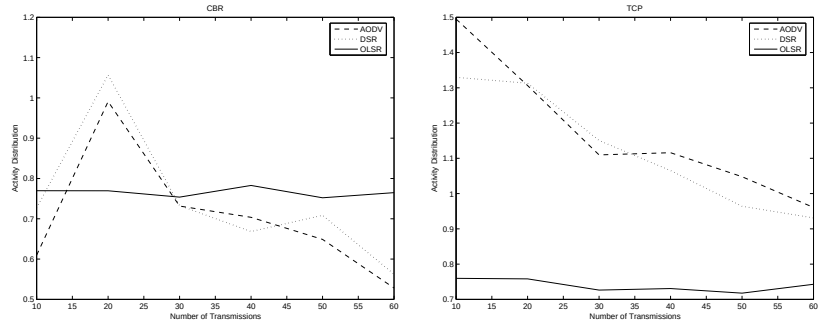
supérieur à ceux d'AODV et de DSR (entre 4 et 12 fois moins).



(a) Flux CBR (b) Flux FTP

FIG. 3.4 – Coût du routage, en fonction de la charge

La concentration de l'activité tend par nature à diminuer lorsque le nombre de transfert augmente (Figures 3.5 (a) et (b)). Elle reste cependant très stable pour OLSR et particulièrement importante pour AODV et DSR dans le cas FTP.

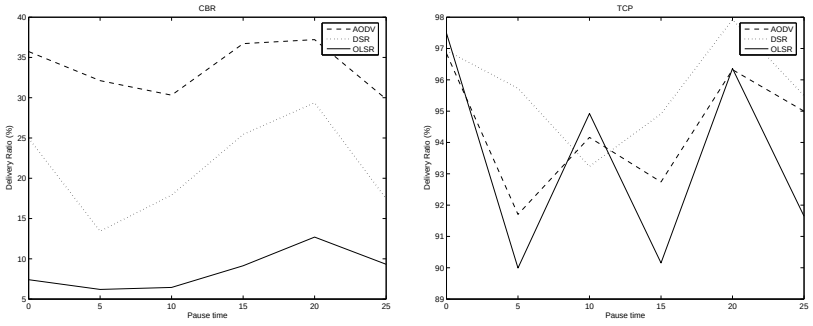


(a) Flux CBR (b) Flux FTP

FIG. 3.5 – Concentration de l'activité, en fonction de la charge

III.2 Impact de la mobilité

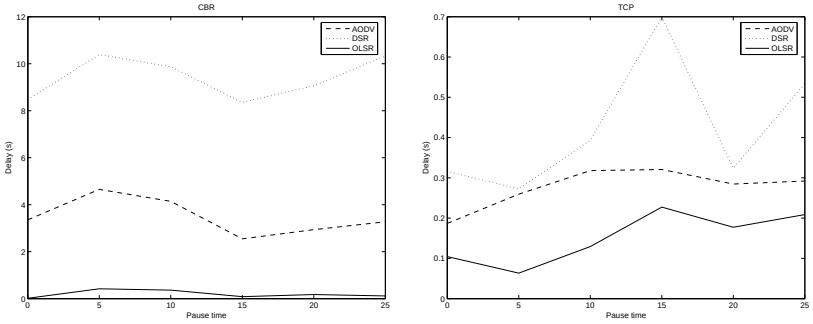
Afin de jouer sur la mobilité globale des noeuds, on utilise le Modified Random Direction tout en faisant varier le temps de pause. Ainsi plus celui-ci est long plus les noeuds sont fréquemment à l'arrêt. Les figures de 3.6 montrent des résultats similaires à celles de 3.1 concernant l'ordre des protocoles. Néanmoins on n'observe pas vraiment de chute globale du taux de paquets délivré avec l'augmentation du temps de pause.



(a) Flux CBR (b) Flux FTP

FIG. 3.6 – Taux de paquet délivré, en fonction de la mobilité

De même le délais et la gigue (figures 3.7 et 3.8) sont relativement invariants par rapport à la mobilité avec un positionnement des protocoles constant (OLSR étant le plus rapide avec une variation faible des délais, DSR étant à l'inverse le plus lent et le moins constant).



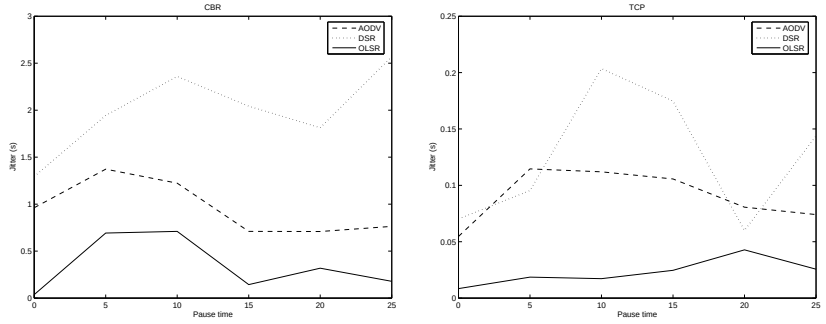
(a) Flux CBR (b) Flux FTP

FIG. 3.7 – Délai, en fonction de la mobilité

Le coût du routage est défavorable à OLSR et sensiblement équivalent pour DSR et AODV (figures 3.9).

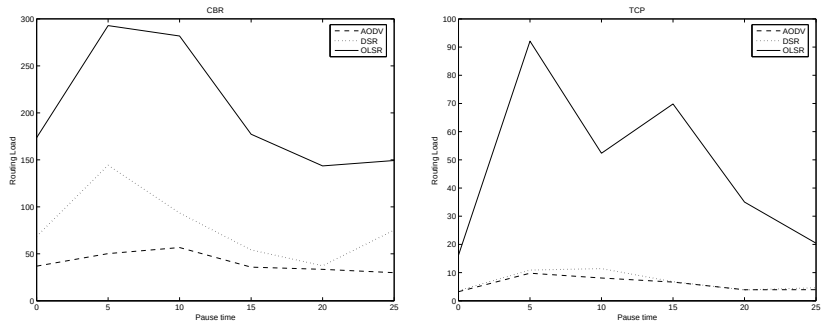
III.3 Impact de la densité du réseau

On cherche dans cette section à établir l'impact de la densité du réseau sur le comportement relatif des protocoles (le modèle de mobilité étant encore Modified Random Direction). Pour ce faire on augmente progressivement le nombre de noeud sur une surface de simulation constante. Le nombre moyen de voisins,



(a) Flux CBR (b) Flux FTP

FIG. 3.8 – Gigue, en fonction de la mobilité



(a) Flux CBR (b) Flux FTP

FIG. 3.9 – Coût du routage, en fonction de la mobilité

qui peut être considéré comme un bon estimateur de densité du réseau est donné par :

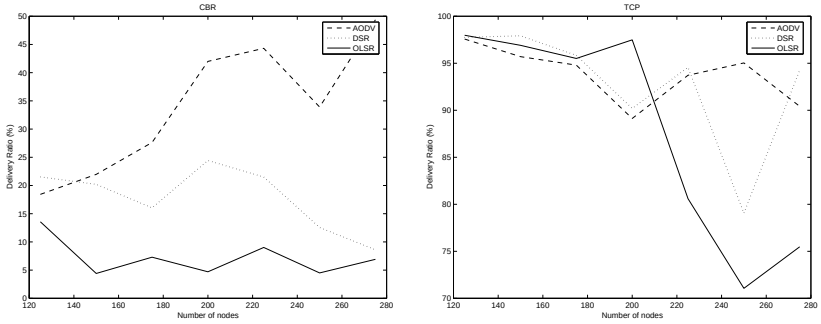
$$n_{vois} = \frac{\pi r^2}{c^2} n_{tot} - 1$$

où n_{tot} est le nombre de noeuds, c la longueur du coté de l'espace de simulation et r la portée de chaque noeud. Dans le cas CBR le taux de paquets délivrés reste faible pour OLSR, diminue sensiblement pour DSR et augmente pour AODV (figure 3.10 (a)). Dans le cas FTP on oberve une différentiation croissante avec l'augmentation de la densité : OLSR n'y résiste pas alors que DSR et surtout AODV semblent moins affectés (voir figure 3.10 (b)).

Le délai est peu affecté par l'augmentation de la taille du réseau, en particulier dans le cas d'OLSR. Ce dernier reste par ailleurs devant AODV ou DSR.

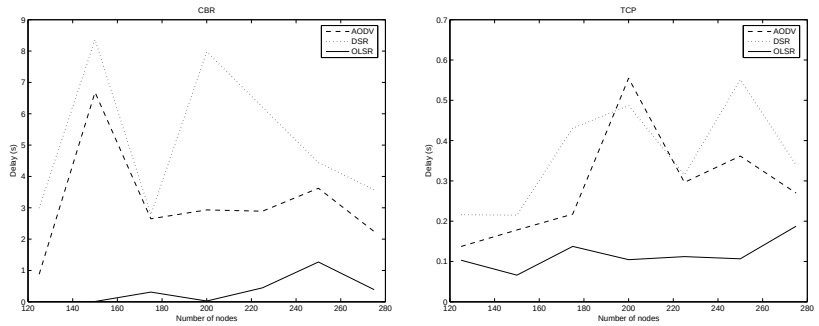
La gigue tend à croître avec l'augmentation de la taille du réseau pour tous les protocoles. L'ordre global reste OLSR en premier, puis AODV et enfin DSR (voir figures 3.12).

La concentration de l'activité est fortement affectée par une variation de taille du réseau pour les protocoles AODV et DSR, et, à l'inverse, bien peu pour OLSR. En effet le fonctionnement proactif de OLSR implique une participation commune de tous les noeuds, que ceux-ci soient ou non impliqués dans un transfert de données. Dans le cas des deux protocoles réactifs, l'écart de participation entre différents type de noeuds est d'autant plus visible qu'il y a peu de noeuds (voir figures 3.13).



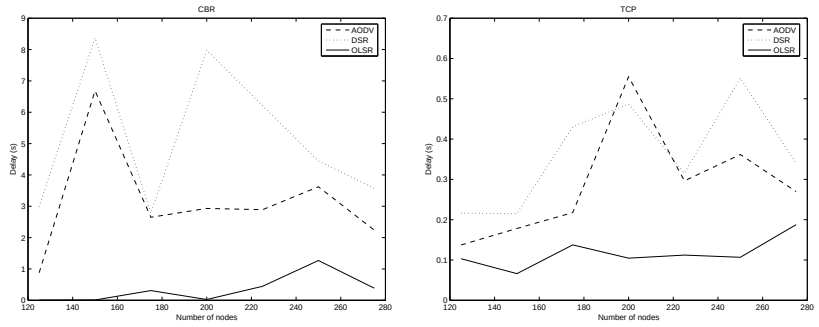
(a) Flux CBR (b) Flux FTP

FIG. 3.10 – Taux de paquet délivré, en fonction de la taille du réseau



(a) Flux CBR (b) Flux FTP

FIG. 3.11 – Délai, en fonction de la taille du réseau

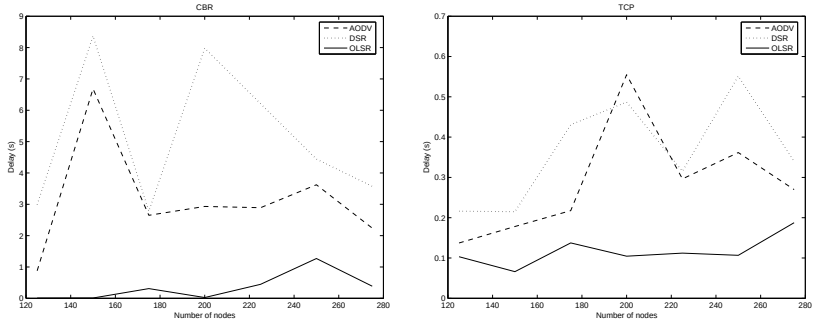


(a) Flux CBR (b) Flux FTP

FIG. 3.12 – Gigue, en fonction de la taille du réseau

IV Analyse

Comme on pouvait s'y attendre, une différence de comportement flagrante apparaît entre, d'une part OLSR, d'autre part DSR et AODV. Le comportement proactif d'OLSR favorise la réactivité et donc offre un délai de transmission moindre que celui de ses concurrents. En revanche OLSR est plus sensible à la taille du réseau : l'augmentation du nombre de nœuds implique une plus forte transmission de paquets de contrôles, même s'il y a peu de transferts de données. Par ailleurs, DSR se distingue de AODV en fournissant un meilleur taux de paquets délivrés dans le cas de FTP. Il semble que l'utilisation d'une procédure



(a) Flux CBR (b) Flux FTP

FIG. 3.13 – Concentration de l'activité, en fonction de la taille du réseau

supplémentaire pour trouver une route de la destination à la source dans DSR favorise la réception des messages d'acquittement de TCP (alors que AODV se contente d'utiliser la route précédement trouvée en sens inverse).

V Conclusion

Nous avons dans ce chapitre introduit d'une part les outils choisis pour la simulation des protocoles ad hoc ; à savoir le logiciel NS2 ainsi que divers outils d'analyse de traces. Nous avons par ailleurs comparé les trois principaux protocoles de routages : OLSR, DSR et AODV. À cette fin, il a été nécessaire de définir un cadre de simulation en choisissant :

- les différents paramètres à faire varier pour évaluer leur impact sur chacun des protocoles (densité, charge et mobilité) ;
- un ensemble de critères susceptibles de rendre compte des performances des protocoles (taux de paquets délivrés, délai, gigue, coût du routage et concentration de l'activité).

Les tests effectués suivant ces modalités ont permis de dégager des comportements propres à chaque protocole.

Chapitre 4

Descriptions multiples sur chemins multiples

I Introduction

Un des problèmes majeurs des réseaux ad hoc réside dans leur incapacité à garantir l'existence de routes entre couples de nœuds communicants. Dans le monde ad hoc, derrière chaque routeur se tient un utilisateur à la fois mobile et susceptible de quitter le réseau quand bon lui semble. Une route entre deux nœuds peut dès lors disparaître à tout moment. Intuitivement, utiliser deux routes semble offrir une perspective d'amélioration. Sous réserve qu'elles ne soient pas liées, il est en effet moins probable que deux routes disparaissent en même temps. Mais qu'envoyer sur chacune de ces routes ? Les paquets originaux en deux exemplaires ? Faut-il plutôt découper le flux original en deux sous-flux ? Doit-il subir une transformation afin de s'adapter aux multiples trajets ? Nous allons dans ce chapitre décrire une approche combinant l'exploitation de chemins multiples avec des techniques de codage de l'information. Le but recherché est de diminuer la dépendance entre les transferts et l'instabilité propre aux réseaux ad hoc. On espère ainsi améliorer la qualité de service en rendant les variations de topologies du réseau les plus transparentes possibles pour les couches applicatives.

Afin d'apporter du crédit à une telle stratégie, nous nous proposons de revenir sur la logique du transfert multichemins et d'y insérer la problématique de leur exploitation. Autrement dit, en supposant plusieurs routes déterminées par une procédure a priori non précisée, il convient de définir comment utiliser de ces dernières. En particulier on cherche à combiner ces routes avec des méthodes de codage de l'information afin d'améliorer la transmission. Dans un second temps, on s'attache plus en détails aux méthodes de choix des routes, en montrant que les propositions connues ne répondent que partiellement à l'objectif souhaité. Ceci nous conduit à proposer une nouvelle méthode de sélection de routes qui réalise un meilleur compromis entre la réalité pratique et les exigences attendues en terme de fiabilité de la transmission. On cherche ensuite à étudier plus finement comment les paramètres de codage de l'information et sa

répartition sur les routes fournies par l'algorithme proposé influencent la qualité globale des transferts. Enfin, des tests sont effectués afin d'évaluer l'intérêt de cette approche, de la comparer au monochemin et d'évaluer de manière appliquée l'impact des paramètres choisis.

II L'intérêt des transfert multichemins

Il existe plusieurs régles de routage possibles dans un contexte où l'on dispose de plusieurs routes reliant un même couple source-destination. Les diverses propositions de protocoles multichemins (comme [eSVKeSKT04] et [eSRD01]) ne précisent généralement pas comment les données doivent être réparties. Autrement dit, leur but est simplement d'offrir une liberté supplémentaire aux couches supérieures. Certaines propositions comme [eMG01] proposent une simple redistribution des paquets sur les routes trouvées (les paquets sont répartie afin que chaque route en prenne en charge autant qu'une autre).
Toutefois d'autres (comme l'article [eSD99] ou, dans un cadre non nécessairement ad hoc, l'article [eWDG94]) impliquent clairement une utilisation successive des routes. Le changement de trajet des paquets est alors supposé s'effectuer lorsque la route en cours d'utilisation rencontre des difficultés. Ce fonctionnement n'est cependant multichemins qu'en apparence : à un moment donné une seule route est réellement utilisée entre les nœuds communicants. En réalité, nous sommes en fait dans un cadre monochemin, où la restauration des routes est facilitée par une anticipation des pertes de connection. L'avantage principal consiste ici en un gain de temps : en cas de problème, des routes de substitutions sont déjà connues, sous reserve bien entendu que ces mêmes routes de substitutions soient par contre restées valides.
À l'inverse, l'emploi simultané des routes donne tout son sens à la notion de multiroutes. L'idée est alors de profiter de cette diversité pour répartir le flux d'information - éventuellement transformé - sur les chemins à disposition.

II.1 Problèmes et objectif

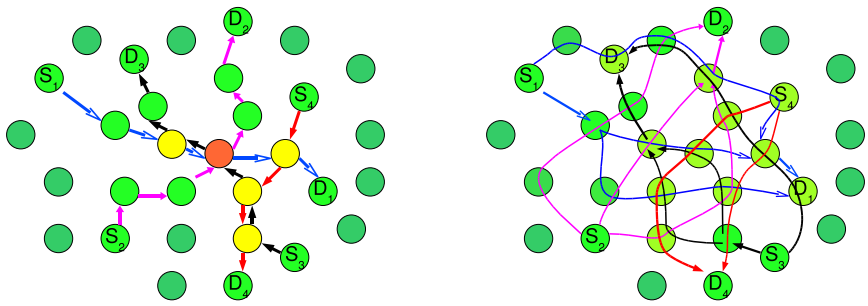
Se contenter de fournir des routes sans en préciser l'utilisation peut sembler de prime abord plus logique : les couches basses forment un support sur lequel s'adaptent les couches supérieures. Néanmoins il parait légitime de penser que c'est bien un certain choix dans la façon de répartir l'information qui fait de l'approche multiroutes une idée intéressante. Plusieurs routes ? Soit ! Mais dans quel but ? S'il s'agit bien d'améliorer la réception de l'information, il convient de se demander ce qui précisément s'y oppose.

Problème 1 : disparition des liens ou des nœuds Il n'y a malheureusement aucun moyen de lutter directement contre ce problème à moins d'imposer une stabilité aux nœuds et de leur interdire de disparaître. De manière plus réaliste, on peut néanmoins penser que si l'information est présente en plusieurs exemplaires sur des trajets distincts, elle sera moins sensible aux aléas du réseau. En poussant le raisonnement à l'extrême, on peut penser que l'inondation - qui consiste à dupliquer l'information en chaque nœud - peut être la meilleure garantie de réception.

Problème 2 : surcharge de nœuds Si l'information routée se concentre systématiquement sur certains axes, un engorgement est possible. Dans certains cas, cet encombrement est structurel. Ainsi, lorsqu'un nœud est l'unique point de connexion entre deux zones plus fortement connexes, toute tentative de communication entre des nœuds situés de part et d'autre requiert nécessairement la participation du nœud de connexion. Néanmoins dans d'autres cas, le protocole de routage utilisé peut avoir tendance à mobiliser plus fortement certains nœuds alors que d'autres resteront inactifs (voir figure 4.1 (a)). Comme toute entité de communication, un nœud trop fortement sollicité ne pourra gérer la totalité des données qu'il reçoit. À l'inverse en éclatant un flux sur un ensemble de routes (comme sur la figure 4.1 (b)), on augmente le nombre total de participants et on réduit de fait le risque de surcharge de certains.

Problème 3 : manque de liens Cette situation peut aboutir, au pire, à l'isolement de certaines parties d'un réseau, ou, dans des cas de moindre gravité, à un choix très limité de routes afin de joindre certains nœuds. Plus encore que dans le problème 1, il paraît difficile de résoudre ce problème par une approche basée sur la couche de routage ou des couches supérieures. On peut bien entendu envisager d'augmenter la portée des nœuds, mais cette stratégie a un impact direct sur les composants matériels du réseau. Or, notre but est d'améliorer le transfert en modifiant l'usage des nœuds et non les nœuds eux-mêmes. L'article [eSVKeSKT04] propose une solution originale : ajouter au réseau des nœuds spécifiquement dédiés à l'amélioration de la connectivité du réseau. Cependant cette logique est en soi à contre courant de l'idée de réseau ad-hoc. Les nœuds en question, ne correspondant pas à des utilisateurs, forment une infrastructure sous-jacente prérequise.

Problème 4 : mauvaise qualité des liens Ceci peut être dû à la présence de perturbations physiques externes au réseau ou à la distance trop importante entre les nœuds. S'il s'agit d'un problème global (concernant tous les liens), aucune solution se limitant à une architecture ad hoc ne peut résoudre cette difficulté. À l'inverse, une grande variabilité dans la qualité des liens incite à préférer l'utilisation des meilleurs et à délaisser les autres.



(a) Concentration des flux (b) Eclatement des flux

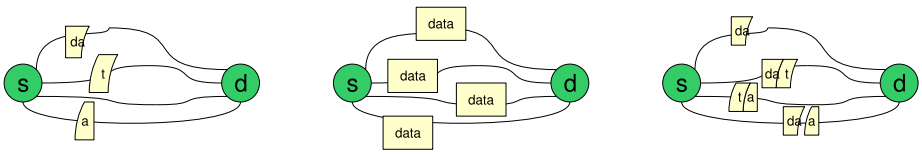
FIG. 4.1 – Risque d'enconbrement dû à la concentration des flux sur certains nœuds

L'utilisation simultanée des routes offre des perspectives intéressantes pour les deux approches évoquées, à savoir la duplication de l'information d'une part et l'éclatement du flux d'autre part. Néanmoins elles ne sont pas pleinement conciliables. On ne peut en effet dupliquer l'information de manière conséquente et souhaiter en même temps un débit local très faible (sauf à supposer un nombre illimité de routes disponibles, ce qui est toutefois irréaliste). Avant de sélectionner les routes à proprement parler il apparaît intéressant de déterminer quels peuvent être les grands types de stratégies de distribution de l'information sur les routes et dans quelles mesures elles peuvent répondre aux approches dégagées ci-dessus.

II.2 Répartir l'information

En supposant connu un jeu de routes entre une source et une destination, on peut envisager les stratégies de répartition de l'information suivantes :

- La répartition pure consiste à diviser l'information en plusieurs parties, chacune étant envoyée sur une route différente comme dans la figure 4.2 (a) Elle peut intervenir à plusieurs niveaux de granularité. Nous pouvons entre autre distinguer la répartition par paquets (les paquets sont un à un dispersés sur une route différente et équitablement répartis), la répartition par flux (des transferts de natures diverses mais ayant même source et destination utilisent des routes distinctes) ou encore des répartitions intra-paquet (les paquets originaux sont découpés en fragments routés indépendamment les uns des autres).
- La duplication consiste à créer des copies de l'information originale et à envoyer chaque version sur une route différente comme dans la figure 4.2 (b)).
- Une stratégie mixte permet d'intégrer une redondance partielle à l'information. La mise en application de cette stratégie peut faire appel à des techniques diverses. On peut par exemple, comme dans la figure 4.2 (c) découper l'information originale puis créer des copies de fragments dispersés sur les routes. D'une manière plus générale on souhaite obtenir une configuration où les données transmises sur une seule route ne correspondent qu'à un fragment de l'information originale, mais où, contrairement à la répartition pure, deux routes distinctes pouvent véhiculer des données partiellement redondantes entre elles. De fait, une sous-partie des routes peut en général suffire à reproduire cette information. C'est très précisément la fonction du codage à description multiple (MDC).



(a) Répartition pure (b) Duplication (c) Stratégie mixte

FIG. 4.2 – Les différentes stratégies d'utilisation des routes

La répartition pure est une très bonne stratégie en terme de répartition de charge, elle n'est en revanche d'aucune utilité face à la disparition d'une route. La duplication possède les propriétés inverses : très

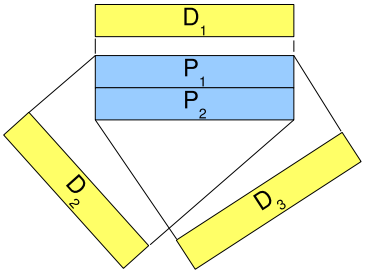


FIG. 4.3 – Exemple de transformation de paquets en descriptions Mojette ($N = 3$,$M = 2$)

robuste face à la perte des routes, elle conserve néanmoins le débit local du transfert et augmente grandement le débit global. L'approche mixte fournit la plus large palette de possibilités. On peut en effet paramétrer le niveau de redondance introduite et les parties sur lesquelles elle porte - ces variations sont expliquées plus en détail dans la section suivante. On notera que les deux premières approches peuvent être vues comme des cas dégénérés de la troisième : une répartition correspond à une redondance nulle, une duplication à une redondance maximale.

II.3 Une redondance contrôlée

Comment introduire de la redondance tout en la contrôlant ? Les techniques de descriptions multiples (MDC) présentées dans le chapitre 2 fournissent une approche intéressante. Un paquet ou un ensemble de paquets de données $P_1, \cdots, P_{N'}$ (formant une information notée $\vec{P}$) est ainsi converti en N descriptions $D_1, \cdots, D_N$ correspondant à une information $\vec{D}$. Lors de l'opération inverse, on suppose que toutes les descriptions préalablement créées ne sont plus nécessairement disponibles, certaines ayant été perdues par le réseau. Néanmoins plus le nombre de descriptions effectivement disponibles est grand plus l'information reconstruite $\hat{P}$ est proche de l'originale.

Lorsque l'information $\vec{P}$ est scalable, cette approche s'adapte particulièrement bien. En effet, dans ce cas, cela fait sens de chercher à reconstruire progressivement l'information. Néanmoins, dans cette étude, on ne se concentrera pas sur un type particulier de données. De fait, aucune hypothèse n'est émise quand à la nature et les propriétés de celles-ci. On suppose simplement que les données sont décomposées par les couches supérieures de la source en paquets pouvant être indépendamment reçus par les couches supérieures de la destination. La notion de qualité de reconstruction ne fait donc pas nécessairement sens. En effet si une image peut être plus ou moins finement reconstruite, il est moins évident de transformer un texte de façon à ce qu'une reconstruction partielle ait systématiquement un sens ou un intêrét. Aussi le choix d'une version binaire dégradée de la MDC semble plus avisé : $\vec{P}$ est au final soit restaurée entièrement, soit perdue intégralement. Dans ces conditions un entier M détermine le seuil de reconstructibilité : M est le nombre minimal de descriptions nécessaires pour retrouver l'information originale. Un exemple standard peut être la création de descriptions Mojette comme dans la figure 4.3.

À noter que la plupart des méthodes MDC génèrent généralement des descriptions équivalentes, c'est

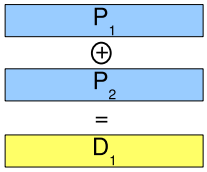


FIG. 4.4 – Exemple de transformation de paquets en descriptions par utilisation de Xor ($N = 3$,$M = 2$)

à dire que chacune possède un pouvoir de reconstruction équivalent à celui d'une autre. Ceci permet d'affirmer que M est bien un paramètre de la méthode de codage caractéristique de tous les paquets.
Il peut être toutefois intéressant de remarquer que les routes sur lesquelles ces descriptions vont être réparties ne sont pas nécessairement aussi sures les unes que les autres. L'équivalence des reconstructions n'est donc pas systématiquement un avantage en soi. Une variante peut néanmoins consister à regrouper certaines descriptions et à former ainsi des méta-descriptions possédant un pouvoir de reconstruction plus grand, au prix d'une taille plus importante. Ce type de méthode permet alors de jouer plus finement sur l'adaptation des descriptions aux routes. Le codage par Mojette permet quant à lui de créer a priori des descriptions dont le pouvoir de reconstruction est variable.
Dans le cas où $\vec{P}$ représente un ensemble de paquets on peut par ailleurs choisir d'en privilégier certains en augmentant la redondance qui leur est liée. Différents seuils de reconstruction peuvent alors être associés avec un nombre croissant de paquets reconstruits. En théorie, la MDC peut alors s'appliquer pleinement. On suppose cependant que la couche de routage ne possède ici aucune connaissance sur le contenu des paquets. Il n'y a donc pas lieu d'en privilégier certains dans le fonctionnement du routage. On peut en revanche considérer des transformations systématiques pour lesquelles l'information avec redondance $\vec{D}$ contient $\vec{P}$. Dans un cadre MDC cela correspond donc à des cas où les paquets originaux constituant $\vec{P}$ sont également des descriptions $P_1, \cdots, P_M$. À celles-ci s'ajoutent des descriptions de redondance pure $D_{M+1}, \cdots, D_N$; le total formant l'information $\vec{D}$. La figure 4.4 propose un exemple simple de codage systématique appliqué à des paquets. La description de redondance est ici le résultat d'un XOR entre deux paquets de données.

II.4 Conclusion

Cette partie a permis de déterminer en termes plus précis le but de notre approche multichemins. Il s'agit d'introduire une redondance maitrisée en transformant le flux original en divers sous-flux routés différement les uns des autres. Suivant le taux de redondance introduit on opére un compromis entre l'équilibrage de la charge (chaque sous-flux possède a priori un débit plus faible) et la baisse d'importance de chaque sous-flux relativement au flux initial (chaque sous-flux est moins critique). Le choix effectif de la répartition est un problème ouvert que nous allons traité dans la section IV. Maintenant que l'usage des transferts multichemins apparaît plus clairement, il convient de déterminer comment sélectionner les différentes routes.

III Sélectionner des routes multiples

L'introduction d'une redondance paramétrée semble potentiellement à même d'augmenter la capacité à recevoir correctement l'information. Néanmoins, on voit bien que cette approche nécessite d'avoir des routes à disposition. L'établissement de ces routes (dèsignés par le k-uplet $\mathcal{K} = (R_1, \cdots, R_k)$) qui est la raison d'être de tout protocole de routage, ne peut pas se faire sans tenir compte des deux points suivants :

– la méthode de récupération de l'information de topologie ;
– les caractéristiques du réseau.

III.1 Récupérer de l'information du réseau

Dans les précédents paragraphes, l'ensemble des routes a été exploité comme s'il s'agissait d'une donnée déjà à disposition. En pratique ces routes doivent être définies à un moment ou un autre par le protocole de routage. Celui-ci n'existe cependant qu'à travers la coopération des nœuds. On se propose d'étudier quelle est la stratégie adoptée par ces derniers pour choisir les routes.

III.1.a Méthodes des procotoles existants

Deux grands types de stratégies apparaissent dans les protocoles existants :

– récupérer les routes tracées par le passage de certains types de paquets ;
– extraire les routes de la topologie connue de certains nœuds.

Cette dualité rejoint en partie la classification usuelle routage réactif / routage proactif. En effet dans le cas des protocoles réactifs, là où les routes se dessinent naturellement avec le retour à la source de paquets de contrôles spécifiques. À l'inverse les protocoles proactifs font plutôt le choix d'accumuler le maximum d'information topologique en chaque nœud. La construction de la ou des routes peut alors souvent se résumer à une opération locale où chaque nœud décide, en consultant sa mémoire, quelle route semble préférable pour atteindre telle ou telle destination. On notera que ces alternatives ne se recouvrent cependant pas exactement : DSDV est l'exemple typique d'un protocole proactif où chaque nœud ne détermine, pour une destination donnée, que le premier segment de la route totale.

Dans le cas de recherche de plusieurs routes, dès lors que le choix des routes est réparti entre plusieurs nœuds, il devient plus complexe de comparer celles-ci. Si l'on considère par exemple les méthodes de [eSVKeSKT04], [eSRD01], [eMG01], [eSD99] et [eDEeJdJ04], aucune vision globale de la topologie n'est prise en compte dans le choix de l'ensemble de routes $\mathcal{K} = (R_1, \cdots, R_k)$. Les routes sont en effet dessinées par le passage des requêtes et réponses lors de la phase de recherche de route.

Dans l'article [eMG01], les auteurs reconnaissent à ce propos que dans ce type d'approche (qui est celle de DSR, AODV et SMR), les chemins empruntés par les requêtes et parvenant à la destination `D` appartiennent généralement à une même branche issue de la source `S` (voir figure 4.5). Ce phénomène limite grandement la diversité des chemins. Des mécanismes propres à SMR, AODVmulti ([eSVKeSKT04]) et AOMDV [eSRD01] permettent d'obtenir des routes disjointes (partiellement pour SMR). Néanmoins,

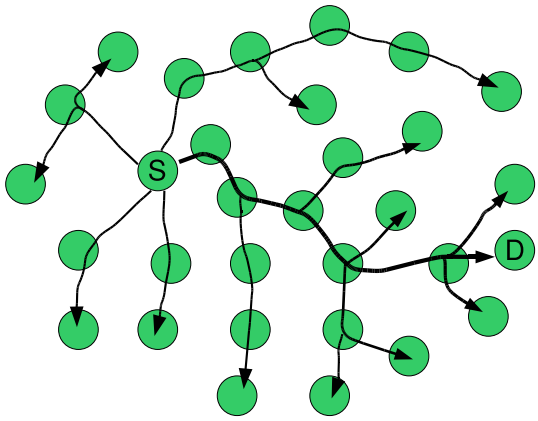


FIG. 4.5 – Convergence des routes

ils ne permettent que partiellement de prendre en considération l'ensemble des caractéristiques de $\mathcal{K}$ décrites dans le paragraphe III.2.
A contrario de l'approche consistant à laisser les routes se dessiner, toute méthode où est utilisée la topologie connue de certains nœuds possède l'avantage indéniable d'offrir une vue d'ensemble du réseau et donc l'avantage de pouvoir choisir des routes en prenant en compte leurs interactions.

III.1.b Le choix par la source

Pour un transfert donné, à supposer que les nœuds puissent acquérir une information suffisante sur le réseau, deux nœuds en particulier peuvent plus naturellement que les autres choisir les routes correspondantes au transfert en question. Il s'agit de sa source et de sa destination. Ainsi le propocole SMR donne à la destination la responsabilité du choix des routes. Néanmoins cette décision se limite à la sélection de routes parmi un ensemble créé par la circulation des RREQ de la source à la destination.
La source étant par définition le nœud à l'origine du transfert il est plus simple qu'elle choisisse elle-même les routes sur lesquelles elle va devoir répartir le flot d'information $\vec{P}$. Dans le cas contraire un échange préalable d'information serait en effet nécessaire pour que la destination informe la source des routes sélectionnées. On se doit toutefois de garder à l'esprit que `S` ne possède en toute rigueur qu'une vue partielle de la topologie. Cette vue est cependant suffisament grande pour contenir la destination `D` (sans quoi aucun transfert n'est de toute façon possible). `S` doit alors choisir les routes menant à `D`, puis déterminer une stratégie de répartition de l'information sur ces dernières.
Ces considérations étant faites, il semble donc que les protocoles accumulant naturellement l'information de topologie (c'est-à-dire *à état de lien* d'après les classifications établies au chapitre 1) sont particulièrement bien adaptés à notre approche. La plupart de ces protocoles sont, comme dit précédemment, des protocoles proactifs (comme OLSR).
La source peut donc à l'aide de ces protocoles rassembler une information conséquente sur le réseau. Comment opère-t-elle ce choix ? On souhaite instinctivement trouver des routes indépendantes, courtes

et en grand nombre. Toutefois la réalité physique limite grandement ces attentes.

III.2 Les contraintes du routage multichemins et la réalité pratique

Théoriquement une situation où le nombre de routes est grand est intéressante. En pratique il faut que ces routes correspondent réellement à des circuits différents pour l'information. L'indépendance des routes est donc un critère fondamental. Elle est réalisée lorsque :

– les routes sont disjointes (elles ne comportent comme nœuds communs que la source et la destination) ;
– les routes n'interfèrent pas (les nœuds d'une route sont hors de portée d'un point de vue électromagnétique des nœuds d'une autre route).

Le terme disjoint peut en fait désigner deux situations : la disjonction par les liens (aucun lien commun) et celles par les nœuds (aucun nœud commun mis à part `S` et `D`). Cette dernière est plus forte au sens où des routes disjointes par les nœuds le sont nécessairement par les liens.

Les prococoles AOMDV est AODVmulti se proposent d'ailleurs de fournir des routes disjointes (par les liens pour le premier, pas les nœuds pour le second). En pratique le nombre de routes disjointes peut être très limité comme le reconnaissent les auteurs de l'article [eSVKeSKT04]. L'exemple typique de la limitation du nombre de routes est celui où la source ou la destination est un nœud pendant (figure 4.6), c'est-à-dire un nœud ne possédant qu'un voisin. Dans ce genre de cas une seule route disjointe existe. D'une manière générale, en considérant les liens munis d'une capacité unitaire, c'est la valeur maximale d'un flot entre la source et la destination qui détermine le nombre maximale de routes totalement disjointes par les liens. Le nombre de routes totalement disjointes par les nœuds est donc encore plus faible. En pratique la limitation du nombre de routes est fréquemment imposée par le nombre de voisins au niveau de la source et de la destination, là où les routes divergent puis convergent. La recherche de routes disjointes à tout prix n'est donc peut-être pas la stratégie la plus adaptée, car elle peut limiter grandement le nombre de routes possibles. Aussi SMR fait-il le choix de ne sélectionner des routes au plus disjointes. En outre, même en supposant qu'il existe un nombre de routes disjointes au moins aussi grand que le nombre de routes recherchées, l'impératif d'indépendance peut conduire à la sélection de routes très longues (au sens du critère de recherche). Or, plus une routes est longue, moins elle est, a priori, satisfaisante. Les figures 4.7 montrent que la recherche d'une route nécessairement disjointe conduit dans (a) à sélectionner une route longue, qui peut s'avérer au final moins performante que la route sélectionnée dans (b), même si celle-ci exploite un lien déjà utilisé.

En ce qui concerne les interférences entre routes, celles-ci sont liées à la couche de liaison de données utilisée par les nœuds. En pratique, le principe général des protocoles de liaisons de données consiste souvent, comme dans l'exemple du 802.11, en une demande de prise de parole successive entre les nœuds. Lorsqu'un nœud a la parole, ses voisins se taisent afin d'empêcher des collisions de données qui résulteraient des interférences des ondes. Deux routes parallèles suffisament proches ont donc, avec ces méthodes de transmission, tendance à se géner mutuellement. Imposer aux routes une distance de sécurité est malheureusement une contrainte encore plus difficile à réaliser que leur disjonction. Néanmoins, dans [Gha08],

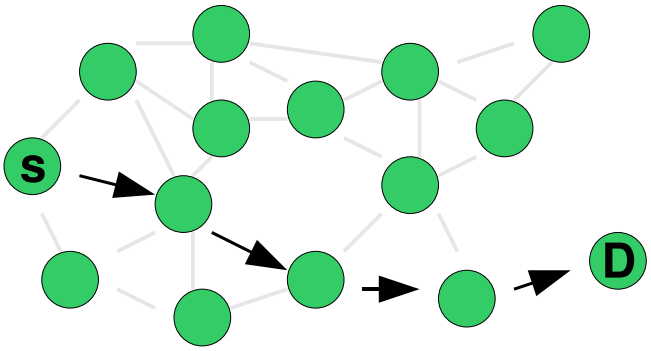
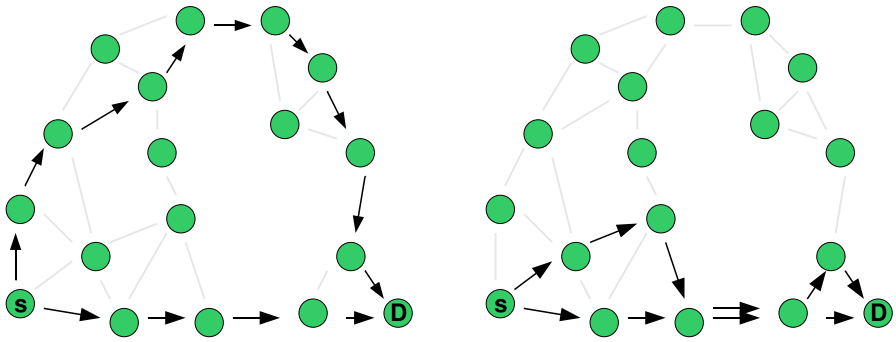


FIG. 4.6 – Noeud pendant



(a) Deuxième route longue (b) Deuxième route courte mais non disjointe

FIG. 4.7 – Allongement des routes sous la contrainte

l'auteur propose une méthode afin de limiter ce type d'influence. L'idée est grosso modo d'utiliser des codages orthogonaux sur des messages envoyés simultanément afin que chacun d'eux puisse être extraits du signal reçu.

En conclusion, on ne prendra en compte que deux critères pour $\mathcal{K}$:

– trouver des routes suffisament disjointes (par les nœuds si possible, par les liens sinon) ;

– éviter des routes trop longues.

III.3 Objectif théorique

L'idéal du choix des routes pourrait être modélisé en terme d'optimal mathématique : trouver un ensemble de routes qui sécurise au mieux la répartition. Encore faut-il être capable de définir clairement le critère à optimiser : délai de transmission ? énergie dépensée ? critères liés à la confidentialité ? Notre souci étant avant tout de favoriser la bonne réception de l'information originale (sans quoi les autres critères ont peu de sens), on considère dans notre étude le taux de paquet délivré comme le critère principale. Ceci revient donc à s'intéresser, pour une information originale $\vec{P}$ générée en `S`, à la probabilité de pouvoir la reconstruire au niveau de `D`. Ce problème est complexe. Cette probabilité, que nous nommerons fiabilité

du transfert $\mathcal{R}$ dépend à la fois :
– de la topologie disponible et de ses caractéristiques (faible ou forte mobilité, nœuds prenant déjà en charge d'autres transferts, énergie des nœuds...) ;
– du choix des routes définies sur celles-ci ;
– du choix de la redondance introduite sur l'information initiale ;
– de la répartition de l'information redondante sur les différentes routes.
Qui plus est, tous ces paramètres ont a priori une dimension temporelle : ils vont changer au fur et à mesure de l'évolution du réseau. La fiabilité n'est donc pas une constante mais une quantité dynamique. On suppose néanmoins qu'au moment du calcul du choix des routes par la source, celle-ci se fait une représentation de l'état du réseau que l'on peut supposer suffisament correcte pour une période T. Dans cet intervalle de temps, les paramètres du problèmes peuvent être considérés constants.

III.3.a Modélisation théorique du réseau

Un réseau ad hoc est modélisé par un *graphe* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ où $\mathcal{V}$ est l'ensemble des *sommets* (ou nœuds) et $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ l'ensemble des *arcs orientés*. Nous supposons d'une part que le graphe est sans boucle (aucun arc ne joint un sommet à lui-même). D'autre part, une paire quelconque de nœuds ne peut être connectée que par au plus un seul arc. Ceci justifie la notation $\mathtt{e} = (\mathtt{V}_1, \mathtt{V}_2)$ pour parler de l'unique arc de $\mathtt{V}_1$ à $\mathtt{V}_2$. On prendra bien soin de noter que $(\mathtt{V}_2, \mathtt{V}_1)$ est un arc différent de $\mathtt{e}$, appelé opposé de $\mathtt{e}$ et noté $-\mathtt{e}$.
Il est possible de munir le graphe d'une fonction de coût cout donnant aux nœuds et aux arcs un poids représentant la qualité qu'on leur prête (plus le poids de e ou de $\mathtt{V}$ est grand moins l'élément en question est intéressant). On peut en fait se limiter à pondérer les arcs : le coût d'un nœud $\mathtt{V}$ peut en effet être reporté de manière simple sur tous ses arcs entrant dans $\mathtt{V}$ (c'est-à-dire de la forme $\mathtt{e} = (\mathtt{V}', \mathtt{V})$). La fonction cout peut donc être définie comme une fonction $\mathcal{E} \to \mathbb{R}^+$. Cette fonction induit une notion proche de celle de distance mathématique entre les nœuds (mais a priori non symétrique sauf si $\mathrm{cout}[\mathtt{e}] = \mathrm{cout}[-\mathtt{e}]$ pour tout lien). Il est souhaitable que le poids corresponde à une quantité additive afin de pouvoir étendre cette notion aux routes. On peut penser à divers critères :
– un poids unitaire pour chaque lien (le coût total d'une route correspond alors au nombre de liens) ;
– le délai moyen de transmission du lien ;
– un critère additif lié au taux d'erreur binaire sur le lien ;
– un critère additif lié au débit sur le lien dû aux transferts déjà en cours sur le lien ;
– un critère additif lié à la stabilité du lien (pouvant notamment utiliser son âge) ;
– une combinaison de ces critères ...
Etant donnée une paire de nœuds distincts $(\mathtt{S}, \mathtt{D})$ nous appelons chemin ou route de $\mathtt{S}$ à $\mathtt{D}$ une séquence de nœuds $(\mathtt{V}_1, \mathtt{V}_2, ..., \mathtt{V}_r)$ telle que $(\mathtt{V}_q, \mathtt{V}_{q+1}) \in \mathcal{E}$, $\mathtt{V}_1 = \mathtt{S}$ et $\mathtt{V}_r = \mathtt{D}$. Si une fonction de coût est définie sur le graphe, on appelle coût de la route la somme $\sum_{q=1}^{r-1} \mathrm{cout}[(\mathtt{V}_q, \mathtt{V}_{q+1})]$.
Un sommet $\mathtt{V}$ est dit atteignable pour un sommet $\mathtt{S}$ s'il existe une route de $\mathtt{S}$ à $\mathtt{V}$. Etant donné que nous avons imposé des coûts strictement positifs aux arcs, il est bien connu que pour tout sommet $\mathtt{V}$ atteignable

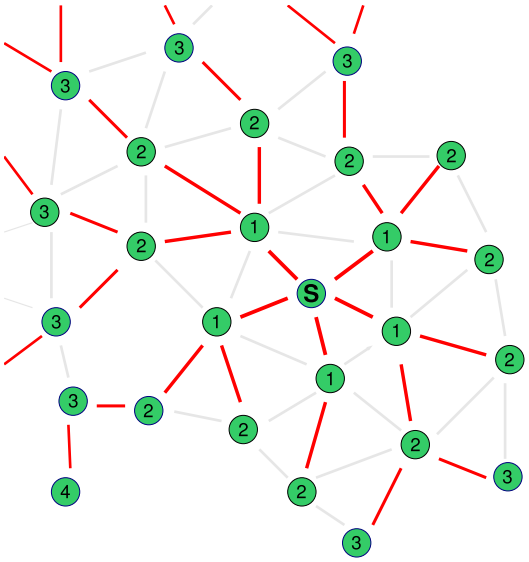


FIG. 4.8 – Arbre source du nœud S

de S il existe au moins une route de coût minimum (mais pas nécessairement une seule). Une telle route est appelée plus court chemin de S à V. Etant donné un sommet S, il est toujours possible de choisir un plus court chemin pour chaque nœud atteignable V de telle sorte que l'union de ces chemins forme un arbre. Ce dernier est appelé arbre source de S (voir figure 4.8).

On notera que la modélisation retenue utilise des graphes orientés, ceci afin de modéliser le fait que la communication entre deux nœuds peut n'être possible que dans un sens. Par ailleurs, même si les deux sens fonctionnent, le critère modélisé par la fonction de coût peut avoir des valeurs différentes suivant le sens. Dans ce cadre, il convient de préciser que deux routes R_1 et R_2 sont dites disjointes par les liens si pour tout arc e dans R_1, ni e ni $-$e ne sont dans R_2.

III.3.b Modélisation stochastique du fonctionnement des routes

La modélisation générale ci-dessus suppose une connaissance parfaite du réseau. Or, en pratique, l'information de topologie pour être exploitée doit être centralisée en un point du réseau. Dans notre étude il s'agit en l'occurence de la source du transfert. Rien ne garantit donc que la topologie $\mathcal{G}$ sur laquelle se développent les calculs corresponde parfaitement à la topologie réelle. Il ne s'agit en fait que de la topologie perçue par la source : celle qu'elle estime vraisemblable à un moment donné. Bien entendu, il y a de fortes chances que l'information connue de S portant sur son voisinage soit très proche de la réalité. À l'inverse, l'information concernant un nœud lointain doit parcourir un trajet plus grand. Elle peut être perdu plus facilement ou ne plus être à jour en arrivant à S. Pour ces raisons, si l'on modélise le réseau *du point de vue de* S, un certain nombre de caractéristiques deviennent aléatoires : celà traduit le fait que la source ne possède pas une connaissance certaine sur le comportement du réseau.

Supposons un k-uplet $\mathcal{K} = (R_1, \cdots, R_k)$ de routes défini sur $\mathcal{G}$. L'information originale $\vec{P}$ est transformée en information redondante $\vec{D}$ constituée de N descriptions D_i avec un seuil M indiquant le nombre de

descriptions nécessaire à la reconstruction. Par ailleurs à chaque route R_i est alloué un nombre N_i de descriptions tels que $\sum_{i=1}^{k} N_i = N$.
La réussite d'une route R_i est modélisée par la variable aléatoire de Bernoulli Y_i telle que :

$$Y_i = \begin{cases} 1 \text{ avec une probabilité } p_i \text{ si } R_i \text{ parvient à transmettre l'information allouée} \\ 0 \text{ avec une probabilité } 1 - p_i \text{ en cas d'échec} \end{cases}$$

La route R_i transmet donc à destination un nombre $N_i \cdot Y_i$ de descriptions. Le nombre total de descriptions reçues est donc $Z = \sum_{i=1}^{k} N_i \cdot Y_i$ et la probabilité de reconstruction définit la fiabilité de la transmission $\mathcal{R}$ par :

$$\mathcal{R} = \mathbb{P}(Z \geq M)$$

On note que dans un cas systématique la notion de fiabilité de $\mathcal{K}$ telle que définie ci-dessus ne s'applique plus. En effet, une reconstruction partielle de l'information originale $\vec{P}$ est possible. Il n'existe de fait plus une seule probabilité pour rendre compte des différents niveaux de reconstruction. On peut néanmoins étendre la fiabilité en la définissant comme la proportion moyenne d'information originale reconstruite. Pour cela, définissons Z' comme le nombre de paquets originaux reconstruits et M_i comme le nombre de *pseudo-descriptions* envoyés sur R_i, c'est-à-dire de descriptions qui soit par ailleurs des paquets originaux et donc directement exploitables (voir figure 4.9). Ces quantités sont regroupées dans le vecteur $\vec{M} = (M_1, \cdots, M_k)$. On notera que $\sum_i M_i = M$. La variable aléatoire Z' s'exprime alors comme :

$$Z' = \left(\sum_{i=1}^{k} M_i \cdot Y_i \right) \cdot \mathbf{1}_{[0,M-1]}(Z) + M \cdot \mathbf{1}_{[M,N]}(Z)$$

où $\mathbf{1}_E$ est la fonction caractéristique de l'ensemble E. La fiabilité est alors définie comme :

$$\mathcal{R} = \frac{1}{M} \mathbb{E}(Z')$$

On note que la nouvelle définition est cohérente avec l'ancienne. En effet, dans un cas non systématique on a $M_i = 0$ pour tout i. Autrement dit aucune route ne véhicule de pseudo-descriptions. D'où :

$$\begin{aligned} \mathcal{R} &= \tfrac{1}{M} \mathbb{E}\left(M \cdot \mathbf{1}_{[M,N]}(Z) \right) \\ &= \mathbb{E}\left(\mathbf{1}_{[M,N]}(Z) \right) \\ &= \mathbb{P}(Z \geq M) \end{aligned}$$

III.3.c Modélisation stochastique du fonctionnement des liens

On notera que les variables Y_i ne sont pas nécessairement indépendantes. Elles peuvent par ailleurs se décomposer en :

$$Y_i = \left(\prod_{\mathtt{e} \in R_i} X_{\mathtt{e}} \right) \cdot \left(\prod_{\mathtt{V} \in R_i} X_{\mathtt{V}} \right)$$

où $X_{\mathtt{e}}$ et $X_{\mathtt{V}}$ sont des variables aléatoires de Bernoulli modélisant respectivement la réussite du lien e et celle du nœud $\mathtt{V}$. La fiabilité $p_{\mathtt{e}}$ de chaque lien modélise le risque de disparition du lien en supposant que

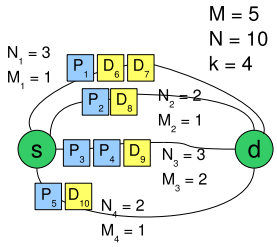


FIG. 4.9 – Répartition de $N = 10$ descriptions (6 pseudo-descriptions et 4 descriptions de redondance) sur $k = 4$ routes

les nœuds qui le composent restent valables. On supposera que les liens sont indépendants dans un but de simplification (même si en pratique ce n'est pas exact : le déplacement d'un nœud a, à priori un effet sur tous les liens auquel il est connecté). De fait, les variables $X_{\mathtt{e}}$ sont indépendantes entre elles, avec :

$$X_{\mathtt{e}} = \begin{cases} 1 \text{ avec une probabilité } p_{\mathtt{e}} \text{ si le lien } e \text{ parvient à transmettre l'information allouée} \\ 0 \text{ avec une probabilité } 1 - p_{\mathtt{e}} \text{ en cas d'échec} \end{cases}$$

La variable aléatoire $X_{\mathtt{V}}$ peut être décomposée en produit de trois variables aléatoires de Bernoulli :

$$X_{\mathtt{V}} = X_{\mathtt{V}}^{sv} \cdot X_{\mathtt{V}}^{tr} \cdot X_{\mathtt{V}}^{col}$$

La variable aléatoire $X_{\mathtt{V}}^{sv}$ modélise le risque de disparition du nœud `V` et est caractérisée par la fiabilité de survie $p_{\mathtt{V}}^{sv}$. La variable aléatoire $X_{\mathtt{V}}^{tr}$ correspond à la capacité du nœud à supporter le débit dont il a la charge. Enfin, $X_{\mathtt{V}}^{col}$ prend en compte la possibilité pour le nœud `V` de recevoir une information inexploitable à cause de collisions. Il paraît raisonnable de considérer que Y_V^{tr} et Y_V^{col} dépendent tous les deux du débit effectivement reçu. On peut donc supposer que le produit $Y_V^{res} = Y_V^{tr} \cdot Y_V^{col}$ est caractérisé par une probabilité p_V^{res}, fonction du débit λ_V reçu en `V` ($p_V^{res} = f(\lambda_V)$). Ce débit est une variable aléatoire puisqu'il dépend de la capacité des flux circulant théoriquement dans `V` à parvenir jusqu'à ce nœud. Or, ce fait est conditionné par le fonctionnement ou non des nœuds et liens précédents v sur les différentes routes. De fait, les différentes variables aléatoires Y_V^{res} ne sont pas indépendantes entre elles.
On notera que les quantitées $-\log(p_{\mathtt{e}})$ et $-\log(p_{\mathtt{V}}^{sv})$ représentent bien des coûts additifs définis sur la topologie perçue $\mathcal{G}$. Il n'en est pas de même pour $-\log(p_{\mathtt{V}}^{res})$ qui est une quantité aléatoire a priori dépendante de la validité des routes. En effet, si un certain nombre de routes initialement prévues pour passer par `V` sont invalidées par le dysfonctionnement d'un lien ou d'un nœud précédent (ce qui est un événement aléatoire), le débit sera moindre en `V` et ce lien aura alors plus de chance de résister au flux d'information. De fait, les quantités $-\log(p_{\mathtt{V}}^{res})$ ne sont pas des quantités intrinsèques à $\mathcal{G}$ mais dépendantes du fonctionnement aléatoire de $\mathcal{K} = (R_1, \cdots, R_k)$.

III.3.d Maximisation de la fiabilité

L'objectif mathématique revient donc à optimiser la fiabilité $\mathcal{R}$ en choisissant le meilleur jeu de routes et la meilleure répartition des descriptions sur celui-ci. On cherche donc dans le cas non systématique :

$$(k^*, \mathcal{K}^*, \vec{N}^*, M^*) = \arg\max_{(k,\mathcal{K},\vec{N},M)} \mathcal{R}$$

et dans le cas systématique :

$$(k^*, \mathcal{K}^*, \vec{N}^*, \vec{M}^*) = \arg\max_{(k,\mathcal{K}\vec{N},\vec{M})} \mathcal{R}$$

avec $\mathcal{K} = (R_1, \cdots, R_k)$, $\vec{N} = (N_1, \cdots, N_k)$, $\vec{M} = (M_1, \cdots, M_k)$ et $M = M_1 + \cdots + M_k$.
Optimiser mathématiquement un tel système est un problème complexe. Dans ces conditions nous nous contenterons d'une méthode pratique pour la sélection des routes en cherchant à réaliser pour celles-ci un bon compromis entre les impératifs d'indépendance et de longueur réduite. La problématique de la répartition pourra alors dans un second temps tenir compte d'un tel choix de routes.

III.4 Algorithmes existants à objectif similaires

Parmi les méthodes habituelles du problème de théorie des graphes consistant en l'extraction de plusieurs routes, la plupart sont basées sur un algorithme du plus court chemin au sens monoroute. Le plus courant de ces derniers est Dijkstra, décrit en 1. Il permet de trouver l'arbre source d'un nœud `S`. Autrement dit, l'arbre inclus dans le graphe $\mathcal{G}$, de racine `S`, défini comme l'union des plus courts chemins de `S` à chaque nœud.
Le problème de trouver non plus un mais plusieurs plus courts chemins a abouti à une littérature abondante étant donné les multiples variantes possibles. Une des plus commune (présentée par exemple dans [Epp99], [eMCS95] ou [eMMeSS07]) est de trouver successivement des chemins distincts deux à deux tels que le premier est le plus court entre `S` et `D`, le second est le plus court disctinct du premier, etc. A priori deux chemins peuvent donc être extrêmement similaires puisque la seule condition requise est qu'il existe au moins un élément qu'ils ne partagent pas. Aussi ce type de problème ne correspond pas vraiment à notre objectif, dans lequel on souhaite trouver des routes suffisament disjointes.
La solution la plus communément utilisée afin d'extraire non pas une mais k routes disjointes (par les liens) est décrite par l'algorithme 3. Elle consiste en une utilisation répétée d'un algorithme du plus court chemin (Dijkstra dans l'exemple) suivie d'une suppression des liens de la route selectionnée dans le graphe. À chaque étape le graphe se réduit afin d'interdire l'usage de liens précédement sélectionnés. Une version permettant d'obtenir des routes disjointes par les nœuds est obtenue facilement en supprimant non seulement les liens mais les nœuds des routes. Cet algorithme fournit donc : la route la plus courte, la route la plus courte et complétement disjointe (par les nœuds ou les liens) de la première, la route la plus courte et complétement disjointe des deux précédentes, etc. Ces routes sont appelées les k successivement plus courts chemins disjoints. L'article [eWDG94] présente une variation dans laquelle un lien peut être emprunté un nombre de fois prédéfini (ce qui revient à autoriser l'existence de liens parallèles, c'est-à-dire reliant le même couple de nœuds).

`Dijkstra`(S,$\mathcal{G}$)

Données : $\mathcal{G}$: un graphe valué, S : un nœud

Résultat : predecesseurs : tableau des prédécesseurs, équivalent à un arbre source

début

  pour tous les $V \in \mathcal{V}$ **faire**

    distances [V] $\leftarrow +\infty$

    predecesseurs [V] $\leftarrow$ nil

  fin

  distances [S] $\leftarrow 0$

  pour tous les $V_{voisins} \in$ `voisinageDe` *(S)* **faire**

    distances $[V_{voisins}] \leftarrow$ cout $[(S, V_{voisin})]$

    predecesseurs $[V_{voisins}] \leftarrow S$

  fin

  $\mathcal{W} \leftarrow \mathcal{V} \setminus \{S\}$

  tant que $\mathcal{W} \neq \emptyset$ **faire**

    $V_{min} \leftarrow \arg\min_{V \in \mathcal{W}}$ distances [V]

    $\mathcal{W} \leftarrow \mathcal{W} \setminus \{V_{min}\}$

    pour tous les $V_{voisin} \in$ `voisinageDe` *(V_{min})* $\cap \mathcal{W}$ **faire**

      `mettreAJour` (distances, predecesseurs, V_{min}, V_{voisin})

    fin

  fin

  retourner predecesseurs

fin

Algorithme 1 : Algorithme du plus court chemin de Dijkstra

`mettreAJour`(distances, predecesseurs, V_{min}, V_{voisin})

Données : V_{min} : le noeud non explorée le plus proche de la source

V_{voisin} : un voisin du précédent

distances : tableau des distances des noeuds à la source

predecesseurs : tableau des prédécesseurs

début

  $e \leftarrow (V_{neigh}, V_{min})$

  si distances *[V_{voisin}]* $>$ distances *[V_{min}]* $+$ cout *[e]* **alors**

    distances $[V_{voisin}] \leftarrow$ distances $[V_{min}] +$ cout$[e]$

    predecesseurs $[V_{voisin}] \leftarrow V_{min}$

  fin

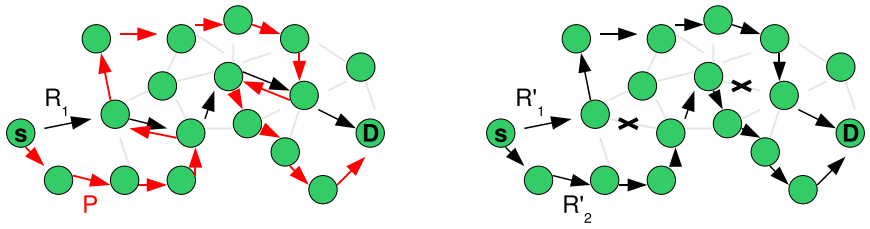
fin

Algorithme 2 : Fonction de mise à jour des tableaux de prédécesseurs et de distance

```
DijkstraEtSuppression( S, D, 𝒢, k )
Données : 𝒢 : un graphe valué, S et D : deux nœuds
k : nombre de routes
Résultat : 𝒦 : ensemble de k routes joignant S à D
début
    𝒦 ← ∅
    Ĝ ← 𝒢
    pour tous les i ← 1 à k faire
        arbreSource ← Dijkstra ( S, Ĝ )
        R ← routeVers ( D, arbreSource)
        pour tous les e ∈ liensDans (R) faire
            Ê ← Ê \ {e, −e}
            Ĝ ← (V̂, Ê)
        fin
        𝒦 ← 𝒦 ∪ {R}
    fin
    retourner 𝒦
fin
```

Algorithme 3 : Algorithme de recherche par suppression de routes

Une autre solution mathématique est l'algorithme de Suurballe (présenté dans [Suu74]). Une variante est présentée pour le cadre ad hoc dans [eZJHeEGS02]. L'objectif de ce type d'algorithme est plus simple à concevoir que celui du précédent : il permet de trouver les k routes disjointes par les nœuds dont la somme des coûts est minimale. Autrement dit, il minimise le critère $J = \sum_{i=1}^{k} \text{cout}[R_i]$. Bien que son fonctionnement soit complexe, on peut en décrire les mécanismes généraux. Après avoir trouvé une première route R_1 une réévaluation des poids des liens tend à inciter de futurs chemins à repasser par les arcs opposés à ceux de R_1 (en leur attribuant éventuellement un coût négatif). Cette méthode permet de trouver un nouveau chemin P a priori non nécessairement disjoint de R_1 (voir figure 4.10 (a)). Enfin une méthode de dèsentrelacement (voir figure 4.10 (b)) déduit deux routes disjointes R'_1 et R'_2 en supprimant les liens parcourus dans les deux sens. Après une nouvelle réévaluation des poids des arcs, un nouveau chemin P' est recherché, là encore non nécessairement disjoint de R'_1 et R'_2. L'algorithme s'arrête si le nombre de routes recherchées est atteint ou si aucune nouvelle route disjointe ne peut être trouvée.
Les deux algorithmes présentés ci-dessus ne sont que partiellement satisfaisants pour les raisons évoquées dans la section III.2. En l'occurence ils imposent tous les deux des contraintes d'indépendance stricte des routes. Par ailleurs, si l'algorithme de Suurballe semble en première approche plus intéressant parce qu'il optimise additivement le poids des routes, cette optimisation n'est pas celle recherchée en terme de fiabilité.



(a) Calcul d'un chemin P non nécessairement disjoint de P

(b) dèsentrelacement aboutissant sur la création des routes R'_1 et R'_2

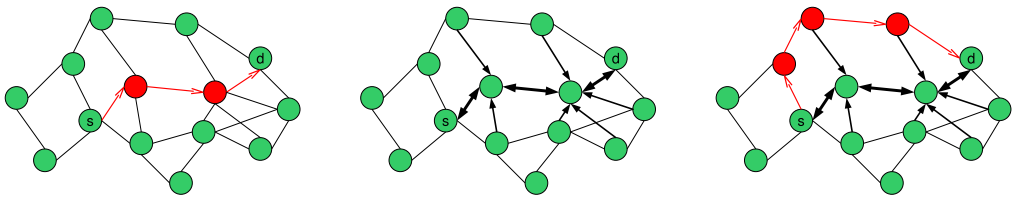
FIG. 4.10 – Etape de dèsentrelacement opérée dans l'algorithme de Suurballe

III.5 Notre proposition

L'algorithme proposé reprend l'idée de recherches successives de plus courts chemins. Néanmoins on va ici non pas interdire les portions déjà utilisées mais simplement les défavoriser. À cette fin, à chaque fois qu'une nouvelle route est définie, nous incrémentons le poids des liens qui la composent ou qui pointent vers elle afin de rendre virtuellement les liens et nœuds qui la compose moins interessants. Si une portion du chemin est malgré tout inévitable il sera empruntée à nouveau. De même, les longs détours sont évités tant que les raccourcis ne sont pas utilisés suffisament ; c'est-à-dire jusqu'à ce que les variations des chemins les plus courts deviennent suffisament pénalisées pour rendre les routes longues intéressantes.

III.5.a Spécification

Soit $f_p : \mathcal{R}^{*+} \to \mathcal{R}^{*+}$ et $f_e : \mathcal{R}^{*+} \to \mathcal{R}^{*+}$ deux fonctions telles que $id < f_p$ et $id \leq f_e < f_p$ (avec id la fonction identitée). L'algorithme proposé (voir 4) prend en paramètre le graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \text{cout})$, le couple source destination $(S, D) \in \mathcal{E}^2$ et l'entier k correspondant au nombre de routes recherchées. Il retourne un k-uplet $\mathcal{K} = (R_1, R_2, ..., R_k)$ de routes entre `S` et `D`.
La figure 4.11 décrit les premières étapes du fonctionnement de l'algorithme proposé.



(a) Extraction de la première route

(b) Incrémentation des coûts des liens

(c) Extraction de la deuxième route

FIG. 4.11 – Exemple du fonctionement de l'algorithme 4

```
DijkstraMultiroute( S, D, 𝒢, k )
Données : 𝒢 : un graphe valué, S et D : deux nœuds
k : nombre de routes
Résultat : 𝒦 : ensemble de k routes joignant S à D
début
    𝒦 ← ∅
    𝒢_1 ← 𝒢
    pour tous les i ← 1 à k faire
        arbreSource ← Dijkstra ( 𝒢_i, S )
        R ← routeVers ( D, arbreSource)
        pour tous les e ∈ ℰ faire
            si e ∈ liensDans (R_i) et −e ∈ liensDans (R_i) alors
                cout_{i+1}[e] ← f_p(cout_i[e])
            fin
            sinon
                si tete (e) ∈ R_i alors
                    cout_{i+1}[e] ← f_e(cout_i[e])
                fin
                sinon
                    cout_{i+1}[e] ← cout_i[e]
                fin
            fin
        fin
        𝒢_{i+1} ← (𝒱, ℰ, cout_{i+1})
        𝒦 ← 𝒦 ∪ {R}
    fin
    retourner 𝒦
fin
```

Algorithme 4 : Extractions de k routes de $\mathcal{G}$ allant de S à D

III.5.b Le rôle des fonctions incrémentales

Le rôle des fonction f_p et f_e est de moduler le compromis entre routes disjointes par les nœuds et routes disjointes par les liens. La disjonction par les nœuds implique bien évidement celle par les liens. Suivant les cas, le comportement de l'algortihme est :

- si $id < f_e = f_p$ de fournir des routes si possible disjointes par les nœuds ;
- si $id = f_e < f_p$ de fournir des routes si possible disjointes par les liens ;
- si $id < f_e < f_p$ de fournir des routes si possible disjointes par les nœuds et dans le cas contraire disjointes par les liens.

III.5.c Complexité algorithmique

L'algorithme de Dijskstra posséde une complexité temporelle en $O(|\mathcal{V}|^2 + |\mathcal{E}|)$. La complexité générale de notre algorithme est donc en $O(k(|\mathcal{V}|^2 + |\mathcal{E}|))$. Cependant, à chaque étape il n'est pas nécessaire de réappliquer l'algorithme de Dijkstra à tous les sommets. Soit R_i la i-ème route, considérons $\mathcal{A}_i$, l'ensemble des sommets de la branche de $arbreSource_i$ qui contient R_i. Autrement dit, le sommet v est dans $\mathcal{A}_i$ si le plus court chemin de s à v de $\mathcal{G}_i$ partage au moins un arc commun avec R_i. Soit $\mathcal{B}_i = \mathcal{V} \setminus \mathcal{A}_i$. On peut prouver que pour chaque sommet x de $\mathcal{B}_i$, le chemin le plus court de s à x extrait de $arbreSource_i$ ne change pas à l'étape $i+1$.

Preuve : Pour $v \in \mathcal{V}$ appelons $R^{s\to v}$ (respectivement $R^{s\to v}_{bis}$) le plus court chemin de s à v dans le graphe $\mathcal{G}_i$ (respectivement $\mathcal{G}_{i+1}$). $R^{s\to n}$ est donc le chemin de s à v contenu dans l'arbre source $arbreSource_i$ à la i-ème étape. On peut noter que si $v \in \mathcal{B}_i$, alors tous les nœuds de $R^{s\to v}$ appartiennent à $\mathcal{B}_i$. Supposons que pour un certain $x \in \mathcal{B}_i$, le plus court de s à x soit différent entre les étapes i et $i+1$. Autrement dit, x est tel que :

$$R^{s\to x} \neq R^{s\to x}_{bis}$$

On se propose de démontrer que ce changement de chemin est sans intêrét.

1. Etant donné un lien e, son coût augmente entre les étapes i et $i+1$ si est seulement si le nœud $tete(\mathtt{e})$ est dans $R_i = R^{s\to d}$. Il faut donc a fortiori que $tete(\mathtt{e})$ soit dans $\mathcal{A}_i$. Comme chaque nœud de $R^{s\to x}$ est dans $\mathcal{B}_i$, le coût de chacun des arcs de $R^{s\to x}$ est inchangé dans $\mathcal{G}_{i+1}$. Idem pour le coût total de $R^{s\to x}$:
$$c_{i+1}(R^{s\to x}) = c_i(R^{s\to x})$$
2. Le fait que le chemin $R^{s\to x}$ ait été selectionné à l'étape i implique que son ancien coût est plus petit ou égal à l'ancien coût de $R^{s\to x}_{bis}$:
$$c_i(R^{s\to x}) \leq c_i(R^{s\to x}_{bis})$$
3. Le fait que le chemin $R^{s\to x}_{bis}$ ait été selectionné à l'étape $i+1$ implique que son nouveau coût est égal ou plus petit que le nouveau coût de $R^{s\to x}$:
$$c_{i+1}(R^{s\to x}_{bis}) \leq c_{i+1}(R^{s\to x})$$
4. Comme à chaque étape le coût des arcs ne peut diminuer, celui des routes ne le peut pas non plus. Il est donc certain que $c_i(R^{s\to x}_{bis}) \leq c_{i+1}(R^{s\to x}_{bis})$. On a donc :
$$c_i(R^{s\to x}) \leq c_i(R^{s\to x}_{bis}) \leq c_{i+1}(R^{s\to x}_{bis}) \leq c_{i+1}(R^{s\to x}) = c_i(R^{s\to x})$$
5. Ceci prouve que le coût de $R^{s\to x}_{bis}$ est également constant et donc égal à celui de $R^{s\to x}$. Ce coût est encore minimal à l'étape $i+1$. Nous n'avons par conséquent pas besoin du nouveau chemin $R^{s\to x}_{bis}$

Soit n_{vois} le nombre de voisins de la source. Après une première étape complète de calcul du plus court chemin, il est possible de limiter les étapes suivantes en n'appliquant Dijkstra que sur une sous-branche (voir figure 4.12). Celle-ci contient en moyenne $|\mathcal{V}|/n_{vois}$ sommets et environ $|\mathcal{E}|/n_{vois}$ arcs, d'où une complexité globale de $O(k(\frac{1}{n^2_{vois}}|\mathcal{V}|^2 + \frac{1}{n_{vois}}|\mathcal{E}|))$.

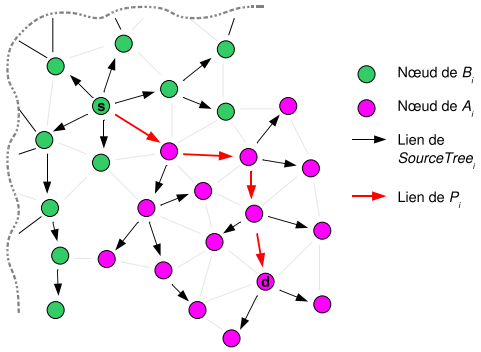


FIG. 4.12 – Partie du graphe à mettre à jour entre deux étapes

III.6 Conclusion

Nous avons donc proposé une approche pratique inspirée d'algorithmes connus dont on peut s'attendre à ce qu'elle s'adapte plutôt bien au problème qui nous intéresse (elle sera d'ailleurs mis en application dans les protocoles proposés aux chapitres 5 et 6). Dans cette approche, les fonctions f_p et f_e permettent de jouer sur le paramétrage. Leur valeur détermine en effet dans quelle mesure on cherche à privilégier soit la divergence des routes et donc leur indépendance, soit leur faible coût dû à une longueur restreinte.

IV Répartition sur les routes

Maintenant que l'on possède un procédé de sélection des routes, la problématique de répartition sur les routes peut être envisagée pleinement. Il s'agit de répondre à la question : combien de descriptions attribuer à chaque route ? Autrement dit, quelles valeurs donner à $N_1, \cdots, N_k$? Dans le cadre d'un codage systématique, il convient par ailleurs de déterminer la répartition des paquets originaux, autrement ditc'est-à-dire de choisir, pour chaque route R_i, en plus du nombre de descriptions N_i envoyés dessus, le nombre M_i (avec $0 \leq M_i \leq N_i$) de *pseudo-descriptions*.

IV.1 Reconsidération du problème d'optimisation de la fiabilité

Le problème d'optimisation précédent peut se ramener à :

– dans un cas non systématique :

$$\begin{aligned}(\vec{N}^*, M^*) &= \arg\max_{(\vec{N},M)} \mathcal{R} \\ &= \arg\max \mathbb{P}\left(\sum_{i=1}^{k} N_i \cdot (\prod_{\mathsf{e}\in R_i} X_{\mathsf{e}}) \cdot (\prod_{v\in R_i} Y_v^{sv} \cdot Y_v^{res}) \geq M\right)\end{aligned}$$

– dans un cas systématique :

$$\begin{aligned}(\vec{N}^*, \vec{M}^*) &= \arg\max_{(\vec{N},\vec{M})} \mathcal{R} \\ &= \arg\max \left(\mathbb{P}\left(\sum_{i=1}^{k} N_i \cdot Y_i \geq M\right)\right. \\ &\left. + \mathbb{P}\left(\sum_{i=1}^{k} N_i Y_i < M\right) \cdot \tfrac{1}{M} \cdot \mathbb{E}\left(\sum_{i=1}^{k} M_i \cdot Y_i | \sum_{i=1}^{k} N_i \cdot Y_i < M\right)\right)\end{aligned}$$

On notera que le nombre de routes k est supposé connu. Or, lors de la présentation de l'algorithme de sélection des routes, à aucun moment il n'a été proposé de valeurs pour k. On peut en fait considérer que l'optimisation de cette valeur est faite en sélectionant a priori un grand nombre de routes. Si une route R_i est jugée inutile, un algorithme de répartition performant l'écartera en ne lui allouant aucune description (c'est-à-dire $N_i = 0$).
Des méthodes de résolution du problème non systématique dans un contexte de routes dijsointes sont proposées dans [eZJH04] et [eZJH01]. Il s'agit cependant de recherche de solutions approchées. En effet les auteurs choisissent de modéliser les variables aléatoires Y_i (indépendantes pour des routes dijsointes) par des lois normales de mêmes espérance et variance. Si cette approximation simplifie effectivement les calculs (une combinaison linéaire de variables aléatoires normales indépendantes suit encore une loi normale), elle introduit néanmoins un biais important. On décide donc de ne pas opérer une telle approximation.

IV.2 Calcul pratique de la fiabilité

On dispose d'une définition de la fiabilité. Il peut en outre s'avérer utile de connaître un algorithme permettant de calculer de manière pratique sa valeur. En effet, bien qu'il soit possible de trouver des algorithmes optimisant un critère sans pour autant le calculer explicitement, pouvoir le faire peut malgré tout s'avérer utile. Pouvoir calculer $\mathcal{R}$ pour un jeu de paramètres donné nous permet en effet d'utiliser un algorithme d'optimisation naïf dans un premier temps (celui consistant à calculer $\mathcal{R}$ pour un jeu de paramètres et à ne retenir que le cas le maximisant).
La fiabilité $\mathcal{R}$ d'un ensemble de routes $\mathcal{K} = (R_1, \cdots, R_k)$ joignant une source donnée `S` à une destination donnée `D`, est définie comme la probabilité que durant une période T entre deux mises à jour l'utilisation de ces routes garantissent la réception de l'information. La fiabilité est avant tout une information du point de vue de la source : quelle confiance accorder à telle méthode basée sur telle combinaison de paramètres, compte tenu de l'information que possède `S` ?

IV.2.a Interdépendance des routes

Ce calcul peut être complexe en considérant à la fois la dépendance de la fiabilité d'un lien au débit $p_{\mathsf{e}}^{tr} = f(\lambda_V) = 1 - \exp(-\Lambda/\lambda_V)$ et la possibilité pour plusieurs routes de se superposer localement. En effet, la disparition de certaines routes peut entrainer un meilleur comportement de certaines sections où il y avait normalement superposition. Si par exemple une route disparait suite à la disparition d'un lien, les liens suivants ont alors à prendre en charge un débit moindre, voire nul. Si, par ailleurs, d'autres routes les utilisent, elles disposent dès lors d'une bande passante supérieure. D'où une situation en apparence paradoxale : l'échec d'un lien peut améliorer la qualité du transfert. On notera que le problème ne se pose pas :

- pour des routes disjointes (les routes ne s'influencent pas mutuellement) ;
- pour une faible dépendance du lien au débit ($p_{\mathsf{e}}^{tr} \approx 1$, la réussite n'est alors due qu'à la survie des liens et des nœuds).

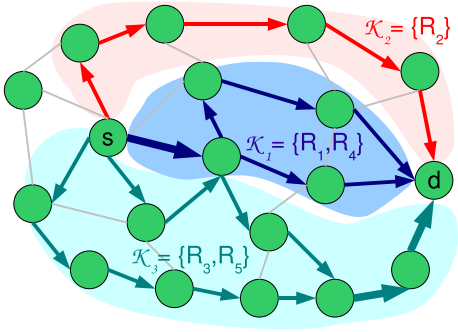


FIG. 4.13 – Découpage de l'ensemble des routes en sous-ensembles indépendants

Cette interdépendance complique le calcul de la fiabilité. Elle rend l'algorithme de calcul de $\mathcal{R}$ dépendant de l'ordre dans lequel les liens sont considérés. À l'inverse, une simplification est possible si l'ensemble des routes $\mathcal{K}$ est divisible en sous-ensembles indépendants $\mathcal{K}_1, \cdots, \mathcal{K}_{k'}$ (figure 4.13). Dans ce cas, pour chaque sous-ensemble $\mathcal{K}_i$ on peut calculer les quantités $\mathcal{E}(\mathcal{K}_i, \nu_i)$ correspondantes aux probabilités qu'exactement ν_i descriptions parmi celles circulant dans le sous-ensemble $\mathcal{K}_i$ arrivent à destination. La fiabilité totale est alors dans le cas non systématique :

$$\mathcal{R} = \sum_{\nu_1+\cdots+\nu_{k'} \in [M,N]} \mathcal{E}(\mathcal{K}_1, \nu_1) \times \cdots \times \mathcal{E}(\mathcal{K}_{k'}, \nu_{k'})$$

et dans le cas systématique :

$$\begin{aligned} \mathcal{R} \quad &= \textstyle\sum_{\nu_1+\cdots+\nu_{k'} \in [M,N]} \mathcal{E}(\mathcal{K}_1, \nu_1) \times \cdots \times \mathcal{E}(\mathcal{K}_{k'}, \nu_{k'}) \\ &+ \textstyle\sum_{\nu_1+\cdots+\nu_{k'} \in [1,M-1]} \sum_{\mu_1,\cdots,\mu_{k'}} \frac{\mu_1+\cdots+\mu_{k'}}{M} \mathcal{E}(\mathcal{K}_1, \nu_1, \mu_1) \times \cdots \times \mathcal{E}(\mathcal{K}_{k'}, \nu_{k'}, \mu_{k'}) \end{aligned}$$

avec $\mathcal{E}(\mathcal{K}_i, \nu_i, \mu_i)$ la probabilité qu'exactement ν_i descriptions et parmi celles-ci μ_i pseudo-descriptions parmi celles circulant dans le sous-ensemble $\mathcal{K}_i$ arrivent à destination.

IV.2.b Notion d'état

On appelle *état* des routes un événement au sens des probabilités pour lequel les routes ont une validité de fonctionnement définie. Un état revient ici à définir un k-uplet $\vec{s} = (s_1, \cdots, s_k) \in \{0,1\}^k$. Ainsi pour $\vec{s} = (1,0,1)$ l'état $\vec{Y} = \vec{s}$ correspond à l'événement où R_1 et R_3 fonctionne mais pas R_2. Les états sont munis d'une probabilité notée $\mathbb{P}(\vec{s}) = \mathbb{P}(\vec{Y} = \vec{s})$. La probabilité $\mathbb{P}(1,0,1) = \mathbb{P}(\vec{Y} = (1,0,1))$ est donc dans l'exemple celle que les routes R_1 et R_3 fonctionnent mais pas R_2. Pour k routes un ensemble de 2^k états sont envisageables. La somme de leur probabilité doit bien évidement égaler 1.

IV.2.c Construction progressive des routes

Dans l'algorithme de calcul de $\mathcal{R}$, les routes sont considérées comme initialement de longueur nulle et ayant une fiabilité globale égale à 1. On stocke les valeurs de toutes les probabilités $\mathbb{P}(\vec{s})$. Initialement on a donc $\mathbb{P}(1,\cdots,1) = 1$ et $\mathbb{P}(\vec{s}) = 0$ pour tout $\vec{s} \neq (1,\cdots,1)$. On considère ensuite un à un chaque nœud `V` (à l'exception de `S` et `D`) et chaque lien e appartenant à au moins une route. L'idée est d'adopter un

processus constructif des routes : on ajoute les éléments o (nœuds ou liens) un à un en mettant à jour les probabilités de chaque état $\mathbb{P}(\vec{s})$.

Il convient de constater que l'ensemble de ces éléments peut en général être muni d'une relation d'ordre partiel (le cas contraire est improbable et à rejeter : il signifierait que certaines routes ont des liens communs qu'elles emploient dans des sens contraires). Un élément o_1 (nœud ou lien) est inférieur à un élément o_2 si au moins une route passe par les deux et si o_1 est plus proche de la source que o_2. On considére les éléments des routes en respectant l'ordre ainsi défini : un élément ne peut être pris en compte que si l'ensemble des éléments inférieurs à lui l'a déjà été. On définit la fiabilité de l'élément o comme la probabilité qu'il fonctionne dans un contexte donné. C'est à dire :

$$\begin{aligned} p_o(\lambda_o) &= p_{\mathsf{V}}^{sv} \cdot p_{\mathsf{V}}^{tr}(\lambda_V) \cdot p_{\mathsf{V}}^{col}(\lambda_V) \text{ si } o = V \in \mathcal{V} \\ &= p_{\mathsf{e}} \text{ si } o = \mathsf{e} \in \mathcal{E} \end{aligned}$$

On notera que d'Après la modélisation précédemment choisie, la fiabilité d'un lien est une probabilité indépendante de l'utilisation qui ne modélise que la capacité du lien à exister. La fiabilité d'un nœud est par contre liée au débit reçu (λ_V) puisque des pertes sont plus probables si la mémoire du nœud ne peut gérer le débit important (pris en compte avec $p_V^{tr}(\lambda_V)$) ou si les collisions sont trop fréquentes (pris en compte avec $p_V^{col}(\lambda_V)$).

IV.2.d Mise à jour des probabilités

Afin de définir les mises à jour des probabilités, on introduit pour chaque élément o une relation d'équivalence sur les états notés $\sim_o$ définis par :

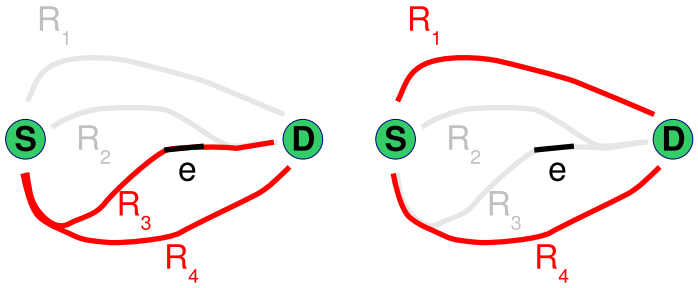
$$\vec{s'} \sim_o \vec{s} \Longleftrightarrow (\forall i, R_i \text{ ne passe pas par } o \Rightarrow s'_i = s_i)$$

Cette relation traduit le fait que deux états $\vec{s}$ et $\vec{s'}$ correspondent à la même validité pour les routes ne passant pas par o. Autrement dit si $\vec{s} \sim_o \vec{s'}$, la situation où o ne fonctionne pas rend $\vec{s}$ et $\vec{s'}$ indiscernables une fois que les routes ont été augmentées de l'élément o.

Pour un élément o les états possibles des routes $\vec{s}$ se répartissent en deux catégories :

- Les états *dépendants* de o requièrent la validité d'au moins une route passant par o. Leur probabilité est mise à jour par $\mathbb{P}(\vec{s})^{nouv} = p_o(\lambda_{\{o,\vec{s}\}}) \cdot \mathbb{P}(\vec{s})$ où $\lambda_{\{o,\vec{s}\}}$ est le débit traversant o si l'état des routes est $\vec{s}$. La probabilitité d'un tel état diminue, ce qui est normal : il faut que o fonctionne pour que l'état considéré $\vec{s}$ se produise (voir figure 4.14 (a)).
- Les états *indépendants* de o ne requièrent aucune des routes passant par o. Leur probabilité est mise à jour par $\mathbb{P}(\vec{s})^{nouv} = \mathbb{P}(\vec{s}) + \sum_{s' \sim_o s \wedge s' \neq s} (1 - p_o(\lambda_{\{o,\vec{s'}\}})) \cdot \mathbb{P}(\vec{s'})$. La probabilité d'un tel état augmente : pour que l'état $\vec{s}$ se produise il faut soit qu'il se produise sur les portions de routes ne prenant pas encore l'élément o en compte (et dès lors peu importe que l'élément o fonctionne ou non), soit que o soit invalide et qu'un ou plusieurs autres états $\vec{s'}$ se réduisent à $\vec{s}$ du fait de cette invalidité (voir figure 4.14 (b)).

Après prise en considération de tous les éléments constituant $\mathcal{K}$, on dispose très exactement de la probabilité de chaque état $\vec{s}$ pour des routes complètes.



(a) $\vec{s} = (0,0,1,1)$ est dépendant de e : il faut que e fonctionne pour que l'état $\vec{s}$ se produise

(b) $\vec{s'} = (1,0,0,1)$ est indépendant de e : le bon fonctionnement de e n'est pas requis pour que $\vec{s'}$ se produise

FIG. 4.14 – Pour $\mathcal{K} = (R_1, R_2, R_3, R_4)$ et un lien e, exemple de différence entre états dépendants et indépendants de e

IV.2.e Exemples de mises à jour

Si l'on considère 4 routes (R_1, R_2, R_3, R_4) et un élément o appartenant à la route R_1 seule, alors pour chaque $\vec{s}$:

– Soit $\vec{s} = (1, s_2, s_3, s_4)$ (état dépendant de o). On procède alors à la mise à jour :

$$\mathbb{P}(1, s_2, s_3, s_4)^{nouv} = p_o(\lambda_{\{o,(1,s_2,s_3,s_4)\}}) \cdot \mathbb{P}(1, s_2, s_3, s_4)$$

– Soit $\vec{s} = (0, s_2, s_3, s_4)$ (état indépendant de o). On procède alors à la mise à jour :

$$\mathbb{P}(0, s_2, s_3, s_4)^{nouv} = \mathbb{P}(0, s_2, s_3, s_4) + (1 - p_o(\lambda_{\{o,(1,s_2,s_3,s_4)\}})) \cdot \mathbb{P}(1, s_2, s_3, s_4)$$

Si l'on considère maintenant l'élément o appartenant à R_2 et R_3 et elles seules, alors pour chaque $\vec{s}$:

– Soit $\vec{s} = (s_1, 1, 1, s_4)$ (état dépendant de o). On procède alors à la mise à jour :

$$\mathbb{P}(s_1, 1, 1, s_4)^{nouv} = p_o(\lambda_{\{o,(s_1,1,1,s_4)\}}) \cdot \mathbb{P}(s_1, 1, 1, s_4)$$

– Soit $\vec{s} = (s_1, 0, 1, s_4)$ (état dépendant de o). On procède alors à la mise à jour :

$$\mathbb{P}(s_1, 0, 1, s_4)^{nouv} = p_o(\lambda_{\{o,(s_1,0,1,s_4)\}}) \cdot \mathbb{P}(s_1, 0, 1, s_4)$$

– Soit $\vec{s} = (s_1, 1, 0, s_4)$ (état dépendant de o). On procède alors à la mise à jour :

$$\mathbb{P}(s_1, 1, 0, s_4)^{nouv} = p_o(\lambda_{\{o,(s_1,1,0,s_4)\}}) \cdot \mathbb{P}(s_1, 1, 0, s_4)$$

– Soit $\vec{s} = (s_1, 0, 0, s_4)$ (état indépendant de o). On procède alors à la mise à jour :

$$\begin{aligned}
\mathbb{P}(s_1, 0, 0, s_4)^{nouv} &= \mathbb{P}(s_1, 0, 0, s_4) \\
&+ (1 - p_o(\lambda_{\{o,(s_1,1,1,s_4)\}})) \cdot \mathbb{P}(s_1, 1, 1, s_4) \\
&+ (1 - p_o(\lambda_{\{o,(s_1,0,1,s_4)\}})) \cdot \mathbb{P}(s_1, 0, 1, s_4) \\
&+ (1 - p_o(\lambda_{\{o,(s_1,1,0,s_4)\}})) \cdot \mathbb{P}(s_1, 1, 0, s_4)
\end{aligned}$$

IV.2.f Calcul final

Connaissant la valeur finale du vecteur $\{\mathbb{P}(\vec{s}),\ \vec{s} \in \{0,1\}^k\}$, il est possible de reconstituer la fiabilité par :
– dans le cas non systématique :

$$\mathcal{R} = \sum_{\vec{s}} \mathbb{P}(\vec{Y} = \vec{s}) \cdot \mathbf{1}_{[M,N]}(\vec{s} \cdot \vec{N})$$

– dans le cas systématique par :

$$\mathcal{R} = \sum_{\vec{s}} \mathbb{P}(\vec{Y} = \vec{s}) \cdot \left(\mathbf{1}_{[M,N]}(\vec{s} \cdot \vec{N}) + \frac{\vec{s} \cdot \vec{M}}{M} \mathbf{1}_{[1,M-1]}(\vec{s} \cdot \vec{N}) \right)$$

Nous savons donc résoudre algorithmiquement le problème dans un sens : calculer de manière pragmatique la fiabilité à partir de valeurs de $\vec{N}$ et M (ou $\vec{M}$) données. Le problème d'optimisation est cependant le problème inverse : trouver $\vec{N}$ et M (ou $\vec{M}$) pour que $\mathcal{R}$ soit maximale. On peut maintenant essayer de procéder à cette inversion en gardant à l'esprit qu'un algorithme naïf est désormais possible : tester pour différentes valeurs de $\vec{N}$ et de M et ne conserver que la meilleure proposition.

IV.3 Problème simplifié

Afin d'essayer de dégager une solution de calcul de l'optimal on restreint dans un premier temps les cas possibles en simplifiant notre problème. Si cette recherche s'avère payante on tentera de la généraliser dans un second temps. On admet donc que :
– les routes sont disjointes ;
– la dépendance des routes au trafic est négligeable ;
– le nombre total de descriptions N et le nombre de descriptions nécessaires à la reconstruction M sont fixés ;
– on se place dans un cas non systématique.

Dans ce contexte, on peut considérer les variables Y_i indépendantes entre elles et indépendantes des quantités N_i correspondantes au nombre de descriptions allouées sur R_i. Elles sont donc entièrement déterminées par une probabilité de validité p_i. Le vecteur $\vec{p} = (p_1, \cdots, p_k)$ est donc un paramètre du problème. La fiabilité se calcule alors comme :

$$\begin{aligned} \mathcal{R}(\vec{p}, \vec{N}, M) &= \textstyle\sum_{\vec{s} \in \{0,1\}^k \ : \vec{s} \cdot \vec{N} \geq M} \mathbb{P}(\vec{Y} = \vec{s}) \\ &= \textstyle\sum_{\vec{s} \in \{0,1\}^k} \mathbf{H}(\vec{s} \cdot \vec{N} - M) \mathbb{P}(\vec{Y} = \vec{s}) \end{aligned}$$

où $\mathbf{H}$ est la fonction de Heaviside (nulle sur les quantités strictement négatives, égale à 1 sinon). Les hypothéses formulées précédement permettent d'affirmer que $\mathbb{P}(\vec{Y} = \vec{s}) = \prod_{i=1}^{k} p_i^{s_i} (1 - p_i)^{1 - s_i}$. D'où une fiabilité :

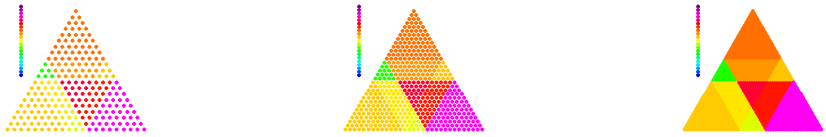
$$\mathcal{R}(\vec{p}, \vec{N}, M) = \sum_{\vec{s} \in \{0,1\}^k} \mathbf{H}(\vec{s} \cdot \vec{N} - M) \prod_{i=1}^{k} p_i^{s_i} (1 - p_i)^{1 - s_i}$$

On recherche donc la répartition $\vec{N}^*$ sur k routes qui maximise la probabilité de reconstruction :

$$\begin{aligned}\vec{N}^* &= J(\vec{p}, N, M) \\ &= \arg\max_{\vec{N},\, \sum_i N_i = N} \mathcal{R}(\vec{p}, \vec{N}, M) \\ &= \arg\max_{\vec{N},\, \sum_i N_i = N} \left(\sum_{\vec{s} \in \{0,1\}^k} \mathbf{H}(\vec{s} \cdot \vec{N} - M) \prod_{i=1}^{k} p_i^{s_i} (1 - p_i)^{1 - s_i} \right)\end{aligned}$$

IV.4 L'espace des répartitions

Les répartitions possibles se présentent sous la forme d'un n-uplet d'entiers $\vec{N} = (N_1, \cdots, N_k)$ vérifiant $\sum_i N_i = N$ et $N_i \geq 0$. Autrement dit l'ensemble des répartitions est une version discrète d'un $(k-1)$-simplexe (un segment si $k = 2$, un triangle si $k = 3$, un tétrahèdre si $k = 4$, ...). Ce simplexe sera noté $\mathcal{T}_k$. Si pour un ensemble de routes $\mathcal{K}$ donné, on affiche la fiabilité de l'ensemble des répartitions possibles pour différents nombres N de descriptions et différents nombres M de descriptions suffisantes comme sur la figure 4.15, il devient évident que le comportement de $\mathcal{R}$ est similaire lorsque le rapport M/N est constant.



(a) $N = 20$, $M = 12$ (b) $N = 30$, $M = 18$ (c) $N = 60$, $M = 36$

FIG. 4.15 – Fiabilités de toutes les répartitions sur l'espace $\mathcal{T}_3$ avec $\vec{p} = (0.87, 0.64, 0.55)$ et un rapport constant $\rho = M/N = 3/5$

Nous pouvons donc nous limiter à considérer le ratio $\rho = M/N$ et le vecteur unitaire $\vec{\theta} = \vec{N}/N$. Le problème revient alors à rechercher :

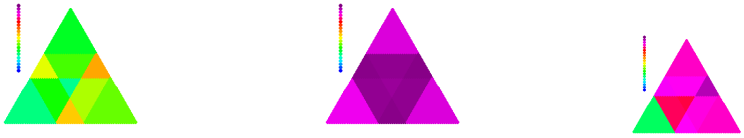
$$\begin{aligned}\vec{\theta}^* &= J(\vec{p}, \rho) \\ &= \arg\max_{\vec{\theta} \,:\, \sum_i \theta_i = 1} \left(\sum_{\vec{s} \in \{0,1\}^k} \mathbf{H}(\vec{s} \cdot \vec{\theta} - \rho) \cdot \prod_{i=1}^{k} p_i^{s_i} (1 - p_i)^{1 - s_i} \right)\end{aligned}$$

IV.4.a Sous espace de $\mathcal{T}_k$

Pour ρ et $\vec{p}$ fixés, $\mathcal{R}$ n'est pas une fonction très complexe : elle ne prend qu'un faible nombre de valeurs lorsque $\vec{\theta}$ varie dans $\mathcal{T}_k$. De plus, les sous-ensembles de $\mathcal{T}_k$ où $\mathcal{R}$ est constant ont des formes simples et symétriques. Leur nombre précis et leur forme dépendent uniquement de $\vec{p}$ (voir les zones unicolores dans les figures 4.16 et 4.17).

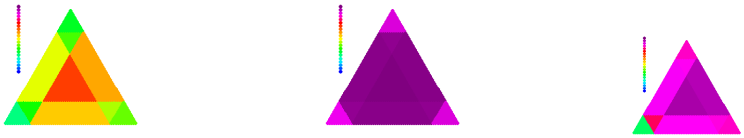
IV.4.b Vecteurs de répartitions équivalents

Deux vecteurs appartenant à un même sous-espace de $\mathcal{T}_k$ (dans une même zone de couleur sur les figures) sont dit équivalents. Ceci implique qu'il n'existe pas nécessairement une seule solution mais bien un



(a) $\vec{p_a}$ (b) $\vec{p_b}$ (c) $\vec{p_c}$

FIG. 4.16 – Fiabilité pour trois vecteurs $\vec{p}$ différents avec $\rho = 0.4$



(a) $\vec{p_a}$ (b) $\vec{p_b}$ (c) $\vec{p_c}$

FIG. 4.17 – Fiabilité les mêmes $\vec{p}$ avec cette fois $\rho = 0.2$

ensemble de solutions. Un algorithme simple de recherche de répartition optimal peut donc se limiter à comparer un point de chaque sous-ensemble de $\mathcal{T}_k$. Il convient alors de ne conserver que le vecteur de meilleure fiabilité (voir figure 4.18).

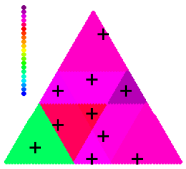


FIG. 4.18 – Sélection d'un vecteur dans chaque sous-ensemble

On peut également prendre en compte le fait que si $\vec{p}$ est tel que $p_1 \leq \cdots \leq p_k$, le sous-ensemble optimal contient nécessairement un vecteur θ tel que $\theta_1 \leq \cdots \leq \theta_k$ (voir 4.19).
Le principal problème reste néanmoins de parvenir à délimiter les sous-ensembles. Cela se fait facilement pour 3 routes. Considérer le cas à 4 routes montre que le problème gagne en complexité lorsque le nombre de routes augmente. Dès lors, comment généraliser à k routes ?

IV.5 Proposition d'heuristiques

Nous proposons une heuristique afin de simplifier notre recherche d'optimalité. On suppose que pour chaque sous-ensemble il existe un vecteur de répartition sous la forme $\vec{\theta} = \left(\frac{\nu_1}{\delta}, \frac{\nu_2}{\delta}, \cdots, \frac{\nu_k}{\delta}\right)$ tel que le

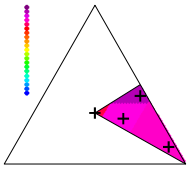


FIG. 4.19 – Sélection d'un vecteur dans chaque sous-ensemble avec valeurs ordonnées

dénominateur commun δ reste petit.

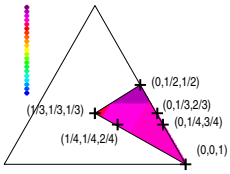


FIG. 4.20 – Sélection d'un vecteur de fraction de dénominateur au plus égal à 4

IV.5.a Dénominateurs

Dans ce contexte, un algorithme naïf peut consister à tester la fiabilité de tous les vecteurs de répartition du type $\left(\frac{\nu_1}{\delta}, \frac{\nu_2}{\delta}, \cdots, \frac{\nu_k}{\delta}\right)$ avec $\delta \leq d_{max,k}$ et $\nu_1 \leq \cdots \leq \nu_k \leq \delta$. Ici $d_{max,k}$ est le dénominateur maximum à prendre en compte afin de garantir que la répartition optimale soit testée. La valeur de $d_{max,k}$ reste à définir.

Pour plusieurs valeurs de k (de 1 à 6) et ρ (de $0,001$ à $1,000$), tirons au hasard différentes valeurs de $\vec{p}$ (environ 250). L'algorithme naïf nous fournit un vecteur $\vec{\theta}$ optimal pour les paramètres k, ρ et $\vec{p}$ du problème. Bien sûr, étant donné que plusieurs solutions coexistent, le résultat $\vec{\theta}$ dépend grandement de l'ordre de test des vecteurs solutions. On choisit de les parcourir par dénominateur δ croissant et des numérateurs évoluant dans un ordre prédéfini.

La courbe 4.21 montre que les dénominateurs possèdent une certaine régularité : ils sont répartis sur des fonctions de ρ de la forme $\rho \mapsto \lfloor \frac{\omega}{\rho} \rfloor$ où $\omega \in \mathbb{N}^*$. Ces fonctions sont indépendantes du nombre de routes k. Si nous considérons par exemple une valeur $\rho = 0.55$ nous nous apercevons qu'aucune fonction ne passe par la valeur 4. Ceci n'implique pas nécessairement qu'il n'existe aucune solution de la forme $\left(\frac{\nu_1}{4}, \frac{\nu_2}{4}, \cdots, \frac{\nu_k}{4}\right)$ pour un tel ρ. Cela signifie simplement que si une telle solution existe elle est équivalente à une autre de dénominateur moindre (pour l'exemple 1 ou 3).

IV.5.b Numérateurs

En ce qui concerne les numérateurs la figure 4.22 montre pour $k = 6$ comment sont répartis les 6-uplets de numérateurs trouvés comme solution par l'algorithme naïf. On peut déjà s'apercevoir que peu de

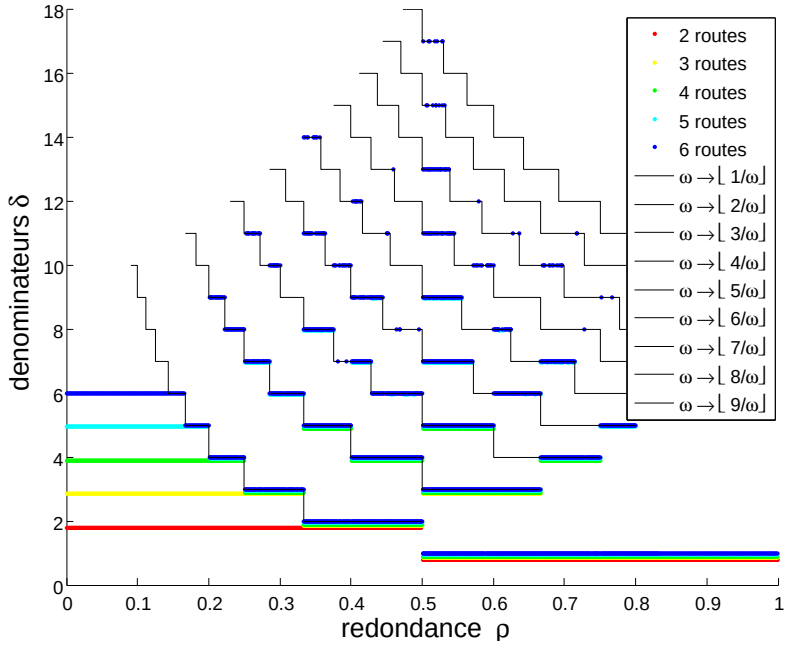


FIG. 4.21 – Différentes valeurs possibles pour le dénominateur en fonction de ρ

répartitions apparaissent comme solution parmi l'ensemble des combinaisons théoriquement possibles. Ceci est une bonne nouvelle : il est donc a priori suffisant de comparer un nombre faible de combinaisons afin de déduire l'optimal.

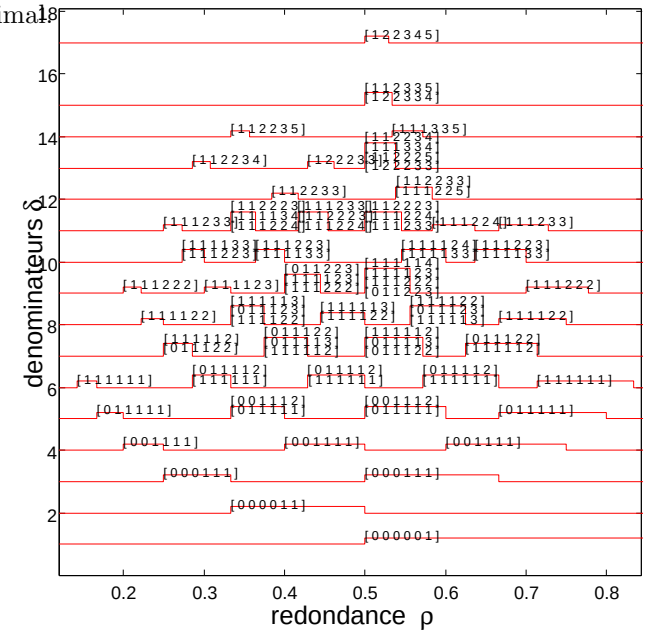


FIG. 4.22 – Différentes valeurs possibles pour les numérateurs en fonction de ρ et pour $k = 6$

Il paraît en revanche difficile d'extraire une règle générale concernant la forme de ces combinaisons.

IV.6 Conclusion

Bien que nous nous soyons limité pour cette étude à un cas très particulier de notre problème d'optimisation, il reste très difficile d'obtenir un algorithme très performant (c'est-à-dire réduisant suffisamment le nombre d'opérations faites par un algorithme naïf). Parce que la valeur de $\mathcal{R}$ n'est pas continue en ses arguments, de légères modifications provoquent des variations brusques de fiabilité. Même si l'on a restreint le champ de test des paramètres optimaux possibles, on est loin de savoir résoudre ce problème, pourtant très simplifié. On se résout donc à abandonner l'optique d'une solution mathématique et à se concentrer plutôt sur l'obtention de régles d'utilisation pratique issues d'analyses de simulations.

V Simulations

Bien que nous possédions désormais un mécanisme de choix de routes, la répartition de l'information sur ces routes reste, malgré les pistes étudiées ci-dessus, un problème ouvert. On se propose donc de simuler le réseau avec la modélisation décrite précédemment. Le but est d'étudier l'impact de $\vec{M}$, $\vec{N}$, f_p et f_e sur la qualité du transfert considéré. Les résultats de ces simulations ont donnée lieu à la publication [eSH07].

V.1 Spécification des tests

On se place dans un cadre de test correspondant à une vision statique du réseau du point de vue d'une source ; autrement dit, durant une période T pendant laquelle la source `S` ne reçoit pas de nouvelles informations à propos du réseau. Dans un souci de simplicité on ne prend en compte que des valeurs $N_i \in \{0,1\}$ et $M_i \in \{0,1\}$. Autrement dit, chaque route ne véhicule au plus qu'une description issue d'un même codage. Pour $k = 6$ routes trouvées, on compare ainsi les différentes valeurs de $N \in [1,k]$ en considérant que les N premières routes sont toujours utilisés. Par ailleurs les valeurs de $M \in [1,N]$ sont testées individuellement.

- Dans le cas non systématique, M correspond simplement à un certain niveau de redondance.
- Dans le cas systématique, les M premières descriptions sont considérées comme des pseudo-descriptions (c'est-à-dire des messages originaux) et constituent donc des données exploitables individuellement en cas de réception.

Pour chaque scénario de simulation, les étapes suivantes sont opérées dans les deux cas (non systématique et systématique) :

- un graphe est généré en répartissant des nœuds sur une aire de forme carrée ;
- on crée des liens en considérant une même portée pour tous les nœuds ; une fiabilité est accordé à chaque lien ;
- la source et la destination d'un transfert de débit λ sont aléatoirement choisies ;

– pour différentes valeurs de f_p et f_e un ensemble de 6 routes est déterminé entre la source et la destination ;
– pour différentes valeurs de N et M on calcule la fiabilité globale du jeu de route $\mathcal{R}$, autrement dit la probabilité que l'information originale puisse être reconstruite.

La fiabilité effective $p_{\mathtt{e}}$ de chaque lien e est définie comme la probabilité de survie du lien. On la suppose connue de la source grâce aux messages TC qu'elle a reçu précédement. Elle permet de définir le coût $\mathrm{cout}[\mathtt{e}] = -\log(p_{\mathtt{e}})$ qui permet à l'algorithme de sélection des routes de s'exécuter. La survie des nœuds est définie par les probabilité p_V^{sv} qu'on suppose égales à 1. Autrement dit, on suppose que les nœuds ont peu de chance de disparaître.

Les probabilités de résistance au trafic et aux collisions (qui sont toutes les deux dépendantes du débit) sont modélisés par une fonction $p_V^{res}(\lambda_V) = p_{\mathtt{V}}^{tr}(\lambda_V) \cdot p_{\mathtt{V}}^{col}(\lambda_V) = 1 - \exp(-\Lambda/\lambda_V)$ avec Λ une constante et λ_V le débit reçu par le nœud `V`. Dans le cadre de simulation présent, chaque route prise en considération transporte exactement une description (si elle n'en porte aucune, elle peut en effet être mise de côté). λ_V peut alors s'écrire comme $\lambda_V = \lambda U_V/M$ avec U_V le nombre de routes passant par `V`. La formule définissant p_V^{res} garantit que sa valeur est proche de 1 tant que le débit n'est pas trop élevé puis s'effondre vers 0 au-delà d'un certain seuil. On fera l'hypothèse que l'état de chaque lien (valide ou non) est constant sur la période T durant laquelle la topologie perçue par `S` n'est pas mise à jour : soit le nœud ou le lien laisse passer tous les paquets qu'il reçoit, soit il les jette tous. La table 4.1 référence les valeurs des différents paramètres de simulation.

Les résultats des 100 simulations effectuées pour chaque jeu de paramètres considéré sont moyennés afin d'obtenir des courbes lissées.

V.2 Résultats

De manière très générale on observe une amélioration de la fiabilité avec d'une part l'augmentation de la qualité des liens du réseau, d'autre part, l'augmentation de sa taille, les autres paramètres étant fixés. Voir à cet effet les figures 4.23. On y constate une meilleure fiabilité globale pour tout N et M et toute fonction de coût.

Il apparaît par ailleurs que le choix des fonctions d'incrémentation f_p et f_e n'a qu'un impact secondaire sur la fiabilité des routes. Même s'il semble que certains choix pour ces fonctions améliorent sensiblement la fiabilité, les variations des autres paramètres ont une incidence nettement plus perceptible sur la qualité du transfert.

V.2.a Cas non systématique

Pour un débit faible, le comportement général est que l'augmentation de M influe négativement sur la fiabilité. Autrement dit, pour un nombre de route donnée, mieux vaut reproduire sur chaque route le même message que chercher à en répartir des morceaux. À l'inverse, à M fixé, $\mathcal{R}$ croît en général avec N. On préférera donc, en ayant imposé un nombre de routes nécessaire à la reconstruction, choisir un

Portée des nœuds	200 m
Dimension de l'air de simulation	1000 m × 1000 m
Λ	60 Mb/s
Nombre de nœuds	30, 60, 120, 180 ou 240
Probabilité de survie p_{e}^{sv} des liens	$[0.9, 1]$, $[0.7, 1]$ or $[0.5, 1]$
Débit à la source λ	3 Mb/s, 6 Mb/s, 10 Mb/s, 15 Mb/s, 22.5 Mb/s ou 30 Mb/s
f_p	$f_p(c) = c + c_0$ $c_0 \in \{0.1, 0.3, 0.7, 1, 2, 5\}$
f_e	$id \leq f_e \leq f_p$
Nombre de routes N	2, 3, 4, 5 ou 6
Nombre de routes nécessaires à la reconstruction M	$1 \leq M \leq N$
Nombre de simulations pour chaque jeu de paramètres	100

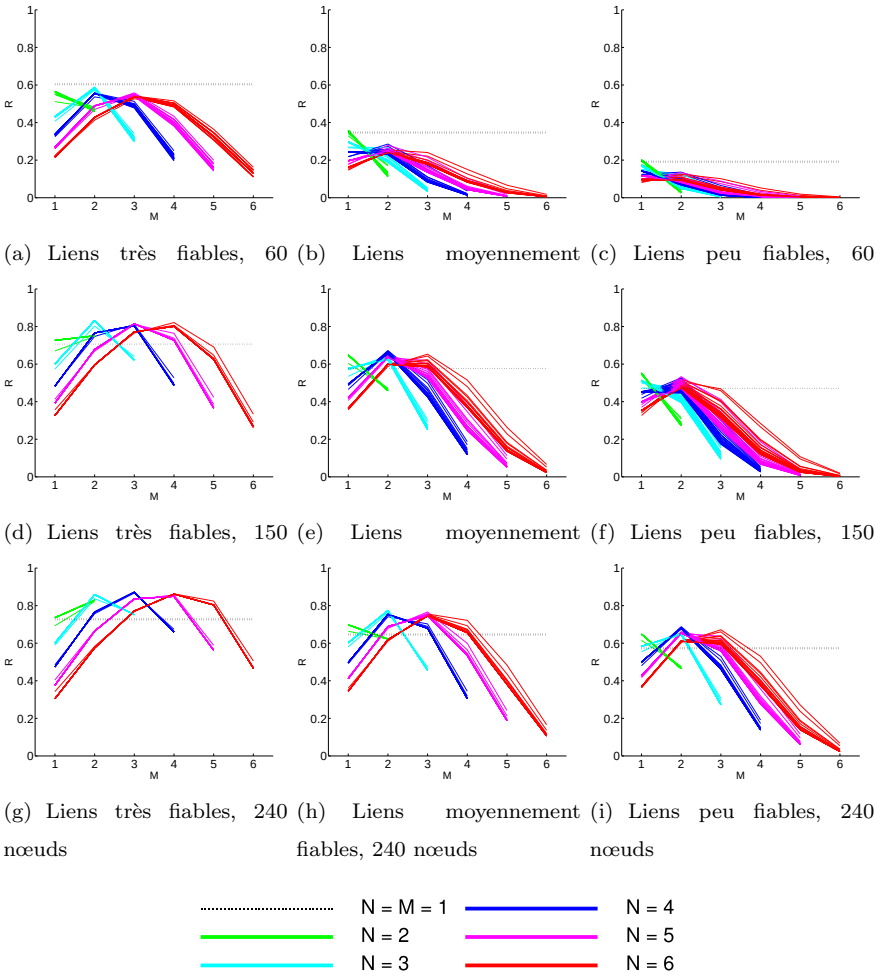
TAB. 4.1 – Paramètres utilisés dans les différents scénarios simulés

nombre total de route important.

Si les liens sont fiables ou si le réseau est grand, on remarque qu'avec $M = 1$ (cas correspondant à la duplication de l'information sur chaque route) le choix de $N = 6$ routes est sensiblement moins intéressant que $N = 5$. On peut supposer que dans ce cas les effets de la duplication commencent à se faire sentir même si le débit à la source est faible.

Si le débit augmente (comparer les figures de 4.23, 4.24 et 4.25) les courbes s'écroulent pour M faible. Autrement dit les cas de duplications ont de plus en plus de mal à supporter la quantité de données routées. On voit par ailleurs que cette chute est d'autant plus importante que N est grand. Les cas multichemins restent cependant au dessus du monochemin tant que M a une valeur moyenne (ni trop faible ni trop proche de N) et que le réseau est suffisamment grand ou que les liens sont fiables.

Avec un débit de grande valeur (voir figures 4.25) le phénomène précédent s'accentue. Ainsi les valeurs intermédiaires de M (aux alentours de $N/2$) donne la meilleure qualité, surtout lorsque le réseau et grand et possède des liens de bonnes qualités. Pour M petit, la fiabilité est faible et augmente à mesure que N diminue (autrement dit, si chaque route contient beaucoup d'information, voire une copie de l'information originale, autant en utiliser peu). Pour M grand, la fiabilité est également faible et, à l'inverse du cas précédent, augmente avec N. Dans le cas de réseaux à la fois petits et avec des liens faibles, la stratégie monochemin est définitivement la meilleure.



(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 60 nœuds (c) Liens peu fiables, 60 nœuds

(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 150 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

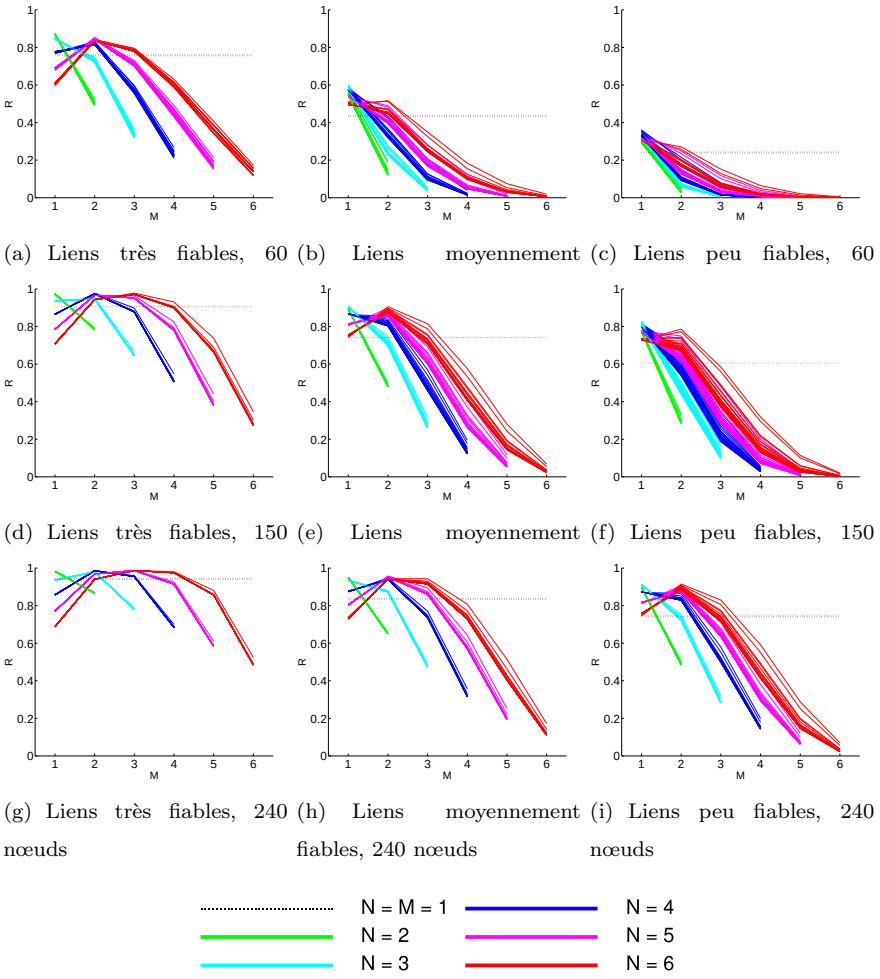
FIG. 4.23 – Variations globales de la fiabilité pour $\lambda = 15Mb/s$, cas non systématique

V.2.b Cas systématique

Lorsque le débit reste faible (figures 4.26), la fiabilité diminue avec l'augmentation de M, mais de manière beaucoup moins marquée que dans le cas non systématique. L'utilisation de routes nombreuses semble encore la meilleure option même si l'on remarque encore une fois un affaiblissement de la fiabilité pour 6 routes lorsque la redondance totale est forte (c'est-à-dire lorsque M est petit).

La diminution du débit (figures 4.27) voit le comportement de la fiabilité s'inverser pour les faibles valeurs de M : la restriction du nombre de routes est la meilleure stratégie. Par ailleurs l'utilisation d'une redondance globale forte devient de moins en moins justifiée. Pour de grande valeur de M, l'ordre précédent est conservé (un grand nombre de routes est préférable).

Les figures 4.28 montrent qu'à débit élevé, la stratégie $M = N$ (correspondant au *round-robin* sur un nombre variable de routes) est de loin la plus payante. Il est donc préférable de ne pas introduire de redondance et de simplement répartir les paquets sur les différentes routes.



(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 240 nœuds (c) Liens peu fiables, 60 nœuds

(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 240 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

FIG. 4.24 – Variations globales de la fiabilité pour $\lambda = 6Mb/s$, cas non systématique

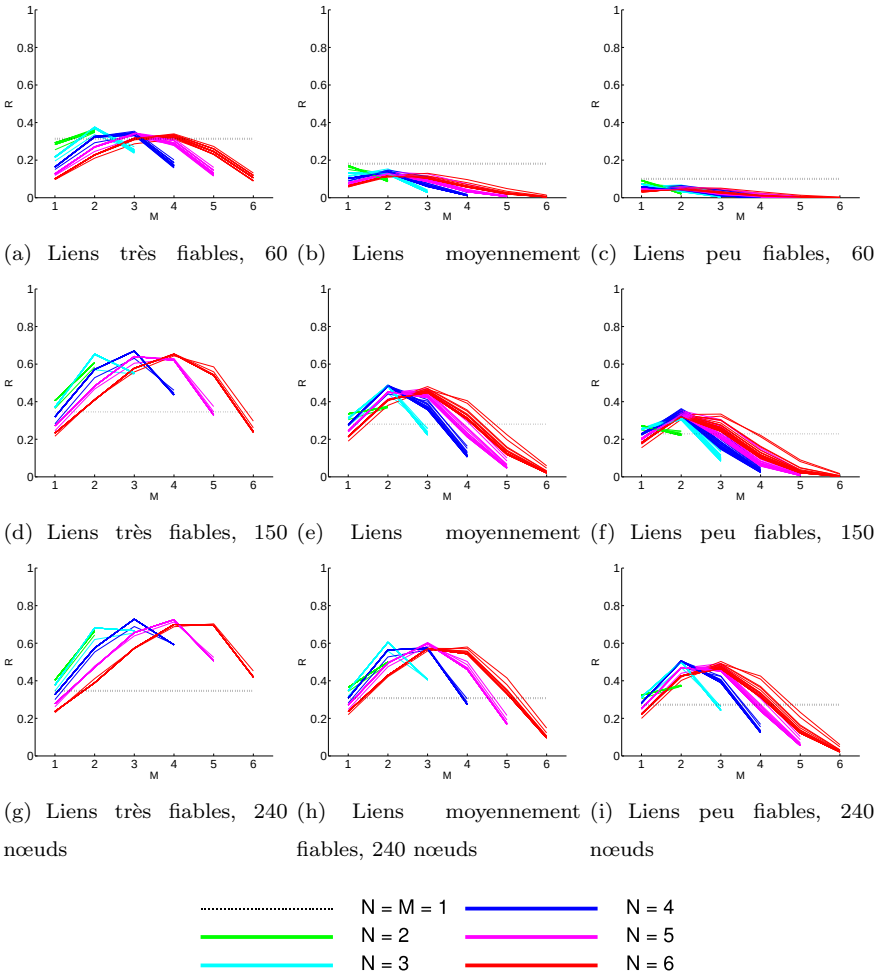
V.3 Analyse

De l'analyse de l'ensemble des courbes de fiabilité il semble que l'on puisse extraire un ensemble de règles à propos de l'impact des différents paramètres sur la qualité de la transmission. Comme attendu, l'augmentation de la taille du réseau, celle de la qualité des liens et la diminution du débit entraîne une amélioration globale de la fiablilité.

Lorsque le débit est faible, la duplication reste une bonne stratégie : le surplus de données n'entrainant pas d'engorgement, le réseau profite grandement de la liberté apportée par la présence de copies de paquets. La stratégie a bien entendu ses limites, et il n'est pas forcément intéressant de dupliquer au-delà de 4 ou 5 fois.

Plus le débit augmente, plus un codage à description multiple non dégénéré devient intéressant, à l'exception toutefois des cas où le réseau est trop petit. L'utilisation d'une seule route est alors préférable pour les réseaux de moins de 50 nœuds. Hormis ce cas particulier, les stratégies où M se situe aux alentour de $N/2$ s'avèrent payantes.

Enfin, l'utilisation d'un codage systématique et d'une faible redondance ($M = N$) est particulièrement adapté pour les très hauts débits. Il s'agit en fait d'une stratégie *round-robin* : les paquets sont simplement



(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 240 nœuds (c) Liens peu fiables, 60 nœuds

(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 240 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

FIG. 4.25 – Variations globales de la fiabilité pour $\lambda = 22.5Mb/s$, cas non systématique

répartis un à un sur les routes, de sorte qu'en moyenne chacune en prend autant en charge qu'une autre.

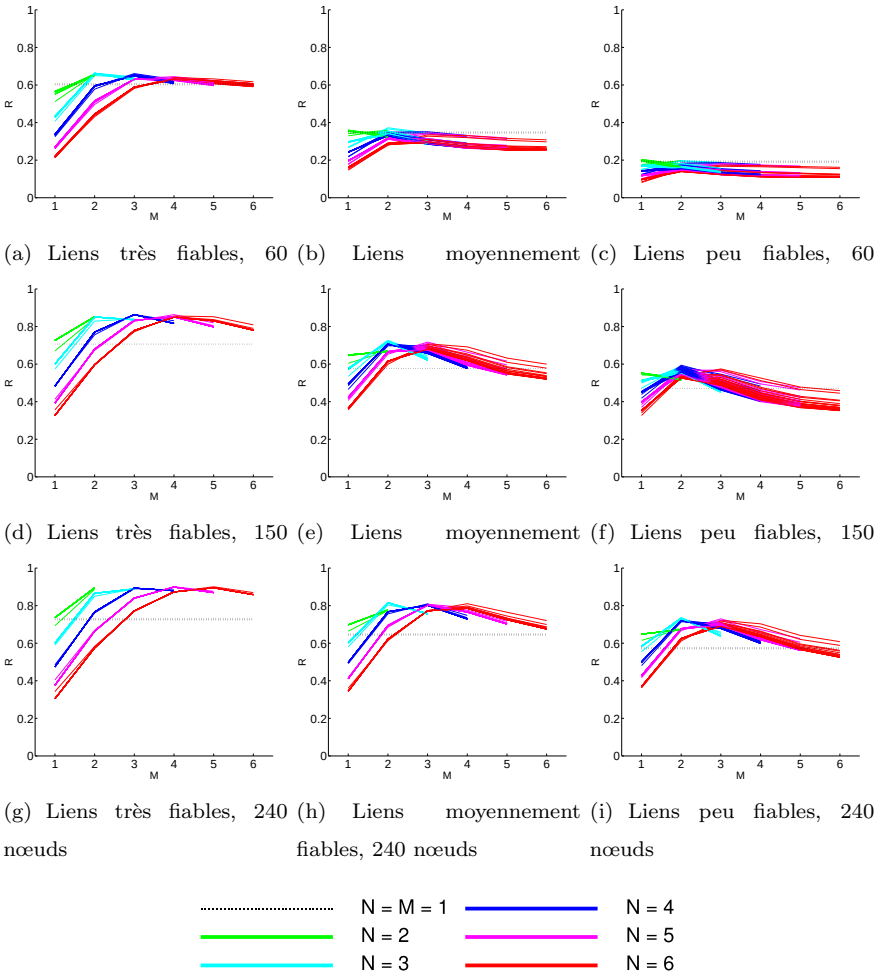
V.4 Conclusion

Une approche pratique basée sur des simulations s'est avérée intéressante. Plutôt que de chercher à déduire mathématiquement la meilleure répartition possible, la simulation de l'algorithme de choix des routes et des méthodes de répartitions utilisant des stratégies MDC a permis d'en apprendre d'avantage sur l'influence des règlages possibles. On sait donc un peu mieux :

– comment le jeu des routes et la redondance introduite influence conjointement la fiabilité ;

– dans quelle mesure cette approche est intéressante par rapport à un simple routage monochemin.

Il faut néanmoins se rappeler que la modélisation choisie ne prend en compte que partiellement les possibles interférences des couches basses du réseau, et considère ce dernier d'un point de vue plutôt statique. Lors du fonctionnement réel, l'évolution de la topologie va contraindre tout protocole à utiliser des messages de controles, lesquels augmentent la charge globale. Des simulations sur NS sont donc nécessaires.



(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 240 nœuds (c) Liens peu fiables, 60 nœuds

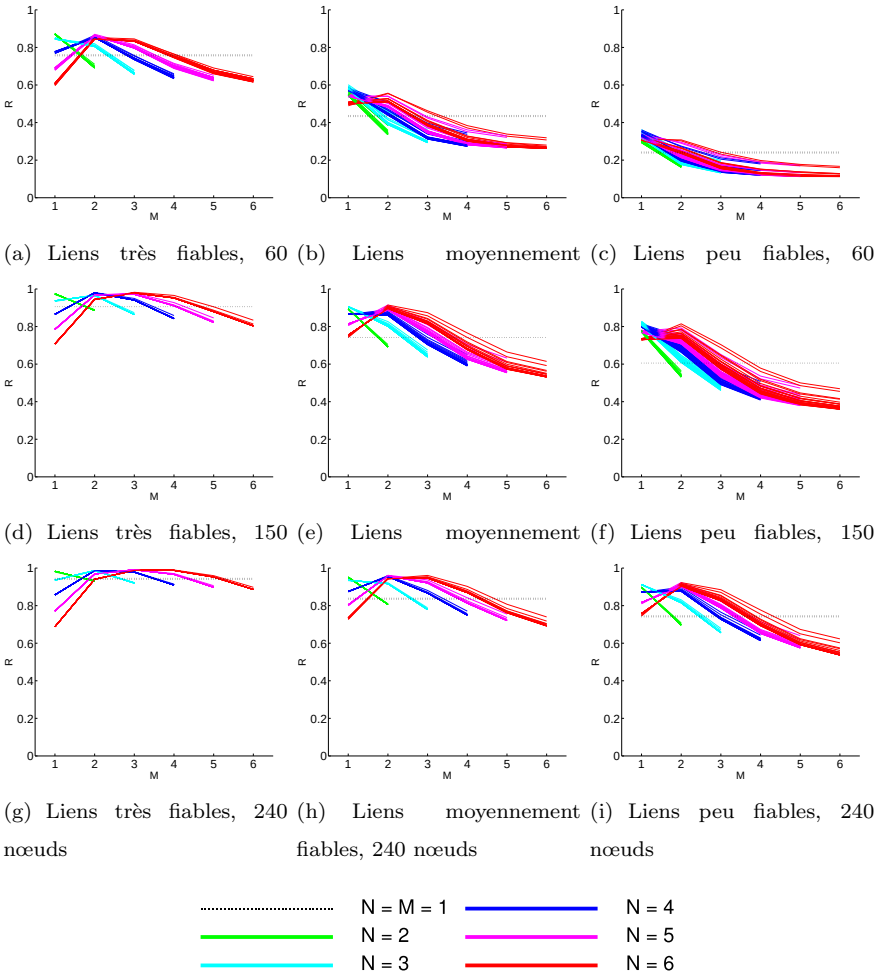
(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 240 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

FIG. 4.26 – Variations globales de la fiabilité pour $\lambda = 15Mb/s$, cas systématique

VI conclusion

Ce chapitre a eu pour objet l'étude de la mise en relation routage multichemins avec l'exploitation des routes via des techniques de codage utlisant les decriptions multiples. Partant du constat que l'utilisation de plusieurs routes pouvait, suivant la façon d'y envoyer les données, apporter une amélioration du routage suivant deux axes (à savoir la diminution du débit local et la diminution du risque lié à la perte d'une route), nous avons déterminé en quoi une approche à description multiple fournissait un compromis intéressant pour opérer ces améliorations. Cette mise en relation de deux domaines a priori distincts (le codage et le routage) nous a permis de repenser chacun d'entre eux en tenant compte de son intégration dans un mécanisme global. Il nous fallait ainsi pouvoir sélectionner des routes suffisamment (et non totalement) disjointes tout en restant acceptables du point de vue de leur longueur. Les algorithmes de routage connus dans la littérature n'apparaissant que partiellement adaptés, un nouvel algorithme de sélection de routes dans une topologie connue a été proposé. Son but est de fournir un nombre déterminé de routes, disjointes par les nœuds si possible, par les liens dans le cas contraire, ou même se superposant. Un critère de fiabilité a été défini afin de hiérarchiser les stratégies d'utilisation des routes : elle correspond à la proportion moyenne de paquets originaux reconstitués à destination. L'optimisation mathématique



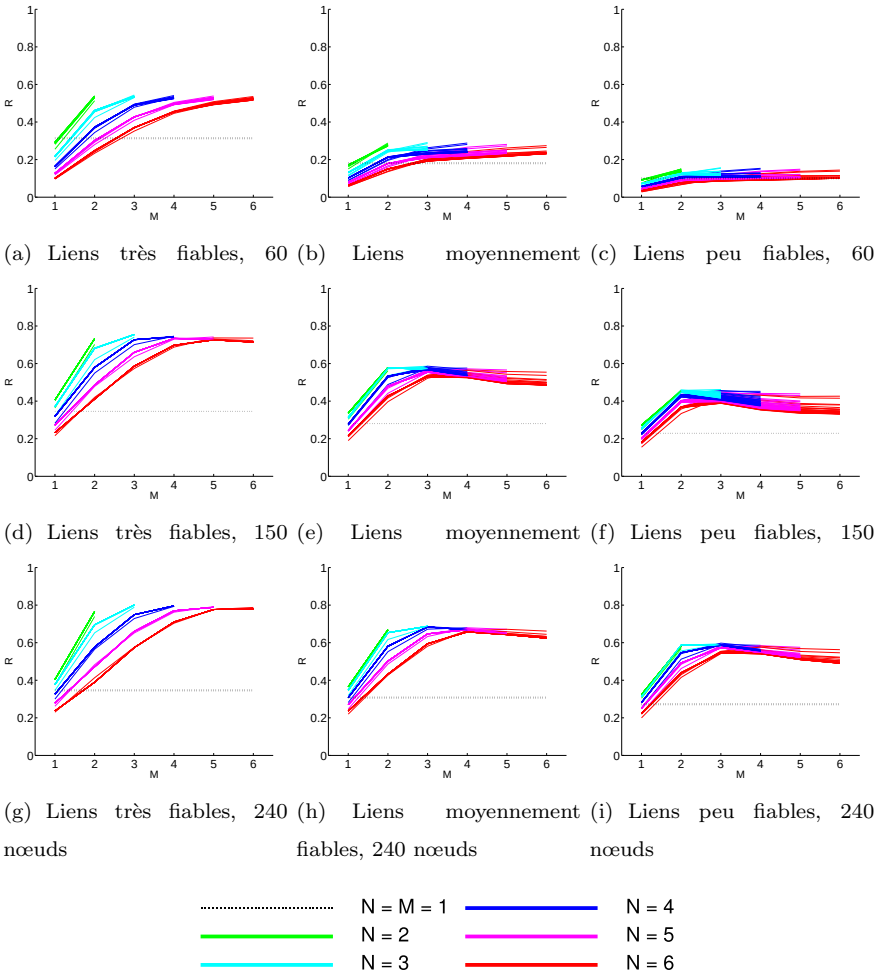
(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 60 nœuds (c) Liens peu fiables, 60 nœuds

(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 150 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

FIG. 4.27 – Variations globales de la fiabilité pour $\lambda = 6Mb/s$, cas systématique

de la fiabilité en fonction des paramètres MDC restant complexe, on s'est au final concentré sur des simulations afin de dégager des liens entre les caractéristiques du réseau et les stratégies de répartition. Afin de valider pleinement l'approche choisie ou au contraire d'en détecter les limites, on se propose dans le chapitre suivant d'élaborer un protocole de routage ad hoc multichemins basé sur les idées développées précédemment. Le but est d'effectuer des simulations sur NS2 afin de comparer les performances réalisées vis-à-vis des protocoles existants.



(a) Liens très fiables, 60 nœuds (b) Liens moyennement fiables, 60 nœuds (c) Liens peu fiables, 60 nœuds

(d) Liens très fiables, 150 nœuds (e) Liens moyennement fiables, 150 nœuds (f) Liens peu fiables, 150 nœuds

(g) Liens très fiables, 240 nœuds (h) Liens moyennement fiables, 240 nœuds (i) Liens peu fiables, 240 nœuds

FIG. 4.28 – Variations globales de la fiabilité pour $\lambda = 22.5Mb/s$, cas systématique

Chapitre 5

Le protocole MPOLSR

Le protocole OLSR fait partie des protocoles proactifs dont le fonctionnement requiert la mise à jour récurrente en chaque nœud du réseau d'un grand nombre d'informations. Ces dernières peuvent au final s'avèrer utiles ou non. On collecte donc une information très générale sur la topologie puis on en extrait l'information destinée à être exploitée. Pour tout autre nœud, considéré comme une destination potentielle, il convient de déterminer le voisin le plus adapté pour acheminer les paquets d'information jusqu'à la destination.
De part le fonctionnement de ce type de protocole, il est particulièrement pratique d'élaborer une version multichemin. Nous proposons ainsi dans ce chapitre une extension de OLSR baptisée MPOLSR (pour Multi-Path OLSR) dont le but est de mettre en application les idées développées dans le chapitre précédent. Sont d'abord exposés les détails de son fonctionnement et de son implémentation. Ensuite des tests réalisées sur NS-2 permettent de comparer ses performances avec celles de OLSR suivant les critères définis dans le chapitre 3.

I Le choix d'OLSR

Différents protocoles pourraient être adaptés afin de tester le fonctionnement des algorithmes de sélection des routes et de répartition de l'information. Il convient cependant de trouver un protocole où la source du transfert possède suffisamment d'information sur le réseau et sa topologie. Cette centralisation de l'information est fréquente dans les protocoles proactifs. En effet, le besoin de connaître à tout moment une façon satisfaisante de joindre n'importe quel nœud implique souvent l'existence d'un mécanisme de collecte de l'information topologique. Grâce à ce dernier tout nœud possède, à tout moment et indépendamment de la nécessité d'un transfert, une représentation en mémoire de la topologie globale du réseau (qui peut bien sûr être partielle et imprécise).
Parmi ces protocoles, OLSR (voir [ePMeTCeALeAQeLV01]) est de loin le plus connu et un de ceux réalisant les meilleures performances. Il est de plus, toutes catégories confondues, un des plus souvent cités avec DSR et AODV (voir entre autre [eTLeNHeBMeMD99] et [eFFeCB06]). Cette popularité est

notamment dû à la limitation de l'inondation réalisée par le mécanisme de relais (appelés MPR) ; cette inondation pénalisant généralement les autres protocoles proactifs. Nous allons donc nous attacher à modifier OLSR en convenant toutefois qu'il n'est qu'un choix parmi d'autres. Notre principal objectif reste en effet d'établir une différence de comportement entre la présence et l'absence des mécanismes de sélection des routes et de répartition de paquets pour un même protocole. Autrement dit, il s'agit d'évaluer si, en se basant sur le fonctionnement d'OLSR, on obtient une amélioration de certains critères d'évaluation du routage en utilisant, d'une part, plusieurs routes pour véhiculer l'information entre sources et destinations, d'autre part une redondance des données (selon un principe de codage à description multiple) sur ces routes.

II Spécifications de MPOLSR

MPOLSR étant construit à partir d'OLSR, on se propose de revenir avant tout sur le fonctionnement de ce dernier. Dans un second temps, on montre où et comment sont apportées les modifications nécessaires aux mécanismes propres à MPOLSR.

II.1 Rappel sur le fonctionnement d'OLSR

Le fonctionnement d'OLSR peut se découper selon les phases suivantes :
– l'utilisation répétitive de messages HELLO afin de mettre à jour les voisinages et les relais ;
– l'utilisation répétitive de messages TC afin de transmettre l'information de voisinage au reste du réseau ;
– la mise à jour en chaque nœud de sa table de topologie après réception des messages TC ;
– la mise à jour en chaque nœud de sa table de routage à partir de la table de topologie ;
– en cas de transfert, l'envoi des données vers le voisin jugé le plus adapté d'après la table de routage.
Les trois premières étapes traduisent le fonctionnement d'un protocole réactif, dans lequel on cherche constamment à mettre à jour la perception que les différents nœuds ont du réseau. Les messages HELLO sont délivrés à fréquence constante par chaque nœud. Le but pour un nœud V est de rappeler ou d'informer ses voisins de son existence. Par ailleurs en incluant également dans ses messages HELLO les adresses des nœuds qu'il considère déjà comme ses voisins, V informe également tout voisin W recevant le message HELLO si ce dernier est lui-même déjà perçu comme un voisin de V. Ce mécanisme garantit à OLSR que les liens effectivement pris en compte sont bidirectionnels. Si un nœud V répertorie un voisin W qui à partir d'un moment ne se fait plus entendre, un temporisateur assure que W finit par disparaître de la liste des voisins connus de V.
Les HELLO servent également à mettre en place le mécanisme de relais multipoints (MPR). En effet les messages HELLO permettent naturellement à chaque nœud de connaître non seulement son voisinage mais également son voisinage d'ordre 2. Il peut alors sélectionner dans son voisinage un sous-ensemble de nœuds qui seront appelés relais. Par définition l'union des voisinages de tous les relais d'un nœud V doit recouvrir son voisinage d'ordre 2.

Si une information transmise par V n'est retransmise que par ses relais, elle atteint malgré tout tous les voisins d'ordre 2 de V. Ce principe peut s'appliquer en cascade lorsque chacun des relais considère à son tour ses propres relais. En cas d'inondation, on limite donc grandement le nombre de retransmissions tout en garantissant que l'information est distribuée partout. Ce principe d'inondation limitée est destiné aux messages de contrôle de topologie (TC). Chaque nœud V en génère un à intervalle régulier. Le but d'un message TC est de renseigner les nœuds du réseau à propos du voisinage de V. Bien qu'il faille pour cela effectivement inonder le réseau (tout nœud doit recevoir au moins une version de chaque TC), les mécanismes de numéro de séquence (qui garantissent qu'une version d'un même TC ne peut être retransmise qu'au plus une fois) et de relais multipoints (qui sélectionnent les nœuds ayant le droit de retransmettre à leur tour les TC) limitent la charge totale due au seul routage.
L'information réunie en chaque nœud lui permet de tenir à jour une table de topologie (qui correspond au graphe de la topologie connue) et une table de routage (déduite de la précédente, elle répertorie les plus courts chemins jusqu'à n'importe quel nœud). Lorsqu'un paquet doit être routé, le nœud qui en dispose (que ce soit la source ou un nœud intermédaire) ne fait que déterminer le nœud suivant dans le chemin qu'il juge être le plus court jusqu'à la destination.

II.2 Intégration de l'algorithme de sélection des routes

On se propose dès maintenant de décrire en quoi MPOLSR diffère de OLSR et notamment comment il intègre l'algorithme de sélection des routes. Il convient tout d'abord de noter que MPOLSR conserve les trois premières phases de OLSR, à la différence près qu'il affecte un poids à chaque lien. Les poids sont associés par chaque nœud aux liens menant à ses voisins. Ils sont par la suite envoyés au reste du réseau au moyen des messages TC. La phase de découverte des routes est donc presque identique pour les deux protocoles. À l'inverse, la gestion interne de l'information de routage ainsi que le comportement en cas de transfert sont repensés dans MPOLSR, afin de prendre en compte l'algorithme de sélection.
Ainsi, la table de topologie de chaque nœud dispose d'attributs supplémentaires destinés à prendre en compte le poids des liens. Ces modifications n'ont cependant aucun impact sur le parcours des messages TC via les relais. La réception d'un message TC provoque, comme dans OLSR, la mise à jour de la table de topologie où la topologie et le coût des liens sont modifiés en fonction des nouvelles informations reçues. On conserve en faite pour chaque lien deux valeurs distinctes :

- le coût réel, qui est celui reçu dans le TC ;
- le coût virtuel, qui est égal au coût réel (éventuellement) augmenté par de l'utilisation de l'algorithme de sélection de routes.

Il parait a priori possible de considérer un grand nombre de critères pouvant tenir lieu de poids, c'est-à-dire représentatifs de la capacité du lien à assurer le transfert effectif des données qui lui sont dévolues. Le chapitre 4 définit ainsi la fiabilité d'un lien e comme la probabilité sur une période T (correspondant à l'intervalle de temps entre deux mise à jour de l'information) que e transfert correctement toutes les données prévues. Or, comment évaluer cette probabilité ? Si e est le lien joignant V à W, on peut

par exemple imaginer que la norme de la vitesse relative de ces deux noeuds $||\vec{v}_V - \vec{v}_W||$ constitue une information assez représentative du risque de disparition de e. En effet, plus cette quantité est grande plus les noeuds vont rapidement s'éloigner l'un de l'autre. Malheureusement, si du point de vue des simulations une telle quantité est aisément calculable, il n'en est rien pour un réseau réel. Cela supposerait en effet que tout nœud soit muni de dispositifs particuliers permettant de déterminer des caractéristiques physiques du réseau (connaissance de la position, vitesse, voir même présence d'obstacle...). Dans les cas les plus communs, les nœds ne peuvent accéder au mieux qu'à des informations de type énergétiques (niveau de la batterie, puissance des signaux perçus...) ou de transfert (nombre d'échec de transmission de paquets avec un voisin donné, débit déjà transmis avec succés...). L'article [eDAeJBeRM05] fournit un exemple de métrique plus complexe ayant pour but de rendre compte du nombre moyen de retransmission nécessaire pour qu'un paquet franchisse effectivement un lien. À l'inverse, le choix du nombre de saut (qui correspond à un poids unitaire pour les noeuds) est une métrique qui a le mérite de sa simplicité et l'inconvénient de ne pas prendre en compte les variations de qualités entre deux liens existants. On convient cependant de l'utiliser dans un premier temps quitte à adopter par la suite des métriques plus complexes.
Alors que la mise à jour de la table de routage est effectuée en temps réel dans le cas d'OLSR, il semble plutôt coûteux de chercher à reproduire cette caractéristique pour MPOLSR. En effet, cela nécessiterait de lancer l'algorithme de recherche de chemin pour chacun des autres nœuds (alors que dans le cas d'OLSR, une unique utilisation de Dijkstra fournit les plus courts chemins vers tous des autres nœuds). Le comportement de l'algorithme de recherche de routes multiples sera donc réactif : l'information de topologie est bien stockée à sa réception mais elle ne sera exploitée que si des routes sont effectivement requises.
Il est à noter que l'information de coût est possiblement transmise deux fois (une fois dans le voisinage de chacune des extrémités du lien). Si le coût diffère, il convient de définir quelle valeur est retenue. On choisit pour des raisons de simplicité de conserver la dernière reçue.

II.3 Routage par la source

MPOLSR accorde un rôle plus important à la source du transfert que ne le fait OLSR. En effet, afin de garantir que le choix des routes se fasse conjointement, celles-ci sont déterminées par un seul et même nœud : la source. À l'inverse, dans OLSR tout nœud (la source ou un intermédiaire quelconque) ne se préoccupe que du prochain saut du paquet. Ce simple fait implique que le paquet n'a pas à contenir d'information sur le routage autre que sa destination. Puisqu'on souhaite dans MP-OLSR imposer aux paquets de données un parcours pré-déterminé par la source, il faut en conséquence leur ajouter (à la manière de DSR) des en-têtes spécifiques, contenant entre autre le chemin à parcourir.
Une fois le paquet de données envoyé sur le réseau, chaque nœud intermédiaire lit l'en-tête en question et détermine quel est le prochain nœud destiné à recevoir le paquet.

II.4 Rupture de routes

Chaque nœud intermédiaire recevant un paquet en transit peut comparer le chemin prédéfini indiqué dans le paquet lui-même avec sa propre connaissance de la topologie. Ceci lui permet entre autre de répondre à une simple question : le nœud initialement prévu comme le suivant est-il effectivement un voisin d'après la table de topologie ? Si oui le protocole poursuit son fonctionnement normal. Mais dans l'hypothèse négative, que convient-il de faire ? Plusieurs solutions sont envisageables :

– supprimer le paquet de données ;
– envoyer un message à la source signifiant la rupture de la route ;
– utiliser la connaissance locale de la topologie afin de terminer le transfert à la destination.

Ces solutions ne sont pas nécessairement exclusives entre elles. Dans MPOLSR, la dernière solution est préférée. Autrement dit, le nœud découvrant la rupture choisit lui-même de substituer à la partie incorrecte de la route un itinéraire bis créé à l'aide de sa connaissance du réseau. Si les informations disponibles ne permettent pas de trouver un tel itinéraire de substitution, le paquet est alors effectivement supprimé. La justification de ce choix est que :

– la suppression pure et simple d'un paquet peut conduire à des pertes non négligeables étant donné que par principe un réseau ad hoc est instable ;
– si `S` ne possède pas une information à jour concernant la partie du réseau où la rupture de route se produit, lui déléguer à nouveau la responsabilité de déterminer un nouveau chemin risque fort d'aboutir à une route encore une fois invalide ;
– l'envoi d'un message d'erreur contredirait la stratégie de routage recherchée : il s'agit justement de limiter les retours en arrière en s'appuyant sur la redondance d'information.

On notera que si la perte de route est détectée directement par la source, une nouvelle recherche globale de route peut être menée.

II.5 À propos du regroupement des paquets

L'usage de descriptions multiples implique la transformation d'une information originale $\vec{P}$ en une information redondante $\vec{D}$. Cette information $\vec{D}$ est composée de N éléments D_i appelées descriptions. Mais de quoi se compose l'information originale $\vec{P}$? A priori d'un ou plusieurs paquets de données initialement dédiés à être routés. Comment choisir le nombre de paquets composant $\vec{P}$?

A priori ce paramètre peut sembler libre. Pour une redondance fixée, nous nous proposons de montrer que le débit envoyé sur le réseau est indépendant d'un éventuel regroupement de paquets. Soit λ et λ' les débits de l'information avant et après codage MDC. L'information arrivant dans le module de codage se présente sous la forme d'un flux d'éléments qu'on nommera paquets de données $(\cdots, P_r, P_{r+1}, P_{r+2}, \cdots)$. Celle sortant du module en question se présente également sous la forme d'un flux d'éléments qu'on nommera cette fois-ci descriptions $(\cdots, D_s, D_{s+1}, D_{s+2}, \cdots)$. On notera Long la fonction qui associe à un paquet sa taille en nombre d'octets. Si τ_P et τ_D représentent respectivement les intervalles de temps séparant le passage de deux paquets de données d'une part et celui de deux descriptions d'autre part, on

peut affirmer que :

$$\lambda = \langle \text{Long}(P_r)/\tau_P \rangle$$

$$\lambda' = \langle \text{Long}(D_s)/\tau_D \rangle$$

Le codage utilise les paramètres (M, N) où N désigne le nombre de descriptions générées et M le nombre de description qu'il est suffisant de recevoir pour pouvoir opérer une recontruction des données originales. Un nombre N' de paquets de données est regroupé afin de créer un bloc d'informations $\vec{P}$ destinées à être transformées conjointement.

$$\langle \text{Long}(\vec{P}) \rangle = N' \langle \text{Long}(P_r) \rangle$$

$$\langle \text{Long}(\vec{D}) \rangle = N \langle \text{Long}(D_s) \rangle$$

Si le codage est optimal (hypothèse que l'on fera) on a :

$$\text{Long}(\vec{D}) = \frac{N}{M} \text{Long}(\vec{P})$$

Cette égalité traduit une réalité simple : en cas de codage optimal, l'augmentation de taille se fait dans un rapport N/M similaire à celui caractérisant la redondance. Ce que l'on gagne en voulant rendre l'information plus robuste coûte autant en terme de compacité du résultat. Considérons maintenant τ le temps entre deux utilisations successives du module de codage. Etant donné qu'on attend un nombre N' de paquets et que le codage produit N descriptions, on a les égalités :

$$\langle \tau \rangle = N' \langle \tau_P \rangle = N \langle \tau_D \rangle$$

Le lien entre les débits λ et λ' est alors :

$$\begin{aligned}
\lambda' &= \langle \text{Long}(D_s)/\tau_D \rangle \\
&= \langle \text{Long}(D_s) \rangle / \langle \tau_D \rangle \\
&= (\tfrac{1}{N} \langle \text{Long}(\vec{D}) \rangle) / (\tfrac{N'}{N} \langle \tau_P \rangle) \\
&= (\tfrac{1}{M} \langle \text{Long}(\vec{P}) \rangle) / (\tfrac{N'}{N} \langle \tau_P \rangle) \\
&= (\tfrac{N'}{M} \langle \text{Long}(P_r) \rangle) / (\tfrac{N'}{N} \langle \tau_P \rangle) \\
&= \tfrac{N}{M} \lambda
\end{aligned}$$

Le regroupement n'influençant pas le débit, on peut donc a priori le choisir quelconque. En pratique, on doit cependant tenir compte des contraintes suivantes :

– Les descriptions doivent être traitées par les couches inférieures lors de chaque saut d'un nœud à un autre. Les mécanismes liés à cette progression (par exemple, pour les normes 802.11, on peut penser aux mécanismes ayant pour but la prévention des collisions) ont une durée en moyenne constante pour chaque entité circulant sur le réseau. Autrement dit, ces actions sont répétées chaque fois qu'une description passe d'un nœud à un autre, ou passe d'une couche à une autre au sein d'un même nœud. Par conséquent, plus le nombre de descriptions est grand, plus ce genre de mécanisme est couteux, même à débit constant. Ce genre de phénomène est observé sur NS2. Nous avaons en effet obtenu les courbes des figures 5.1 et 5.2 lors de simulations. Ces courbes montrent qu'à débit constant, il

est préférable d'utiliser des descriptions plus volumineuses et en nombre restreint plutôt qu'un grand nombre de descriptions légères. L'impact est alors positif à la fois sur le taux de paquets délivrés et sur le délai moyen.

– Les paquets circulants sur le réseau sont généralement prévus pour avoir une taille n'excédant pas une valeur prédéfinie ; ce que l'on désigne couramment par l'acronyme anglais MTU (Maximum Transmission Unit).
– La gigue augmente généralement avec la taille des paquets.

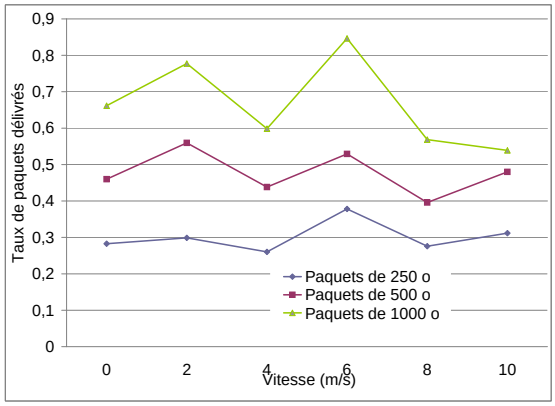


FIG. 5.1 – Taux de paquets délivrés

Deux stratégies apparaissent naturelles en ce sens qu'elles cherchent à conserver des quantités caractéristiques entre flux de paquets de données et flux de descriptions :

– Conserver le nombre d'éléments ($N' = N$). Aucune différence n'est alors visible du point de vue des mécanismes dépendant du nombre de paquets. Cette méthode a néanmoins l'inconvénient d'être susceptible d'augmenter la taille des paquets au delà du MTU.
– Conserver la taille des éléments (la taille moyenne d'une description étant alors égale à la taille moyenne d'un paquet). Pour un système de codage optimal, cela implique que $N' = M$. Les descriptions, prises une à une, sont alors similaires aux paquets de données du point de vue du réseau. Leur nombre est néanmoins plus important.

Si l'on se place dans le cadre systématique, où $\vec{P}$ est également une partie de $\vec{D}$, la stratégie $N' = M$ va de soi : on ajoute aux paquets de données $(N - M)$ paquets de redondance de même taille. Le total constitue un ensemble de N descriptions. Dans un cadre non systématique, aucune des deux stratégies ne semble présenter un avantage évident.

MPOLSR intégre donc un module de regroupement des paquets permettant de grouper les paquets

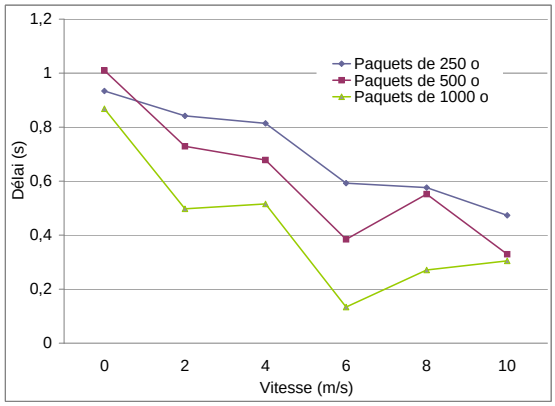


FIG. 5.2 – Délai

ensemble suivant les règles $N' = M$ ou $N' = N$.

II.6 Module de descriptions multiples

Il doit être possible de paramètrer les valeurs M et N. Le nombre de routes sera choisi égal au nombre de descriptions afin que chaque route porte une et une seule description issue d'un même codage ($k = N$). On note que deux listes de stockage supplémentaires sont nécessaires par rapport à OLSR :

- L'une, à l'entrée du module de codage, doit conserver temporairement les paquets de données lorsque la création de descriptions n'est pas encore possible (paquets en nombre inférieur à N') ;
- L'autre, à l'entrée du module de décodage, doit conserver les descriptions ne pouvant pas encore être utilisées pour la reconstruction (en nombre inférieur à M).

L'utilisation de temporisateurs pour chacune des deux listes semble souhaitable. Dans le premier cas, un temporisateur permet de limiter l'attente si les paquets de données suivant tardent à venir, ou s'il n'y en a plus. Le codage est alors effectué même si le nombre N n'est pas atteint. Dans le second cas, un système de temporisateurs assure que les descriptions en attente ne peuvent pas rester éternellement dans la liste. Si par exemple des descriptions manquantes ont été définitivement perdue par le réseau, les descriptions reçues en nombre insuffisant seront supprimées après l'expiration du temporisateur correspondant.

II.7 Le projet ANR SEREADMO

Lancé en novembre 2005, le projet ANR SEREADMO, mené en collaboration avec les entreprises Thales Communications et KEOSYS ainsi qu'avec le département Signal Image Communication du laboratoire XLIM de Poitier, a eu pour but la spécification et la mise en œuvre d'un protocole ad hoc sécurisé. Le

protocole MPOLSR constitue l'apport de l'équipe IVC du laboratoire IRCCyN à ce projet.
L'équipe de Thalès a été chargée de la coordination du projet, avec préparation et pilotage des réunions d'avancement. Elle a également travaillé sur la confidentialité en intégrant des algorithmiques de cryptages. Le laboratoire SIC s'est concentré sur les couches basses de la transmission : prise en compte d'un canal de transmission réaliste dans NS2, (développement de couches physiques réalistes SISO-OFDM et MIMO-OFDM dans NS2). Keosys a travaillé sur la méthodologie associée à la phase d'expérimentation (configurations des tests types permettant l'expérimentation des points clés du protocole) ainsi que sur la mise en œvre du protocole sur un système Linux 2.6.21.5 à partir d'une modification du logiciel Olsrd 0.3. Des tests sur des ordinateurs utilisant la norme 802.11g sont en cours de réalisations par l'équipe IVC-IRCCyN.

II.8 Conclusion

Nous venons donc de décrire un nouveau protocole baptisé MPOLSR. Ce dernier est une variante de OLSR intégrant les caractéristiques suivantes :

- Chaque nœud maintient une vision de la topologie dans une table de topologie au moyen de mécanismes similaires à ceux de OLSR. La recherche de routes n'est en revanche effectuée que lorsque le nœud a besoin de joindre une destination donnée.
- N routes sont recherchées dans la topologie grâce à l'algorithme de recherche de chemins multiples.
- Un nombre N' de paquets de données sont transformées en N descriptions, de sorte que M suffisent à la reconstruction.
- Pour une même transformation, chacune des descriptions créées est assignée à l'une des routes. À cette fin, chaque description comporte dans un en-tête spécifique la liste des nœuds qu'elle est cencée parcourir.
- Lors du parcours de la description dans le réseau chaque nœud intermédiaire lit dans la description elle-même l'adresse du prochain nœud. Si l'information est jugée erronée, le nœud intermédiaire détermine une route correcte. Si une telle route existe, la description est redirigée sur un nouveau trajet, sinon elle est supprimée.

III À propos de la mise en œvre dans NS2

La mise en œvre du protocole MPOLSR a nécessité un certain nombre de choix pour la mise en oeuvre des mécanismes théoriques énoncées ci-dessus afin de limiter le nombre de variantes possibles. Cela concerne :

- les mécanismes de files d'attentes des paquets de données (à la source) et des descriptions (à la destination) ;
- les paramètres liés à la création des routes ;
- les paramètres de codage ;
- l'utilisation des informations de la couche liaison de données ;

– les stratégies de maintenance.

III.1 Files d'attente des paquets et descriptions

Deux structures ont été intégrées pour modéliser l'attente des paquets au niveau de la source et de la destination. Ainsi chaque nœud possède une file d'attente permettant de faire patienter des paquets de données générés par les couches supérieures et en nombre insuffisant pour opérer la transformation en descriptions (c'est à dire en nombre inférieur à N'). Dès que la longueur de la file atteint N', elle est vidée et ses éléments sont convertis en N descriptions.
Par ailleurs, on pourrait imaginer que l'étape de codage, qui consiste en pratique à transformer les paquets en descriptions, soit reproduite telle quelle par le simulateur. Ce dernier opérerait alors des transformations mathématiques sur les paquets de données. Ces opérations sont en faite inutiles. Dans NS, les paquets de donnée ne contienne rien, ils simulent l'existence de données. Par conséquent, il s'avère bien plus pratique, lors de l'étape simulant le codage, de stocker les paquets de données dans une structure fictive partagée par tous les noeuds. Ces paquets sont, du point de vue du réseau simulé, en attente d'être reconstruit. Les descriptions correspondantes comportent alors une information (identifiant ou référence) permettant de faire le lien. La destination a alors pour consigne, lors de l'étape simulant le décodage, de n'extraire ces paquets que lorsqu'un nombre suffisant de descriptions correspondantes est reçu.

III.2 Création des routes

Les fonctions d'incrémentation, qui permettent de pénaliser les liens menant aux routes déjà sélectionnée, sont choisies telles que $f_p(c) = f_e(c) = 2c$. Les poids transmis sont ici unitaires. Autrement dit les coûts des routes sont basés sur le nombre de sauts qui les composent.

III.3 Choix de stratégies de descriptions multiples

En ce qui concerne la couche de codage en descriptions multiples, deux versions ont été mises en place :
– La version MDC non systématique a pour paramètres $N' = N = k = 4$ et $M = 2$. Autrement dit, 4 routes portent chacune une description ; 2 étant nécessaires à la reconstruction. Etant donné le principe du codage non systématique, un délai est imposé à certains paquets puisqu'il faut attendre que $N' = 4$ paquets soient réunis pour que le codage soit applicable.
– La version de répartition ("rand-robin") se contente de disperser les paquets de données sur N routes. On peut considérer qu'il s'agit d'un codage systématique dégénéré avec $N' = N = k = M$ et pour valeur commune 3 ou 4 selon les cas (autrement dit les paquets originaux sont répartis sur les routes sans qu'aucune description de redondance ne soit ajoutée). Les paquets sont successivement distribués sur les routes.

Les tests ne portent que sur des flux UDP. De fait la problématique liée au routage des messages d'acquitement TCP n'est pas considérée.

III.4 Feed-back

Le simulateur NS2 permet par ailleurs de prendre en compte une éventuelle information de retour ("feed-back") de la part de la couche MAC vers la couche de routage. Celle-ci avertit que la couche inférieure (liaison de donnée) n'a pas pu envoyer le message en cours de transmission au nœud initialement dèsigné comme le suivant. Ce mécanisme favorise une mise à jour plus rapide de l'information de voisinage. Alors qu'habituellement ce sont les seuls HELLO qui indiquent à un nœud si oui ou non tel voisin est encore accessible, l'impossibilité de lui transmettre une information peut également être interprétée comme révélatrice du disfonctionnement du lien. L'utilisation ou non de cette information a donc éventuellement un impact sensible sur la qualité du routage. On se propose donc de comparer le comportement des protocoles avec ou sans ce mécanisme.

III.5 Maintenance des routes ("Routes recovery")

Afin de mettre en évidence la validité de notre hypothèse concernant les stratégies de maintenance de routes, deux méthodes sont implémentées. La première est celle consistant, pour un nœud intermédaire, à supprimer tout paquet comportant une route non valide. Dans la seconde, le nœud intermédiaire cherche à remplacer une telle route par un chemin déduit à partir de sa propre connaissance de la topologie. Cela revient à exécuter l'algorithme de Dijkstra une seule fois sur sa table de topologie.

IV Tests NS

Au chapitre 3, nous avons déjà comparé entre eux les protocoles de routage ad hoc les plus courant (OLSR, DSR et AODV) afin d'évaluer dans quels contextes ils fonctionnaient mieux. Nous avons par ailleurs dans ce chapitre, d'une part proposé un nouveau protocole basé sur OLSR permettant de prendre en compte les fonctionnalités théoriques étudiées dans le chapitre 2, d'autres part implémenté ce protocole sur NS2.

Il est à présent nécessaire de vérifier par des simulations si nos attentes théoriques se trouve ou non corroborées. En l'occurrence, on cherche à vérifier si l'ajout d'une statégie multichemins et d'un codage MDC de l'information routée peut améliorer les performances de routage.

Nous effectuons donc des simulations sur MPOLSR et OLSR utilisant les mêmes scénarios dans différents contextes (simulations publiées dans [eECeSHeBP08]). Afin d'évaluer l'apport de l'utilisation de routes multiples d'une part et de l'utilisation de descriptions multiples d'autre part, deux séries de tests sont réalisées. Les résultats sont ensuite analysés en utilisant les métriques définies dans le chapitre 3 : taux de paquets délivrés, délai de transmission et coût du routage. La concentration de l'activité a en revanche été délaissée en faveur d'un critère similaire mais plus courament utilisé, appelé répartition de charge. Nous concluons enfin en récapitulant les différences de comportement observées entre les deux protocoles.

IV.1 Tests concernant les chemins multiples

Le but de cette section est de comparer l'impact du fonctionnement multichemins apporté par MPOLSR par rapport à l'utilisation classique d'une route unique dans OLSR. Par ailleurs, sont également analysés l'intérêt du "feed-back" et celui du "route recovery" de MPOLSR. Le test du "feed-back" consiste à savoir si, en cas de perte de paquets en court de routage, la fonctionnalité optionnelle de OLSR/MPOLSR dans laquelle la couche MAC avertit la couche de routage constitue ou non un avantage significatif. L'analyse de la stratégie de "route recovery" a, quant à elle, pour but d'évaluer s'il est ou non préférable de laisser un nœud intermédiaire choisir une nouvelle route lorsque celle initialement prévue par la source n'est plus correcte.

IV.1.a Spécification des tests

Pour chacun des deux protocoles OLSR et MPOLSR, deux versions sont implémentées et testées ; ce qui aboutit à un total de 4 variantes :

– OLSR dans sa version originale (OLSR) ;
– OLSR avec utilisation de la fonctionnalité "feed-back" (FB-OLSR) ;
– MPOLSR avec "feed-back" et routage par la source seule (SR-MPOLSR) ;
– MPOLSR avec "feed-back" et "route recovery" (RE-MPOLSR).

Le tableau 5.1 fait référence aux différents paramètres utilisés lors des simulations. Le modèle de mobilité ici est le Random Waypoint. Afin d'évaluer l'impact de la mobilité sur les performances de tous les protocoles, plusieurs paramètrages du modèle de mobilité sont utilisées. Si le temps de pause T_{pause} est fixé, la vitesse des nœuds est en revanche tirée aléatoirement dans un intervalle $[V_{min}, V_{max}]$ avec $V_{min} = 0m/s$ et V_{max} prenant différentes valeurs. Les différents choix de V_{max} permettent de générer des scénarios où la mobilité moyenne des nœuds est de plus en plus grande.

On notera que le codage est ici absent : il s'agit simplement de distribuer les paquets sur les 3 routes disponibles, et d'évaluer en quoi l'éclatement du flux peut être un premier avantage.

IV.1.b Résultats

La figure 5.3 montre l'évolution du taux de paquets délivrés à mesure que la vitesse des nœuds augmente. Comme on peut le constater, l'utilisation du "feed-back" améliore le fonctionnement d'OLSR. La version SR-MPOLSR réalise des performances comparables à celle d'OLSR ; autrement dit de 10 à 20 % inférieures à celles de FB-OLSR, alors que ces deux versions bénéficient pourtant des informations de la couche liaison de donnée. RE-MPOLSR se place au dessus de ses concurrents en offrant le meilleur taux de paquets délivrés quelle que soit la vitesse.

Comme on peut également le constater sur la figure 5.4, les courbes du coût de routage ressemblent fortement à celles sur le taux de paquets délivrés, bien qu'inversées. Etant donné que les 4 versions étudiées utilisent le même système proactif de mise à jour de l'information topologique, on peut supposer que la quantité de message de contrôle est grosso modo équivalente dans les 4 cas. Le coût de routage

Paramètres du scénario	
Nombre de nœuds n_{tot}	50
Taille de l'aire de simulation	$1000m \times 1000m$
Durée de simulation	200 s
Nombre de transferts	30 CBR (sur UDP)
Durée des transferts	20 s
Débit de chaque transfert λ	10 paquets de 512 octets par secondes
	$= 5120o/s$
Paramètres de mobilité	
Modèle de mobilité	Random Waypoint
T_{pause}	5 s
V_{min}	0 m/s
V_{max}	0,1,2,3,4,5,6,7,8,9,10,15,20,25 ou 30 m/s
Paramètres physiques	
Protocol MAC	IEEE 802.11
Modèle de reflexion	Two-ray ground
Porté des nœuds r	250 m
Paramètres de codage et de routage	
Codage	aucun
k	3
Fonctions d'incrémentation	$f_p(c) = f_e(c) = 2c$

Tab. 5.1 – Paramètres de test pour la comparaison OLSR/MPOLSR

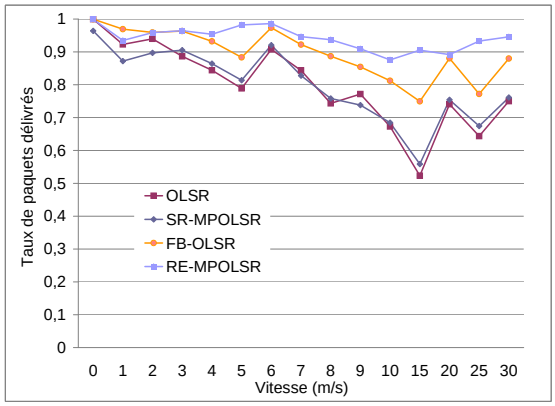


FIG. 5.3 – Taux de paquet délivré

évaluant le nombre de paquets de contrôle utilisés pour chaque paquet de données *effectivement reçu*, il apparait logique de constater une corrélation. Ceci nous amène à penser que l'étude du coût de routage est dans le cas présent peu informative.

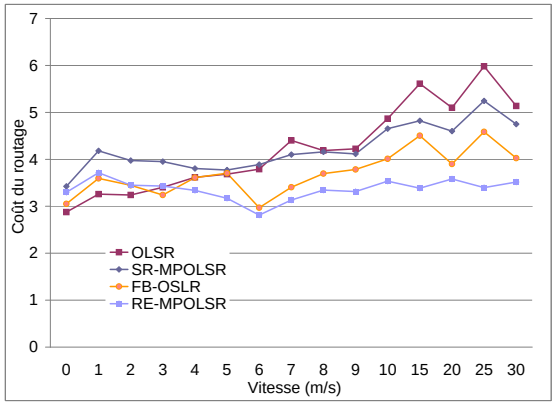


FIG. 5.4 – Coût du routage

Le délai moyen de réception est présenté dans la figure 5.5. La version OLSR reste à la traîne, en particulier lorsque le réseau est fortement mobile. FB-OLSR et SR-MPOLSR sont côte à côte. L'utilisation du

"route recovery" permet à RE-MPOLSR de bénéficier des délais les plus courts en confiant aux nœuds intermédiaires un rôle plus important dans le routage que dans le cas de SR-MPOLSR.

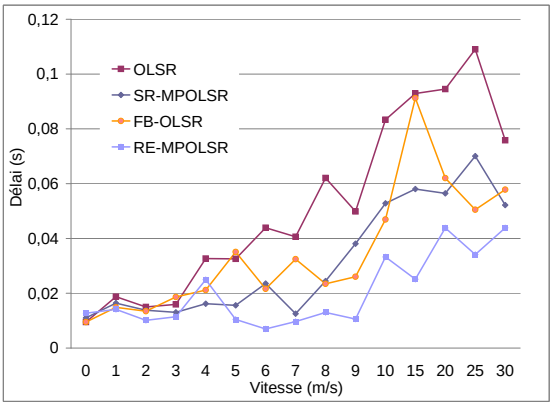


FIG. 5.5 – Délai moyen

La répartition de charge évalue si oui ou non les nœuds participent dans une même proportion à la retransmission des paquets de données. Soit $\Phi_{rep}(V) = |\mathcal{P}^{-V.RTR\rightarrow}|$ le nombre de paquet retransmis par V et soit $\mu_{rep} = \frac{1}{|\mathcal{V}|}\sum_{V\in\mathcal{V}}\Phi_{rep}(V)$ la moyenne de Φ_{rep}. On définit alors la répartition de charge comme :

$$\texttt{RepartitionDeCharge} = \frac{1}{\mu_{rep}}\sqrt{\frac{1}{|\mathcal{V}|}\sum_{V\in\mathcal{V}}|\Phi_{rep}(V) - \mu_{rep}|}$$

La charge est mieux répartie sur le réseau par le protocole MPOLSR que par OLSR (voir figure 5.6). Ce résultat est conforme à nos attentes : l'utilisation de plusieurs routes favorisent une participation plus équilibrée des nœuds. Ceci limite en particulier les situations où un nombre réduit de nœuds devrait supporter la charge générée par un grand nombre de transferts.

IV.1.c Analyse

Il apparaît que le routage par la source pure s'avère une stratégie moins payante en terme de taux de paquets délivrés que des techniques exploitant l'information locale (cas des deux versions d'OLSR et de RE-MPOLSR). Ceci s'explique simplement par le fait que la source dispose nécessairement d'une information moins à jour sur les zones du réseau plus éloignées d'elle. Chaque nœud étant plus à même de connaître son environnement immédiat, l'implication des nœuds intermédiaires dans le processus de routage s'avère payante. Cela ne veut cependant pas dire que le routage par la source est problématique : la dispersion du flux opérée par la source améliore en effet les performances globales des transferts. Cette amélioration est même valable pour le délai de transmission, alors qu'on aurait pu croire que l'utilisation

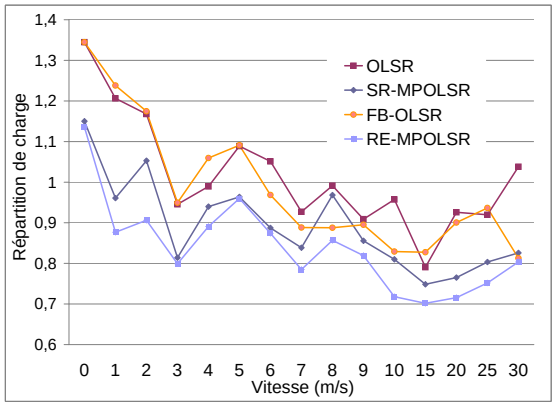


FIG. 5.6 – Concentration de l'activité

de routes a priori plus longues que le plus court chemin aboutirait à une augmentation du temps de trajet. Notre explication de cet état de fait est que le nombre plus faible de collision, dû à un débit moindre sur chaque route, favorise une circulation des paquets plus rapide.

IV.2 Tests sur l'utilisation de descriptions multiples

Dans cette seconde série de tests, nous cherchons à présent à évaluer l'impact sur les performance causé par l'utilisation de descriptions redondantes se substituant aux paquets de données et envoyées sur les routes multiples.

IV.2.a Spécification des tests

On se propose d'introduire dorénavant un codage MDC non systématique (type codage Mojette). Pour évaluer son impact, 3 protocoles sont comparés :

– FB-OLSR ;

– RE-MPOLSR ;

– RE-MPOLSR avec codage MDC (Moj-MPOLSR).

Le tableau 5.2 fait référence aux paramètres spécifiques de RE-MPOLSR et Moj-OLSR. Les autres paramètres de simulation sont identiques aux paramètres présentés dans le tableau 5.1.

IV.2.b Résultats

Nous remarquons que le taux de paquets délivrés est amélioré par l'utilisation de descriptions multiples, en particulier lorsque la vitesse du réseau augmente (voir figure 5.7).

Paramètres de codage et routage de RE-MPOLSR	
Codage	sans
k	4
Fonctions d'incrémentation	$f_p(c) = f_e(c) = 2c$
Paramètres de codage et routage de Moj-MPOLSR	
Codage	Mojette
k	4
M	2
N	4
N'	4
Fonctions d'incrémentation	$f_p(c) = f_e(c) = 2c$

TAB. 5.2 – Paramètres de test pour évaluer l'impact du codage MDC

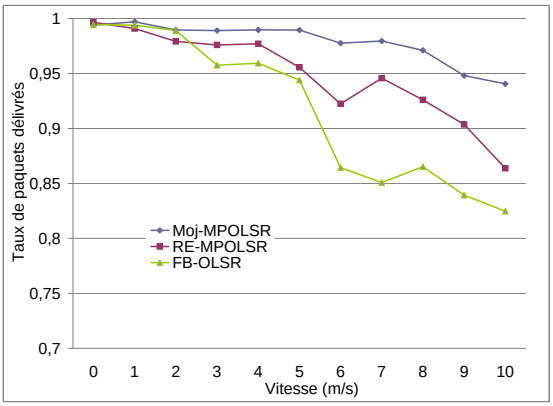


FIG. 5.7 – Taux de paquet délivré

En terme de coût du routage, RE-MPOLSR et Moj-MPOLSR font plus ou moins jeu égal. L'amélioration par rapport à OLSR est cependant peu significative (voir figure 5.8).

IV.2.c Analyse

L'utilisation d'un mécanisme de codage basé sur les techniques de descriptions multiples augmente le taux de réception. Cette augmentation est par ailleurs d'autant plus signicative que les nœuds sont rapides. Comme on pouvait s'y attendre la redondance d'information a un effet positif sur la probabilité de réception de l'information originale. Autrement dit, si la vitesse nuit à la bonne réception de l'information,

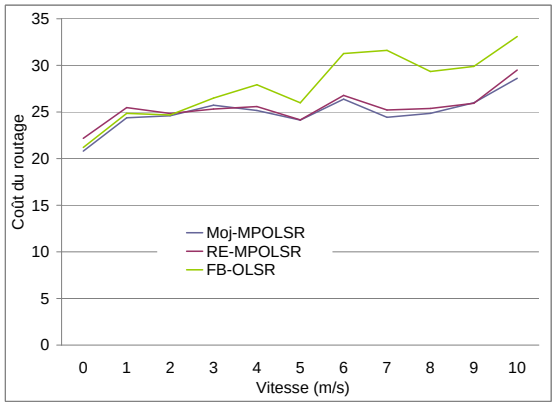


FIG. 5.8 – Coût du routage

son impact négatif est limité en cas de redondance. Pour les taux utilisés, l'augmentation de données à transmettre reste par ailleurs suffisament acceptable pour ne pas provoquer un engorgement du réseau étant donné qu'elle ne provoque pas un effondrement du réserau.

IV.3 Conclusion des tests

Dans les tests effectués avec notre implémentation de NS2, la comparaison du comportement de MPOLSR avec OLSR nous a permis de mettre en avant plusieurs caractéristiques : répartir un flux de données sur différentes routes contribue à diminuer les risques de perte sans impliquer pour autant une augmentation du délais de transmission (en autorisant toutefois les nœuds intermédiaire à jouer un rôle dans le routage en cas de perte de route). En outre, les noeuds participent plus égalitairement au bon fonctionnement du réseau. Le risque d'épuiser énergétiquement certains nœuds et d'en sous-exploiter d'autres est donc moindre. Enfin, l'utilisation de descriptions multiples permet une amélioration supplémentaire du taux de paquets délivrés.

IV.4 Conclusion

Nous avons dans ce chapitre présenté un nouveau protocole de routage réactif essentiellement basé sur le protocoles OLSR. Le choix de OLSR a été motivé par son comportement proactif à état de lien qui permet une intégration pratique des idées proposées dans le chapitre 2. Ce protocole, baptisé MPOLSR intégre ainsi un algorithme de sélection de routes multiples. Celui-ci s'appuie sur la topologie perçue par le nœud source, laquelle est constituée lors de la phase de découverte de la topologie de OLSR. Par ailleurs, est également ajouté un module de codage en descriptions multiples permettant à chaque

source de générer des descriptions redondantes à partir des paquets reçus des couches supérieures. Les descriptions sont ensuite réparties sur les routes, et enfin collectées à la destination par un module de décodage qui détermine quels paquets peuvent être reconstruit.

L'implémentation de ce protocole sur le simulateur NS2 nous a permis dans un second temps de réaliser des tests dont le but principal était d'évaluer les avantages et inconvénients des mécanismes introduits par rapport à un protocole OLSR standard. Ces tests ont confirmé que l'usage de chemins multiples d'une part, de descriptions redondantes d'autre part, peuvent contribuer à améliorer la qualité du transfert, notamment en terme de taux de paquets délivrés et de répartition de la prise en charge de chaque transfert.

Chapitre 6

Le protocole TMR

I Introduction

L'algorithme de calcul de chemins multiples et le schéma général de répartition de l'information sur ces chemins présenté dans le chapitre 4 sont facilement adaptable pour un protocole proactif, comme cela a été montré dans le chapitre 5. On sait cependant que l'approche réactive possède des avantages importants en terme de limitation de charge du réseau. Dans ce chapitre, nous nous consacrons à l'étude d'un protocole de routage baptisé TMR (Topology Multipath Routing), qui se veut le pendant réactif de MPOLSR. Son but est d'associer une procédure réactive de découverte de routes inspirée de DSR et AODV mais utilisant par la suite une sélection de routes et une répartition de l'information similaires à celles utilisées dans MPOLSR. Nous cherchons donc ici à vérifier si l'approche multiroutes et descriptions multiples est adaptable en environnement réactif.
Dans une première partie une présentation générale du protocole TMR est faite. Des tests sous NS2 sont ensuite présentés afin d'évaluer si l'approche multiroutes est pertinente.

II Spécifications de TMR

Comme une grande majorité de protocoles ad hoc réactif, le fonctionnement de TMR contient une phase de recherche de route déclenchée en cas de présence de données à transmettre et basée sur l'utilisation de messages de requêtes/réponses. Cependant, contrairement à DSR et AODV, la route ou les routes ne sont pas définies à partir des chemins parcourus par ces messages. Les réponses transportent en fait des informations diverses sur la topologie du réseau. Ces informations, une fois reçues sont utilisées pour définir des routes en utilisant l'algorithme de recherche de routes du chapitre 4 (d'une façon similaire à MPOLSR). Enfin les routes trouvées sont exploitées et des mécanismes de maintenance participent au renouvellement de l'information de routage.

II.1 Les tables

Afin de réaliser les tâches décrites ci-dessus, chaque nœud `V` gère en interne deux tables. La table d'accès contient les nœuds connus de `V` ainsi que leur distances à `V` et le voisin de `V` à utiliser pour les atteindre. Cette table est principalement utilisée lors de la diffusion des requêtes et le retour des réponses. La table de topologie contient la partie de la topologie connue de `V`. Elle est exploitée en cas de transfert de données, afin de faire circuler les paquets (ou les descriptions générées) en parallèle sur plusieurs routes.

II.1.a La table d'accès

La table d'accès de `V` a pour mission d'enregistrer la distance à laquelle se situent tout autre nœud connu `W`, ainsi que la "direction" pour y faire parvenir une réponse. Autrement dit, le ou les nœuds voisins de `V` jugés les plus à même de router l'information vers `W` (ces nœuds sont alors qualifiés de *relais* vers `W`). On trouve donc dans une entrée `V.aTable[W]` les informations suivantes :

– `V.aTable[W].distance` : distance supposée entre `V` et `W` (si elle vaut 1, `W` est un voisin) ;

– `V.aTable[W].relays` : liste de voisins de `V` à utiliser pour atteindre `W` (les relais) ;

– `V.aTable[W].lastSN` : dernier numéro de séquence de `W` connu.

Deux informations supplémentaires sont ajoutées si et seulement si `W` est un voisin ;

– `V.aTable[W].expiryDate` : date à laquelle l'information sur `W` périme ;

– `V.aTable[W].weight` : poids décrivant la qualité du lien à `W`.

On notera qu'un processus permet de supprimer tout voisin avec lequel `V` n'a pas eu de communication directe avant la date `expiryDate` correspondante. À l'inverse, toute diffusion de message par un nœud voisin `W` provoque la mise à jour de `expiryDate` si cette date existe, ou bien l'ajout d'une nouvelle entrée `V.aTable[W]` dans le cas contraire. De cette façon, la table d'accès dispose d'information de voisinage à jour.

II.1.b La table de topologie

La table de topologie de `V` contient des informations générales sur la topologie connue. Ces informations sont utilisées lors des transferts de données afin de déterminer plusieurs routes sur lesquels les envoyer. Les trajets des requêtes et réponses n'en tiennent pas compte. En revanche, c'est bien grâce aux réponses que les tables de topologie des différents nœuds sont mises à jour.

Une entrée `V.topoTable[W]` contient entre autre la table `V.topoTable[W].neighbourhood`. Cette dernière répertorie le voisinage de `W`. Chaque entrée de `neighbourhood` indique donc un nœud voisin de `W`. Cette entrée est par ailleurs caractérisée par la valeur `V.topoTable[W].neighbourhood[U].weight` correspondant au poids du lien (autrement dit sa qualité) de `W` à `U`. On notera que `V.topoTable` a une structure similaire à un graphe : à chaque nœud connu est associé une liste d'autres nœuds constituant son voisinage. Les liens entre chaque couple de nœuds sont considérés comme symétriques. Par conséquent, l'existence d'un lien entre `X` et `Y` doit se traduire par l'existence :

– d'une entrée `V.topoTable[Y].neighbourhood[X]` (`X` est un voisin de `Y`) ;

– d'une entrée `V.topoTable[X].neighbourhood[Y]` (`Y` est un voisin de `X`) ;

II.1.c Information commune

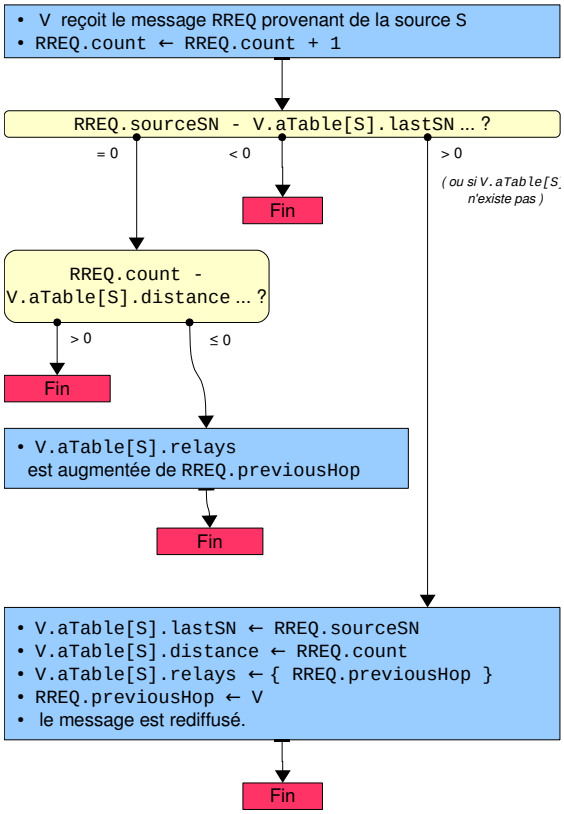
Dans chaque nœud `V`, la table de topologie `V.topoTable` et la table d'accès `V.aTable` contiennent a priori une information commune : le voisinage du nœud courant. Pour éviter d'éventuelles incohérences entre ces deux versions, on convient que la table d'accès est prioritaire sur la table de topologie. Autrement dit, en cas d'utilisation de la table de topologie (lorsque des données doivent être envoyées), le voisinage du nœud courant contenu dans la table d'accès (c'est-à-dire l'ensemble des nœuds `W` tels que `V.aTable[W].distance`= 1) est utilisé pour mettre à jour la table de topologie (et plus spécifiquement la liste `V.topoTable[V].neighbourhood`).

II.2 Requête et réponse

L'objectif des requêtes et des réponses n'est plus comme dans les cas DSR et AODV de laisser les routes se tracer d'elles-même. Il s'agit de permettre à la source de récupérer l'information de topologie constituée des nœuds jugés intéressants pour le transfert entre `S` et `D` : autrement dit, l'ensemble des noeuds suffisament proches du plus court chemin de `S` à `D`.

II.2.a Procédure de requête

Le but des requêtes est double : d'une part, permettre à la source `S` d'atteindre la destination `D` ; d'autre part, permettre aux nœuds intermédiaires de mettre à jour leur table d'accès.
Une requête `RREQ` est générée et diffusée par `S` à tous ses voisins. Elle contient :
– `source` : l'identifiant de `S` ;
– `destination` : l'identifiant de `D` ;
– `sourceSN` : un numéro de séquence donné par `S` ;
– `previousHop` : l'identifiant du nœud précédent, initialisé à `S` ;
– `count` : un compteur de saut, initialisé à 0.
La requête inonde par la suite le réseau. Son traitement par un nœud intermédiaire `V` est décrit par l'algorithme de la figure 6.1 (a). Le passage d'une requête permet à `V` de mettre à jour certaines des informations de sa table `V.aTable`, en l'occurrence : sa distance à `S` et les relais vers `S`. À noter que `V.aTable[S].distance` ne correspond pas systématiquement à la distance au sens du plus court chemin. Elle représente simplement le nombre de sauts effectués depuis `S` par la première version du message `RREQ` atteignant `V`.
Lorsque la procédure de requête prend fin (c'est à dire lorsqu'elle a atteint tous les nœuds), les nœuds atteints connaissent leur distance à `S` et savent comment l'atteindre. Si la destination `D` fait partie de ces nœuds, elle peut envoyer une réponse.



(a) Traitement d'un message `RREQ` par un nœud intermédiaire `V`

(b) Traitement d'un message `RREP` par un nœud intermédiaire `V`

FIG. 6.1 – Algorithmes de traitement des paquets de contrôle

II.2.b Procédure de réponse

L'objectif des réponses est de permettre à la destination `D` de renvoyer un message en retour à `S` tout en collectant l'information de topologie au passage.

Une réponse `RREP` est générée et diffusée par `D` à tous ses voisins. Afin de ne pas superposer l'inondation des réponses avec celle des requêtes, une temporisation retarde la diffusion de la première réponse. Ce n'est que lorsqu'aucune version de la requête n'est plus détectée depuis un certain délai que `D` commence à répondre. Chaque version de la réponse contient :

– `source` : l'identifiant de `S` ;

– `destination` : l'identifiant de `D` ;

– `destinationSN` : un numéro de séquence donné par `D` ;
– `relay` : l'identifiant d'un relai, initialisé avec un élément de `D.aTable[S].relays` ;
– `count` : un compteur de saut, initialisé à 0 ;
– `distSD` : la distance entre `S` et `D`, initialisée grâce aux informations de `D` ;
– `tokens` : un tableau d'information sur les nœuds dont chaque entrée `X` contient la table `tokens[X].neighbourhood` de ses voisins.

Chaque élément de `tokens` est appelé *jeton* (voir figure 6.2). Ainsi, le jeton de `X`, noté `tokens[X]`, est composé d'un tableau tel que l'entrée `tokens[X].neighbourhood[Y]` existe si et seulement si `Y` appartient au voisinage de `X`. Par ailleurs, un attribut `tokens[X].neighbourhood[Y].weight` indique le poids du lien de `X` à `Y`. Le jeton de `X` représente donc son voisinage, et par conséquent, chaque réponse transporte un ou plusieurs voisinages vers la source. Lors de la création de la réponse `RREP` en `D`, elle ne contient alors que le jeton de ce dernier (`tokens[D]`) initialisée grâce à la table d'accès.

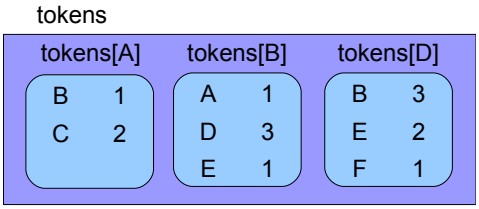


FIG. 6.2 – Tableau des jetons

L'ellipse : zone de dispersion des réponses

Le traitement d'un message `RREP` par un nœud intermédiaire `V` est décrit par l'algorithme de la figure 6.1 (b). Ici la réponse permet avant tout à chaque intermédiaire `V` de définir, de manière symétrique à la requête, sa distance à `D` (enregistrée également dans `V.aTable[D].distance`). Pour un tel nœud, la conjonction de 3 informations sémantiquement liées sont alors accessibles (voir figure 6.3) : sa distance à `S` (`V.aTable[S].distance`)), sa distance à `D` (`RREP.count`), et la distance entre `S` et `D` (`RREP.distSD`). Un noeud recevant la réponse peut alors déterminer s'il se trouve à l'intérieur de l'ellipse de foyer `S` et `D` définie par $\{V \in \mathcal{V} : dist(S,V) + dist(V,D) < \zeta \cdot dist(S,D)\}$ où ζ est une constante supérieure à 1. Cette ellipse est une zone intéressante pour le transfert entre `S` et `D` : elle contient les noeuds qui sont dans la périphérie du plus court chemin de `S` à `D`. Par conséquent, tout nœud recevant une telle réponse et déterminant qu'il est en dehors de l'ellipse la supprime. Il n'est en effet pas tenu de participer au transfert. La réponse ne se propage donc qu'à l'intérieur de l'ellipse.

Traitement de la réponse par un nœud intermédiaire

La réponse, tout comme précédemment la requête, n'est pas redirigée vers un voisin en particulier mais diffusée à l'ensemble des voisins du nœud courant `W`. Elle porte néanmoins le champ `RREP.relay` qui

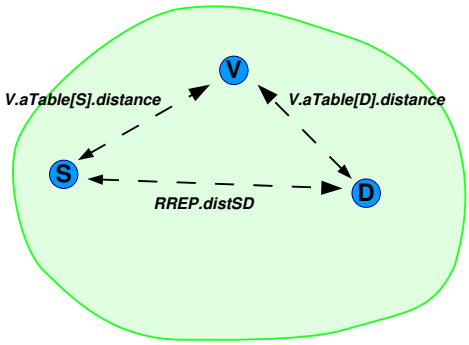


FIG. 6.3 – Ellipse

confère à l'un des voisins de `W` un rôle particulier : celui de relai. Lors de la réception par un nœud `V`, le traitement de `RREP` dépend donc de la nature du nœud `V` (voir figure 6.1 (b)).

- Si le noeud `V` est hors de l'ellipse, le message est ignoré.
- Si le noeud `V` est dans l'ellipse, mais différent de `RREP.relay`, il porte le nom de nœud *témoin*. Il n'a alors pas à tenir compte des jetons reçus `RREP.tokens`. Sa seule préoccupation doit être de déterminer si une version de la réponse venant de `D` a déjà été reçue précédemment (grâce au numéro de séquence). Si oui, le noeud témoin supprime simplement la réponse. Dans le cas contraire, il se doit de participer à la procédure de réponse. Pour celà, une réponse similaire est constituée avec incrémentation du nombre de saut `RREP.count`. Le tableau `RREP.tokens` est initialisé : `RREP.tokens[V].neighbourhood` doit contenir la liste des voisins de `V` récupérée dans `V.aTable`. Le champ `RREP.relay` est défini en sélectionnant au hasard un nœud de `V.aTable[S].relays`. Le prochain relai est donc un noeud du voisinage de `V` situé dans la direction de la source.
- Si le noeud `V` est dans l'ellipse, et d'identifiant égal à `RREP.relay`, il est l'unique nœud *relai* de `W` vers `S`. Son rôle consiste alors à prendre en charge les jetons reçus `RREP.tokens`. Via le numéro de séquence, il peut déterminer si une version de la réponse a déjà été reçue précédemment. Si ce n'est pas le cas, il ajoute sa propre information de voisinage à `RREP.tokens`. Dans tous les cas, l'entrée `V.aTable[D]` est mise à jour avec le nombre de saut `RREP.count` et le nouveau numéro de séquence `RREP.destinationSN`. Le champ `RREP.relay` de la réponse est réactualisé en choisissant un élément quelconque de `V.aTable[S].relays`.

Parcours de jetons

Le cheminement des jetons est détaillé sur la figure 6.4. Lorsque les différentes versions de `RREP` traversent l'ellipse en passant de relais en relais, elles collectent les jetons des nœuds intermédiaires (s'ils s'y trouvent encore) et les stockent dans `RREP.tokens`. N'importe quel nœud de l'ellipse témoin du passage d'une réponse, s'il possède encore son propre jeton, crée sa propre version de la réponse et l'envoie à son tour. On comprend bien que si chaque nœud ne transmettait la réponse qu'à un seul de ses voisins, seule une faible partie de la topologie serait transmise à `S` (correspondant aux voisinages des nœuds sur le plus

court chemin de S à D). À l'inverse si chaque nœud n'établissait pas de différence entre ses voisins (relais et témoins), lesdits voisins récupèreraient des copies des mêmes jetons. Certains jetons seraient donc présents en de multiples exemplaires. Grâce à la différence témoins/relais, le jeton de chaque nœud de l'ellipse est acheminé jusqu'à S en un seul exemplaire.

À noter qu'un même nœud peut recevoir une version de RREP avant même d'avoir renvoyé une précédente version. Autrement dit, deux versions de la réponse peuvent coexister temporairement dans un seul nœud (ces versions contenant toutefois des jetons différents). Dans ce cas, il est possible fusionner de ces versions : il suffit de réunir les jetons qu'elles contiennent. À l'inverse, le cumul des jetons peut aboutir à l'existence de réponses trop volumineuses. Une scission est alors possible.

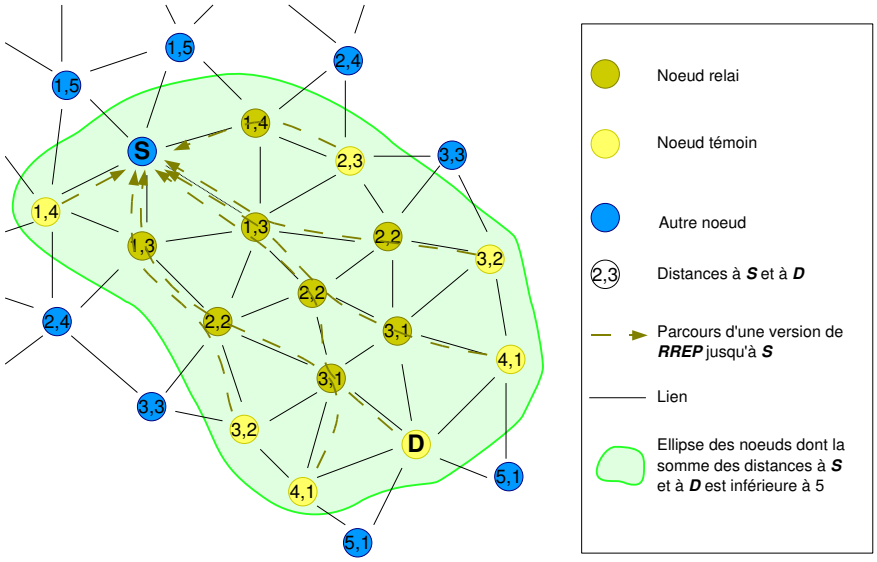


FIG. 6.4 – Processus de collecte des jetons

II.2.c Réactualisation de la topologie

Afin de mettre à jour les routes, tant qu'un flux de données est actif au niveau de la source, celle-ci doit veiller à ce que ses connaissances sur la topologie se renouvelle. À cette fin, une procédure de maintien de route est lancée à intervalle régulier pendant la durée du transfert. Cette dernière est très similaire à la découverte des routes : il s'agit, au moyen de requêtes et réponses, de récolter à nouveau des informations sur les nœuds intéressants. La principale différence tient dans le fait qu'une information de distance entre S et D (distSD) est alors également intégrée aux requêtes. L'inondation des requêtes est alors à son tour restreinte à l'intérieur de l'ellipse. On retrouve donc les deux phases de requête/réponse.

- Une requête innonde le réseau (en se restreignant cette fois à l'ellipse) ; les nœuds atteints mettent à jour leur voisinage, leur distance à S et les relais vers S.
- La réponse correspondante se déploie dans l'ellipse en mettant à jour les distances à D. Les voisinages des nœuds atteints sont collectés sous forme de jetons et acheminés vers S.

II.3 Création de routes

La procédure de création de routes utilisée suit directement toute procédure de recherche de routes par requêtes et réponses. Elle a pour but d'utiliser la topologie rassemblée par `S` pour déterminer un k-uplet de routes jusqu'à `D`.

Dans un premier temps, les réponses sont collectées par la source pour alimenter sa table de topologie `S.topoTable`. Le jeton `RREP.tokens[V]`, répertoriant la liste des voisins de `V` (et le poids des liens), est ainsi utilisé pour mettre à jour `S.topoTable[V].neighbourhood` .

Un temporisateur est enclenché à l'arrivée de la première réponse et remis à jour à chaque version de la réponse reçue. Lorsqu'il expire (aucune réponse reçue dernièrement), le calcul des routes est effectué. Comme dans MPOLSR, cela se traduit par l'application de l'algorithme de recherche de routes multiples présentés dans la première partie. k routes sont donc déterminées dans la topologie connue en utilisant des fonctions de coûts f_p et f_e (voir chapitre 4).

II.4 Répartition de l'information

Pour un ensemble de k routes définies entre `S` et `D`, on rappelle que plusieurs stratégies de répartition de l'information à transmettre sont possibles. Le protocole TMR est prévu pour offrir les 3 possibilités suivantes : le Round Robin (les paquets de données sont dispersées sur les différentes routes disponibles), le codage MDC non systématique (les paquets de données sont regroupés et transformés en descriptions équivalentes, envoyées sur les routes disponibles) et le codage MDC systématique (les paquets de données sont regroupés afin de générer des descriptions de redondance ; ces dernières et les paquets originaux sont envoyées sur les routes disponibles). Comme pour DSR et MPOLSR, la route prévue pour chaque paquet/description est directement intégrée dans celui-ci dans un en-tête prévu à cet effet. Ce dernier comporte également divers paramètres (M, N, un identifiant pour le groupe de paquets originaux). Le routage des paquets est donc a priori le choix de la source seule.

Cas du RoundRobin

Les paquets sont considérés un à un et répartis sur les k routes disponibles par distribution cyclique. Si aucune route n'est perdue, chacune transporte en moyenne autant de paquets qu'une autre.

Cas MDC non systématique

Si les paramètres du codage par descriptions multiples sont N (nombre de descriptions générées) et M (nombre de descriptions suffisantes), les paquets sont regroupés par groupe $\vec{P}$ de $N' = M$ et les N descriptions générées à partir de ces derniers sont réparties sur $k = N$ routes. Comme pour MPOLSR, cette méthode nécessite d'introduire un retard : N' paquets de données doivent être parvenus à la couche routage de la source pour que les descriptions puissent être générées.

Cas MDC systématique

Les paquets sont ici aussi regroupés par groupe de $N' = M$ et $(N - M)$ descriptions de redondance sont générées. Le total des $N = k$ descriptions est réparti sur les routes. Aucun retard n'est nécessaire : les M pseudo-descriptions (autrement dit les paquets de données originaux) peuvent être délivrées en temps réel, une copie devant cependant être conservée en mémoire pour pouvoir créer les descriptions de redondance dès que N' paquets sont reçus des couches supérieures de `S`.

II.5 Utilisation des routes

Comme dans MPOLSR, les en-têtes des paquets sont utilisés par les nœuds intermédiaires pour sélectionner le nœud suivant. Toutefois, les routes définies dans les en-têtes ne sont pas nécessairement respectées à la lettre. Si la route prévue n'est plus valide ou si le nœud courant estime qu'elle peut être racourcie, des actions particulières sont effectuées.

II.5.a Ruptures

Comme dans MPOLSR, lorsqu'un paquet parvient à un nœud intermédiaire, la table de voisinage de ce dernier est inspectée afin de déterminer si le prochain nœud prévu dans la route portée par le paquet est effectivement un voisin. En cas de réponse négative, un calcul de route est effectué par le nœud courant afin de dévier le paquet sur une nouvelle trajectoire vers la destination. Si aucune trajectoire n'est trouvée ou si la trajectoire calculée conduit à l'existence d'une boucle avec le chemin déjà parcouru, le paquet est supprimé. Dans le cas contraire, la route portée par le paquet est mise à jour et ce dernier est renvoyé sur le réseau.
Contrairement à DSR ou AODV, la découverte d'une rupture de route n'implique pas l'envoi d'un message particulier à la source. Celle-ci doit à terme redéfinir les routes utilisées après chaque procédure de réactualisation de la topologie.

II.5.b Court-circuits

Un autre mécanisme limite l'aspect orienté source du protocole : il correspond à une possibilité de court-circuit sur les nœuds intermédiaires. En effet, lorsqu'un nœud intermédiaire `V` doit retransmettre une description, il peut déterminer en lisant l'en-tête s'il existe, dans le reste de la route prévue, un nœud plus intéressant que le nœud suivant. Est ainsi sélectionné le nœud le plus proche de la destination qui soit par ailleurs un voisin de `V` (ce qui peut correspondre ou non au nœud suivant dans le trajet prévu dans l'en-tête). Ce mécanisme permet ainsi d'emprunter des raccourcis méconnus de la source en supprimant des intermédiaires inutiles. On notera bien que le paquet ne change pas de trajectoire. En effet, le mécanisme de court-circuit ne fait que supprimer des intermédiaires initialement prévus mais jugés inutiles ; il ne peut en revanche pas ajouter de nouveaux nœuds à la route.

II.6 Conclusion

TMR est donc un protocole de type réactif utilisant une procédure de type requête / réponse pour permettre à une source de collecter à la demande l'information de topologie pour un transfert vers une destination donnée. Cette information, une fois rassemblée au niveau de la source, permet de reconstituer une vision partielle du réseau et d'en extraire, au moyen de l'algorithme de sélection de routes, un k-uplet de chemins. Les paquets de données peuvent alors être acheminés à destination sur ces routes, soit directement, soit après codage à description multiple. Comme dans MPOLSR, le routage est orienté source mais peut utiliser l'information locale en cas de rupture.

III Tests

Les tests effectués sur TMR ont pour but d'évaluer les différents types de fonctionnement possible du protocole. Le but est de le comparer avec DSR mais également de vérifier si un gain est obtenu avec l'utilisation de routes multiples et de redondance. Dans l'affirmatif, on cherche en particulier à savoir pour quelle nombre N de routes et quel niveau M de redondance les meilleurs résultats sont obtenus.

III.1 Paramètres

Le tableau 6.1 répertorie les paramètres généraux utilisés pour les tests. Pour chaque jeu de paramètres donné, huit scénarios générés aléatoirement ont été simulés. Les résultats sont donc obtenus par moyennage de ces scénarios.

III.2 Résultats

Nous nous sommes concentrés sur trois critères : le taux de paquets délivrés, le délai et le coût du routage.

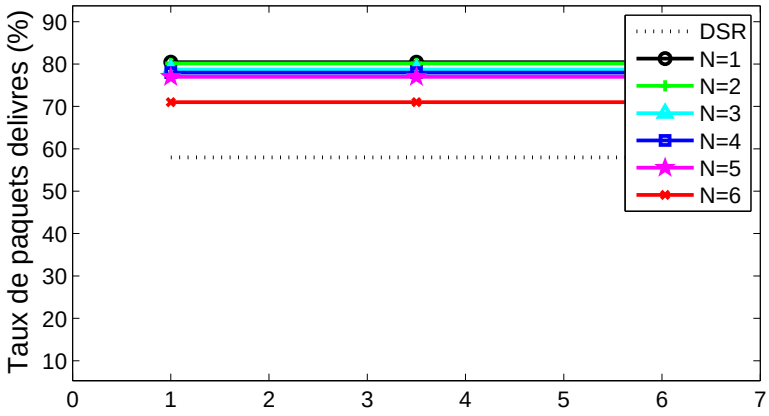
III.2.a Impact de la méthode de codage

Sont ici considérées des simulations de 100 nœuds échangeants 10 paquets par secondes. La portée des nœuds est de 175 m.

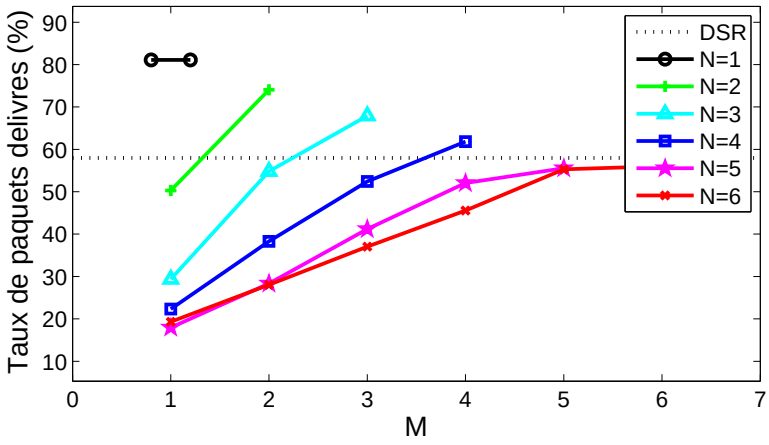
L'analyse de la figure 6.5 (a) montre que la répartition de l'informations sur plusieurs routes n'implique pas d'amélioration du taux de paquets délivrés par rapport au cas $N = 1$. De manière générale, les résultats avec $N > 1$ sont en faite comparables à $N = 1$ sauf si N devient trop grand. Lors de l'utilisation d'un code non systématique (voir la figure 6.5 (b)), le taux de paquets délivrés s'améliore quand M augmente (c'est-à-dire quand la redondance diminue) à constant, mais également lorsque N diminue (quand peu de routes sont utilisées). Le codage non systématique, n'est donc vraisemblablement pas une méthode très efficace. En ce qui concere la version systématique, les résultats sont meilleurs. Les configurations où $N = M$ fournissent ainsi des taux similaires à ceux obtenues avec une seule route (sans toutefois les dépasser). On doit cependant garder à l'esprit qu'un codage systématique où $N = M$ est en faite équivalent au Round Robin. De fait, l'information de 6.5 (a) est contenue dans 6.5 (c). Par

Paramètres du scénario	
Nombre de scénarios	8
Nombre de nœuds n_{tot}	50, 75 or 100
Aire de simulation	$1000m \times 1000m$
Durée	300 s
Nombre de transmissions	30 CBR (sur UDP)
Durée des transmissions	20 s
Taux de paquets par seconde	10, 25 paquets/s
Taille des paquets $size(P)$	512 octets
Paramètres de mobilité	
Modèle de mobilité	Random Waypoint
Pause time	5 s
Vitesse minimale	5 m/s
Vitesse maximale	10 m/s
Paramètres des couches basses	
Protocol MAC	IEEE 802.11
Modèle de réflexion	Two-ray ground
Portée des nœuds	175 or 250 m
Paramètres de TMR et de codage	
N	1,2,3,4,5 ou 6
M	$1 \leq M \leq N$
Fonction d'incrémentation des poids	$f_p(c) = c + 1000$
	$f_e(c) = c + 500$

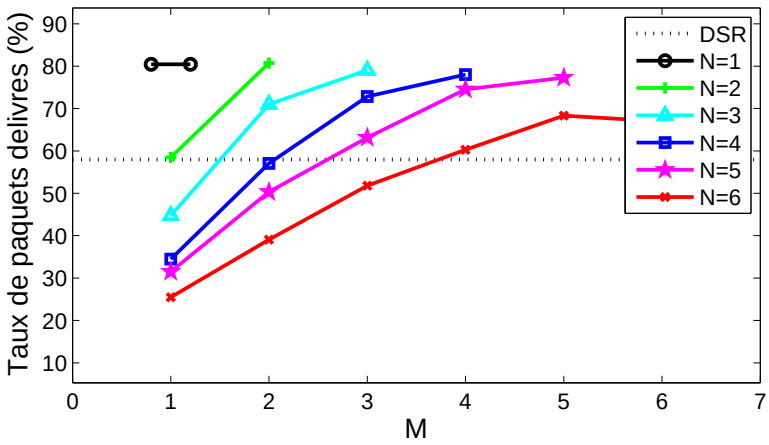
TAB. 6.1 – Paramètres de test pour l'évaluation de TMR



(a) Round robin



(b) Codage non systématique



(c) Codage systématique

FIG. 6.5 – Taux de paquets délivrés, 100 nœuds, 10 paquets/s, portée de 175 m

ailleurs, comme le codage non systématique offre toujours des résultats moindre, il est possible de s'en tenir à l'analyse du codage systématique. On note enfin que, dans ce contexte, TMR obtient de meilleures

performances que DSR pour la plupart des valeurs de (M, N).

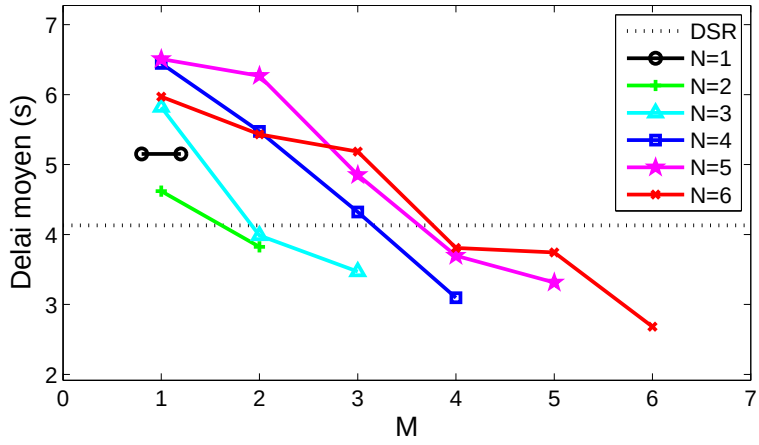


FIG. 6.6 – Délai, 100 nœuds, 10 paquets/s, portée de 175 m, codage systematique

En ce qui concerne le délai (voir figure 6.6), il apparait que, bien que DSR ait un délai plus court que la plupart des cas d'utilisation de TMR, ce dernier s'en sort toutefois mieux que DSR dans la plupart des cas de Round Robin (c'est à dire pour $M = N$). On prendra toutefois soin de noter que le cas $M = N = 6$ n'est pas pertinent puisque le faible délai s'explique en partie par un faible taux de paquets délivrés. Concernant le codage, il ne semble pas apporter ici non plus d'amélioration probante.

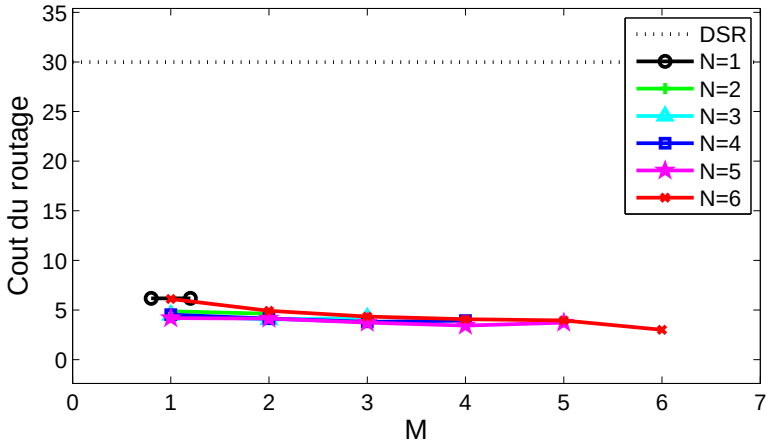


FIG. 6.7 – Coût du routage, 100 nœuds, 10 paquets/s, portée de 175 m, codage systematique

La figure 6.7 montre que TMR est beaucoup moins gourmand en paquet de routage que DSR, et ce quelle que soit la stratégie de codage (ce qui est cohérent avec le fait que l'ellipse a toujours les mêmes dimensions quel que soit le nombre de routes recherchées).

III.2.b Impact du débit

On considère ici des simulations de 100 nœuds exchangeant 10 ou 25 paquets par secondes. La portée commune est de 175 m. En comparant les figures 6.5 (c) et 6.8, nous pouvons observer que le comportement

général de TMR n'est pas vraiment affecté par l'augmentation du débit. Le taux de paquets délivrés est toutefois divisé par deux (mais il est divisé par trois pour DSR).

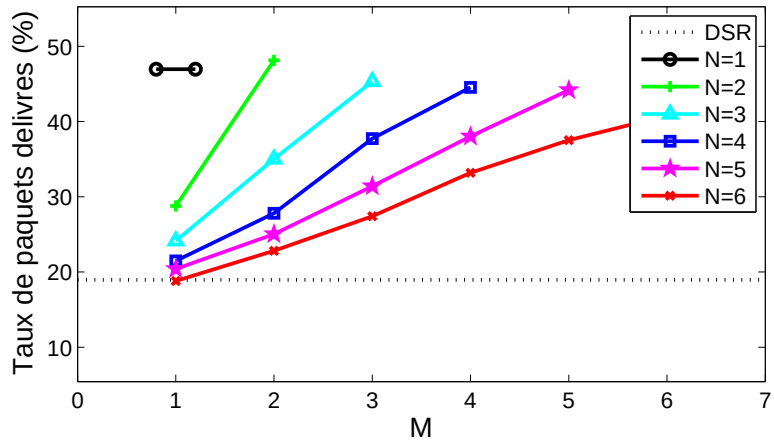


FIG. 6.8 – Taux de paquets délivrés, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique

Par ailleurs, le coût du routage est inchangé pour TMR, alors qu'il augmente pour DSR. (comparer les figures 6.7 et 6.9)

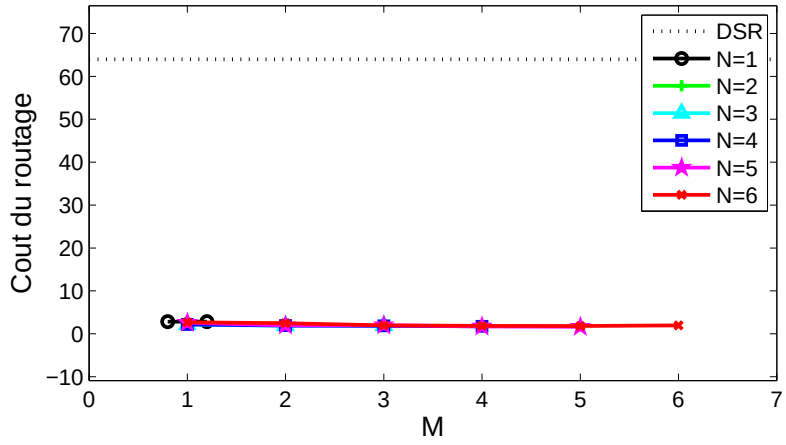


FIG. 6.9 – Coût du routage, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique

Enfin, avec l'augmentation du débit, le transfert devient plus rapide pour TMR multichemins que pour DSR, même pour les cas ou cette amélioration n'est pas due à la baisse du nombre de paquets délivrés (comparer la figure 6.10 à la figure 6.6).

III.2.c Impact de la densité des nœuds

Les figures 6.11 (a), (b) et (c) présentent le taux de paquets délivrés pour une portée de 250 m avec des transferts de paquets par seconde et respectivement 50, 75 et 100 nœuds. À mesure que le réseau gagne en densité, les performances de DSR se dégradent alors que celle de TMR sont peu affectées.

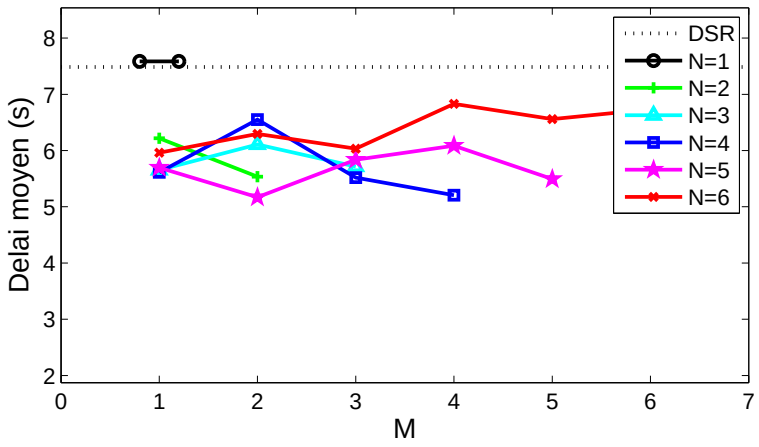
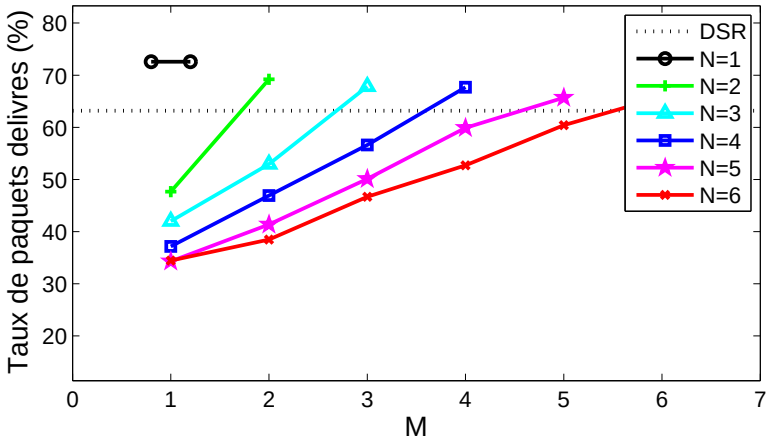


FIG. 6.10 – Délai, 100 nœuds, 25 paquets/s, portée de 175 m, codage systematique

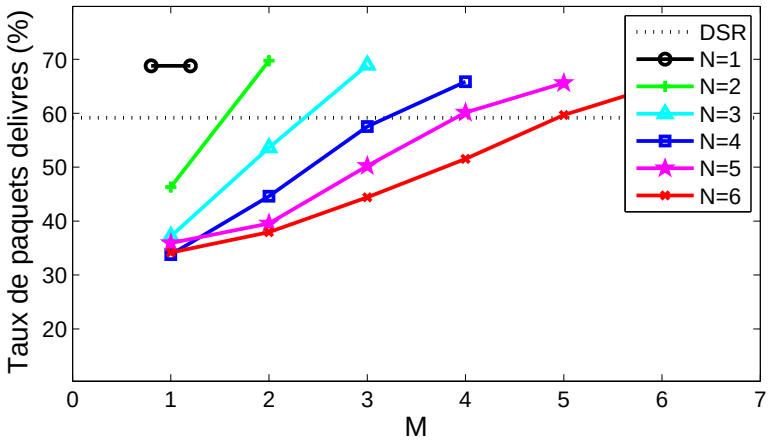
Le délai devient même plus court pour TMR multichemins (par exemple en passant de 4,5 s à 3,7 s pour $N = M = 3$) et augmente pour DSR (d'environ 2,2 s à 4,5 s) de tel sorte que TMR obtient un délai plus faible dans un réseau de 100 nœuds (voir figure 6.12).
Le coût du routage reste plus faible pour TMR (entre 1 et 2) que pour DSR (entre 1,2 et 25). La figure 6.13 montre ce coût pour 75 nœuds.

III.3 Analyse

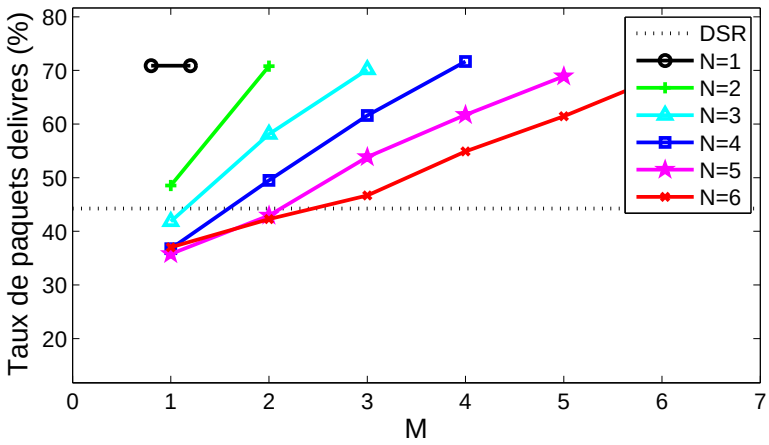
Dans les cas étudiés, l'utilisation de routes multiples n'a pas d'influence positive notable sur le taux de paquets délivrés. À l'inverse, le codage non systématique offre comme attendu des performances moindres par rapport au codage systématique. En revanche, alors qu'on aurait pu s'attendre à une amélioration du taux de paquets délivrés dans certaines configurations basées sur le codage, les résultats sont au mieux équivalents à ceux obtenus avec une route unique. Une explication possible est que l'augmentation nécessaire du volume des données en cas de codage a un impact négatif plus important sur les transferts que la protection apportée par la redondance. Toutefois, TMR dépasse DSR en terme de taux de paquets délivrés lorsqu'utilisé avec les meilleures configurations.
En ce qui concerne le délai, les stratégies multichemins s'avèrent payantes, fournissant même des valeurs plus faible que DSR en cas de portée limitée et de débit important. Enfin, le coût de routage est nettement meilleur pour TMR. Ce fait pourrait être la conséquence de la restriction de l'inondation des requêtes et réponses à l'ellipse, alors que DSR diffuse ses messages de contrôle à l'ensemble du réseau.



(a) 50 nœuds



(b) 75 nœuds



(c) 100 nœuds

FIG. 6.11 – Taux de paquets délivrés, 25 paquets/s, portée de 250 m, codage systematique

IV Conclusion

Ce chapitre a été l'occasion de définir un protocole baptisé TMR dont le but est d'appliquer à un contexte réactif les idées développées dans le chapitre 4 et mises en œuvre une première fois dans un

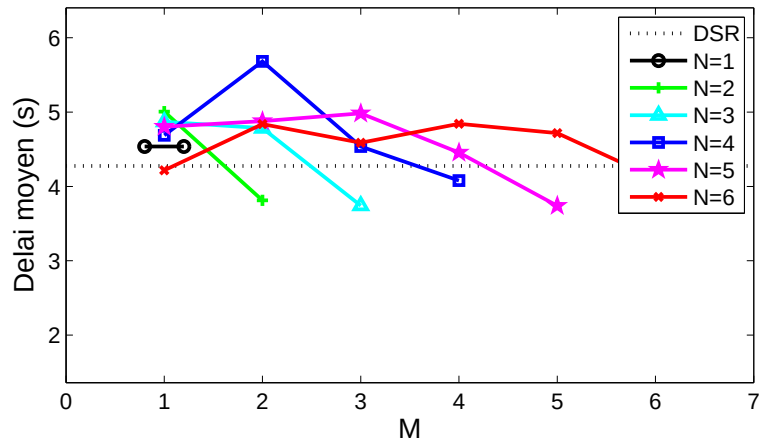


FIG. 6.12 – Délai, 100 nœuds, 25 paquets/s, portée de 250 m, codage systematique

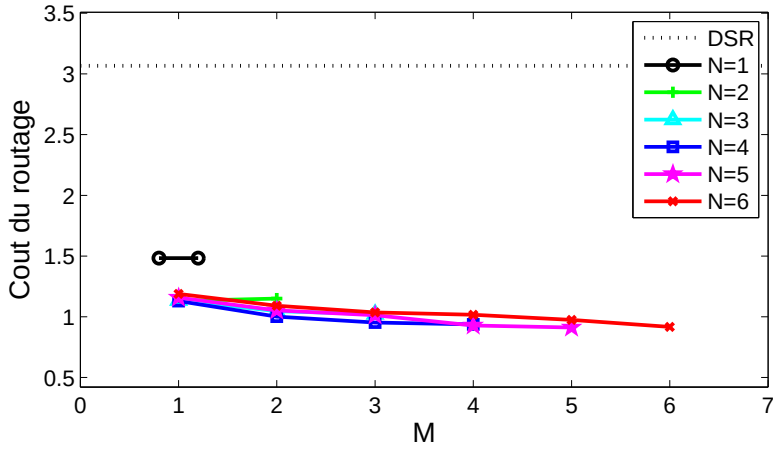


FIG. 6.13 – Coût du routage, 75 nœuds, 25 paquets/s, portée de 250 m, codage systematique

contexte proactif au chapitre 5. TMR est donc un protocole inspiré de DSR, cherchant à collecter de l'information de topologie du réseau par un mécanisme de requête/réponse. L'exploitation des routes est en revanche similaire à celle de MPOLSR : lors d'un transfert, les paquets (éventuellement transformés en descriptions) sont réparties sur les routes et l'utilisation d'un en-tête spécifique permet de garantir leur bon cheminement.

Les tests effectués sur NS2, s'ils ont permis de montrer l'intérêt d'une utilisation simultanées de plusieurs routes (notamment en terme de débit), n'ont en revanche pas mis en évidence d'apport positif dû au codage MDC comme c'était le cas en environnement proactif.

Conclusion

Le routage dans les réseaux ad hoc est une thématique complexe étant donné les propriétés particulières de ce type de réseaux. En effet, l'absence de propriété stable oblige les différents nœuds à échanger des informations sur la topologie plus fréquemment que dans un réseau filaire, et ne peuvent pour autant qu'accorder une confiance limitée dans ces informations. Si des protocoles de routage assez performants ont vu le jour (comme OLSR, AODV, DSR), ils souffrent néanmoins de l'instabilité naturelle de ce type de réseaux.

L'objet de cette thèse a donc été, d'une part, d'utiliser de la redondance fournit par un codage par descriptions multiples appliqué aux paquets destinés à être routés, d'autre part, de disperser les descriptions générées par ce codage sur plusieurs routes joignant un même couple source destination. Cette stratégie avait pour but d'améliorer les performances globales du routage : en introduisant de la redondance sur plusieurs routes, on espérait rendre le transfert moins sensible à la perte d'un certain nombre d'entre elles tout en contrôlant le débit envoyé sur chacune. À cette fin, a été développé un algorithme simple de recherche de routes multiples entre deux nœuds d'un graphe. Son but est de trouver de manière pratique un nombre prédéfini de routes joignant une source à une destination, et réalisant un compromis entre une disjonction souhaitée (mais non impérative comme c'est souvent le cas dans ce genre d'algorithmes) et une longueur raisonnable pour chacune de ces routes. L'étude de la répartition optimale du nombre de descriptions par route ayant révélé un problème complexe, et dont la solution théorique pouvait s'avérer peu intéressante vis-à-vis de la réalité pratique, nous nous sommes finalement restreint à l'usage d'une description par route. Ceci nous a permis d'étudier en détail quelle était la meilleure stratégie à adopter en terme de nombre de route et de redondance globale de l'information, et ce vis-à-vis de la taille du réseau, du débit des flux ou de la stabilité globale des liens du réseaux.

La vérification expérimentale de cette stratégie nous a conduit à proposer un protocole de routage baptisé MPOLSR, adaptation du protocole proactif standard OLSR auquel ont été intégrées nos propositions. Le fonctionnement de ce protocole consiste donc, dans un premier temps, à extraire de la topologie connue de la source du transfert un nombre variable de routes pour chaque destination envisagée. Afin de garantir le respect des différents trajets sélectionnés par la source, ces derniers sont décrits dans un en-tête spécifique ajouté à chaque paquet. Toutefois, si certaines de ces routes deviennent invalides, les nœuds intermédiaires sont autorisés à faire dévier les paquets vers des chemins calculés pour l'occasion. Nous constatons que l'utilisation de routes multiples et la répartition naïve des paquets sur ces routes améliore déjà dans

certaines conditions les performances de routage vis-à-vis d'OLSR. L'ajout d'un codage à description multiple sur les paquets, puis d'une répartition des descriptions ainsi générées sur les routes favorisent à nouveau ces performances en augmentant notamment le taux de paquets reçus.

Le dernier chapitre de cette thèse a eu pour but d'appliquer les mêmes idées que celles précédemment développée, dans un cadre cette fois-ci réactif. L'information de topologie est alors uniquement réunie en cas de transfert de donnée. Il s'agit de TMR, le second protocole proposé dans cette thèse. L'obtention par la source, non plus seulement d'une unique route, mais d'informations plus générales sur la topologie, a cependant nécessité la mise en place de mécanismes différents de ceux de DSR. Ainsi, requêtes et réponses doivent-elles être inondées au sein d'une ellipse entre la source et la destination, afin de collecter l'information topologique correspondante. Le reste du fonctionnement de TMR rejoint essentiellement celui de MPOLSR : sélectionner un certain nombre de routes, créer des descriptions redondantes à partir des paquets de données et disperser ces descriptions sur les routes. Notre analyse s'est ici concentrée sur l'étude de la répartition de la redondance apportée aux données à transférer grâce au codage à descriptions multiples. Le taux de paquets délivrés, le délai et le coût du routage ont été évalués pour différents taux de redondance et deux méthodes de codage : systématique (pour laquelle les données originales sont une partie des descriptions) ou non systématique (pour laquelle les descriptions sont toutes équivalentes). Les résultats ont dans ce cas mis à mal l'intérêt de l'utilisation de codage, notamment dans un cadre non systématique. L'idée d'exploiter de la redondance d'information répartie entre plusieurs routes reste donc une approche pouvant connaître des résultats variables suivant le contexte d'utilisation.

Dans le cadre du projet SEREADMO, le protocole MPOLSR a été mis en œuvre par la société Keosys (voir [eECeSHeBPePL08]). Des tests effectués *in situ* en cours de réalisation doivent permettre de confirmer ou d'infirmer les apports de la stratégie proposée.

D'un point de vue des perspectives, l'algorithme développé pour la sélection de routes multiples pourrait être appliqué à des réseaux filaires ; l'utilisation de plusieurs routes pouvant notamment résoudre les problèmes de congestion. On peut également penser à une adaptation aux réseaux de capteurs qui, étant données leur propriétés particulières, peuvent nécessiter des adaptations spécifiques. Un autre point pouvant faire l'objet de développements futurs est le problème d'optimisation de la répartition des paquets sur les routes. Un algorithme pourrait être élaboré dans le but de fournir une solution approchée susceptible d'améliorer les résultats. À ce propos, il est à noter que cette problématique dépasse le contexte des réseaux ad hoc. Elle peut en effet s'appliquer également à un contexte de stockage distribué d'information redondante.

Enfin, les différences de résultats entre TMR et MPOLSR restent à creuser. Toutefois il convient de noter que l'opposition réactif/proactif a un impact sur la phase de recherche des routes. À l'inverse, la stratégie de répartition de redondance sur plusieurs routes concerne la phase d'utilisation de celles-ci. Dans ce contexte, nous pourrions théoriquement nous attendre à constater une corrélation faible entre ces deux aspects. Il n'est en effet pas aisé d'expliquer en quoi divers méthodes de recherche de routes pourraient avoir des influences différentes sur le comportement des stratégies de répartition de l'information (en

favorisant le routage multichemin dans le cas de MPOLSR, le routage monochemin dans celui de TMR).

Annexe A

Terminologie des réseaux sans fil

Paquet (*packet*) : Volume de données de taille finie échangé dans un réseau.

Nœud (*node*) : Chacun des composants informatiques concernés par la circulation des paquets.

Topologie physique (*topology*) : Répartition physique des nœuds et des liens dans le réseau (non constante pour un réseau mobile).

Lien (*link*) : Capacité à échanger des paquets entre deux nœuds via le média de communication sans transite par un nœud intermédiaire. Par défaut on le considère comme symétrique.

Voisin d'un nœud (*neighbor*) : Chacun des nœuds avec lequel il possède un lien.

Voisin d'ordre 2 d'un nœud (*neighbor*) : Voisin d'un voisin, différent du nœud en question et non directement accessible par celui-ci.

Source : Nœud créateur et émetteur d'un paquet.

Destination ou cible : Nœud auquel est destiné un paquet.

Saut (*hop*) : Franchissement d'un lien.

Route : Chemin utilisé pour le véhicule d' entre une source et une destination et consistant en une suite de sauts.

Routage (*routing*) : Domaine de l'informatique regroupant les méthodes de découverte, d'optimisation et d'entretien de routes sur un réseau donné pour le transport de paquets. Dans la modélisation OSI, le routage constitue la couche 3.

Protocole de routage (*routing protocol*) : Description d'un algorithme particulier utilisé pour le routage.

Arbre source d'un nœud (*source tree*) : Arbre virtuel extrait du graphe du réseau et indiquant pour chaque destination le plus court chemin théorique à utiliser en partance du nœud en question (Fig. A.1).

Prédécesseur (*predecessor*) : Pour un nœud source donné, le prédécesseur d'un nœud `W` est l'avant dernier nœud sur le chemin choisi menant à `W` (en considérant que le dernier est `W` lui-même).

Successeur : Le successeur d'un nœud source `V` donné, dans le chemin à une destination `W` est le voisin de `V` par qui faire transiter les informations destinées à `W`. Dans l'arbre source de `V`, c'est la racine de la branche qui contient `W`.

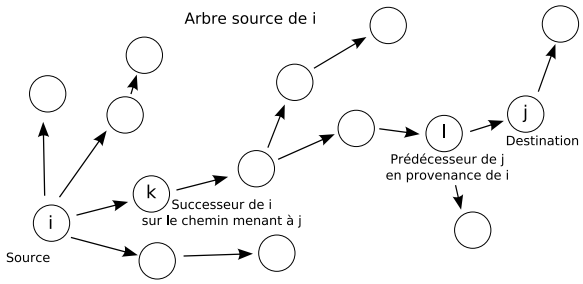


FIG. A.1 – Arbre source de `V`

Annexe B

Format des traces de NS2

Paramètres généraux de l'événement	
	Type (`s` pour envoi, `f` pour retransmission, `r` pour réception, `d` pour perte)
`-t`	Date de l'événement
`-Hd`	Adresse IP du prochain nœud
`-Hs`	Adresse IP du précédent nœud
Paramètres concernant le nœud où a lieu l'événement	
`-Ni`	Adresse IP du nœud courant
`-Nx`	Coordonnée x du nœud courant
`-Ny`	Coordonnée y du nœud courant
`-Nz`	Coordonnée z du nœud courant
`-Ne`	Energie du nœud
`-Nl`	Couche du nœud concernée par l'événement
`-Nw`	En cas de perte, motif
Paramètres concernant la couche MAC	
`-Ma`	Durée du dernier transfert
`-Md`	Adresse MAC destination
`-Ms`	Adresse MAC source
`-Mt`	Type ethernet
Paramètres concernant la couche IP	
`-Is`	Adresse IP source et port correspondant
`-Id`	Adresse IP destination et port correspondant
`-It`	Protocol utilisé au dessus de IP (FTP,DSR,etc)
`-Il`	Taille du paquet

`-If`	Identifiant du flux du paquet
`-Ii`	Identifiant du paquet
`-Iv`	TTL (durée de vie)
Principaux paramètres propres à AODV	
`-P`	Indique `aodv`
`-Ph`	Nombre de saut
`-Pb`	Identifiant de broadcast
`-Pd`	Adresse IP de la destination
`-Pds`	Numéro de séquence de la destination
`-Ps`	Adresse IP de la source
`-Pds`	Numéro de séquence de la source
`-Pl`	Durée de vie
`-Pc`	Opération (requête, réponse, hello, erreur)
Principaux paramètres propres à DSR	
`-P`	Indique `dsr`
`-Ph`	Nombre de saut
`-Pq`	Drapeau de requête
`-Ps`	Numéro de séquence de la requête
`-Pp`	Drapeau de réponse
Principaux paramètres propres à OLSR et MPOLSR	
`-P`	Indique `olsr` ou `mpolsr`
`-Pn`	Nombre de messages
`-Pq`	Numéro de séquence du paquet
`-Pt`	Type du message (HELLO,TC,etc)
`-Po`	Adresse IP de la source du message
`-Ph`	Nombre de saut du message
`-Pms`	Numéro de séquence du message
Principaux paramètres propres à DTDR	
`-P`	Indique `tmr`
`-Pt`	Type du paquet (requête ou réponse)
`-Ph`	Nombre de saut
`-Ps`	Adresse IP de la source de la requête/réponse
`-Pd`	Adresse IP de la destination de la requête/réponse
`-Psn`	Numéro de séquence

`-Pd`	Distance supposée entre source et destination
`-Pnbh`	Information accumulée par la réponse
Principaux paramètres des descriptions utilisées dans DTDR	
`-P`	Indique `description`
`-Pi`	Identifiant de la description
`-Pt`	Type (paquet original ou description de redondance)
`-Pp`	Paramètres M et N du codage
`-Ppath`	Parcours effectué par rapport à celui prévu

Annexe C

Protocoles et modèles de mobilité

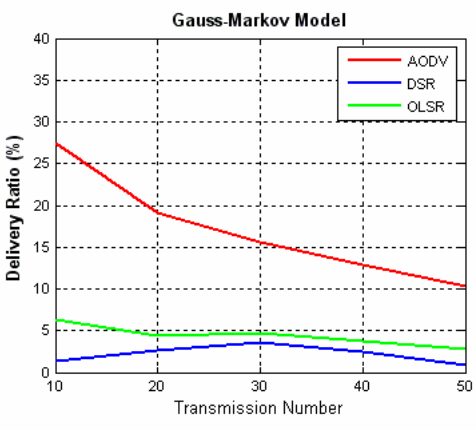
Les différents tests ci-dessous mettent compare les performances des protocoles OLSR, AODV et DSR dans des scénarios utilisant sur les modèles de mobilité suivant : Random Walk, Random Waypoint, Gauss-Markov et Manhattan. Les différents paramètres des tests sont détaillés dans les tableaux C.1 et C.2. Les critères utilisés pour évaluer les résultats des simulations sont : le taux de paquets reçus, le délai, la gigue, le coût du routage et la concentration de l'activité.

Paramètres du scénario	
Nombre de nœuds n_{tot}	200
Taille de l'aire de simulation	$1100m \times 1100m$
Durée de simulation	300 s
Nombre de transferts	10,20,30,40,50,60
Durée des transferts	entre 20 et 50 s,
Débit de chaque transfert λ	16 paquets de $512o$ par secondes $= 8ko/s$ sur CBR
Nombre de scénarios moyennées	3
Paramètres physiques	
Protocol MAC	IEEE 802.11
Modèle de reflexion	Two-ray ground
Portée des nœuds r	100 m

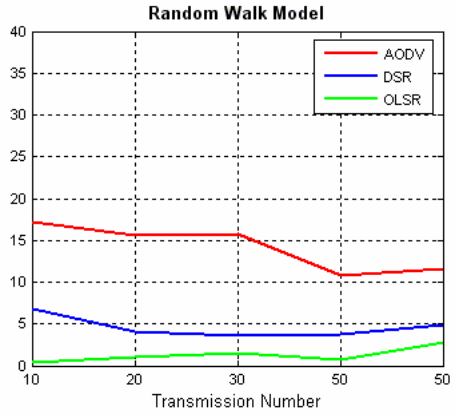
Tab. C.1 – Paramètres généraux

Paramètres du Random Walk Model	
Vitesse minimale V_{min}	2 m/s
Vitesse maximale V_{max}	20 m/s
Pas minimale D_{min}	50 m
Pas maximale D_{max}	100 m
Paramètres du Random Waypoint Model	
Pause time T_{pause}	0 s
Vitesse minimale V_{min}	2 m/s
Vitesse maximale V_{max}	20 m/s
Paramètres du Gauss-Markov Model	
Période T	1 s
Vitesse maximale V_{max}	20 m/s
Écart type de vitesse σ_V	2 m/s
Écart type d'angle σ_θ	0,4 rad
Paramètres du Manhattan Model	
Nombre de blocs sur l'axe Y	10
Nombre de blocs sur l'axe X	10
Probabilité de changement de direction p_θ	0,333
Distance de mise à jour D	50 m
Probabilité de changement de vitesse p_V	0,5
Vitesse moyenne V_{moy}	11 m/s
Écart type de vitesse σ_V	2 m/s
Vitesse minimum V_{min}	2 m/s

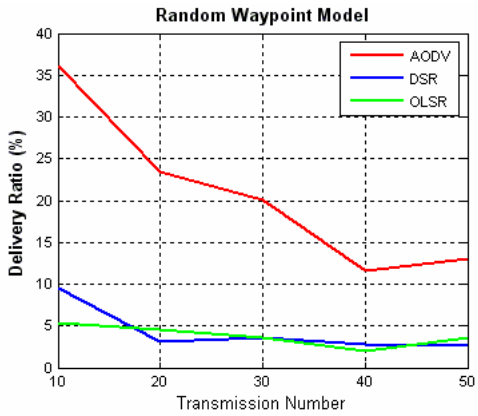
Tab. C.2 – Paramètres des modèles de mobilité



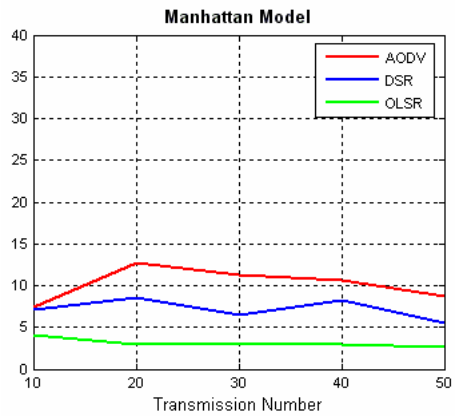
(a) Modéle Gauss Markov



(b) Modéle Random Walk

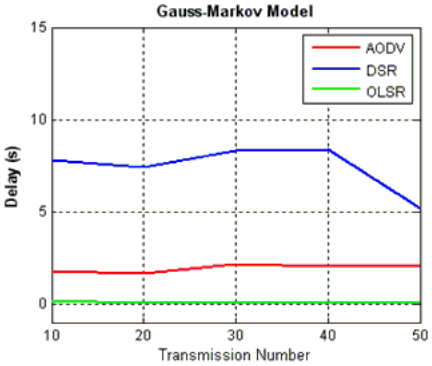


(c) Modéle Random Waypoint

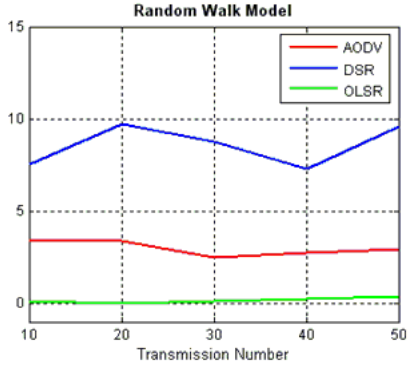


(d) Modéle Manhattan

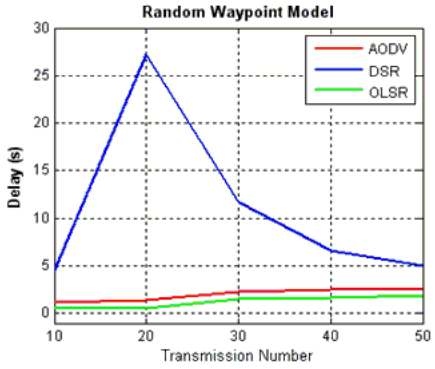
FIG. C.1 – Taux de paquets délivrés en fonction de la charge



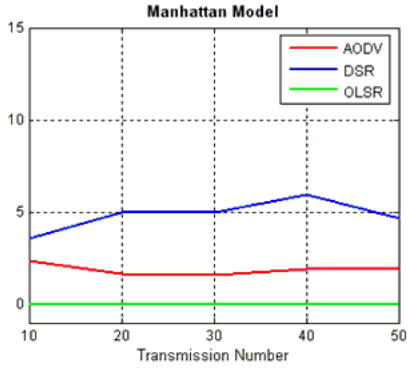
(a) Modéle Gauss Markov



(b) Modéle Random Walk

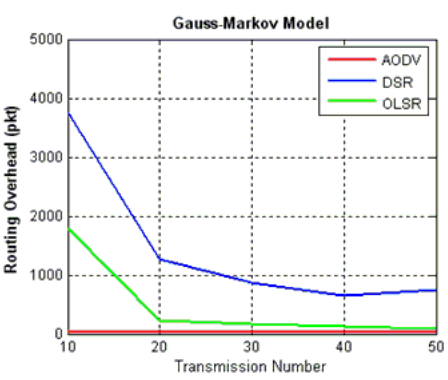


(c) Modéle Random Waypoint

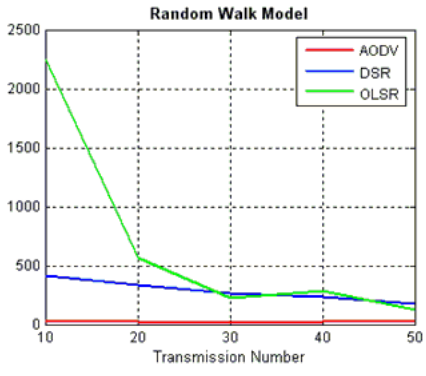


(d) Modéle Manhattan

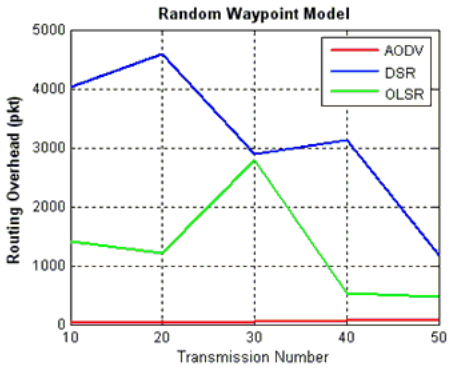
FIG. C.2 – Délai en fonction de la charge



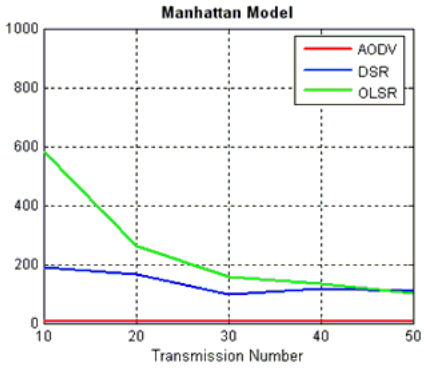
(a) Modéle Gauss Markov



(b) Modéle Random Walk

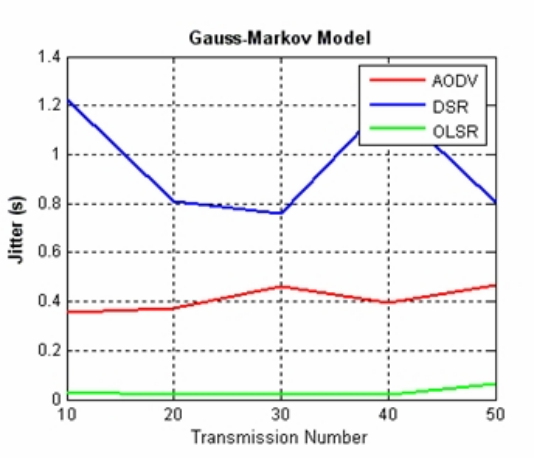


(c) Modéle Random Waypoint

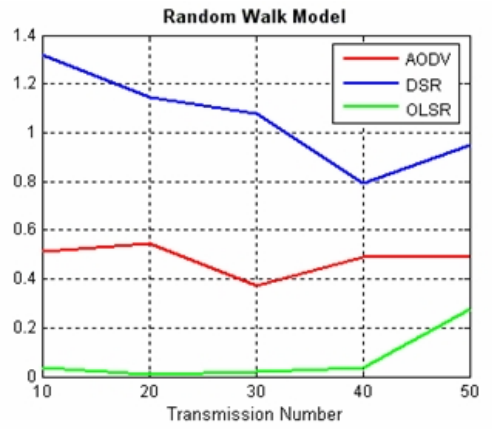


(d) Modéle Manhattan

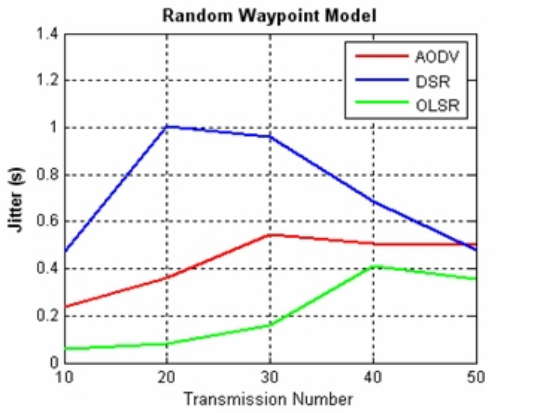
FIG. C.3 – Coût du routage en fonction de la charge



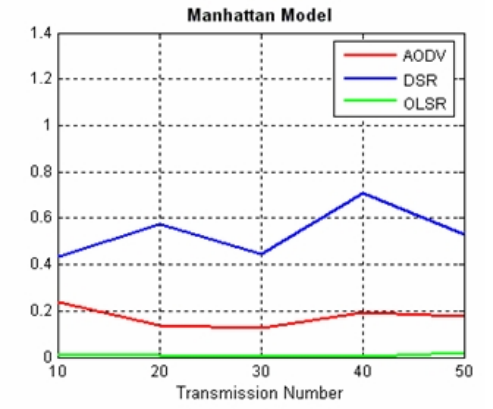
(a) Modéle Gauss Markov



(b) Modéle Random Walk

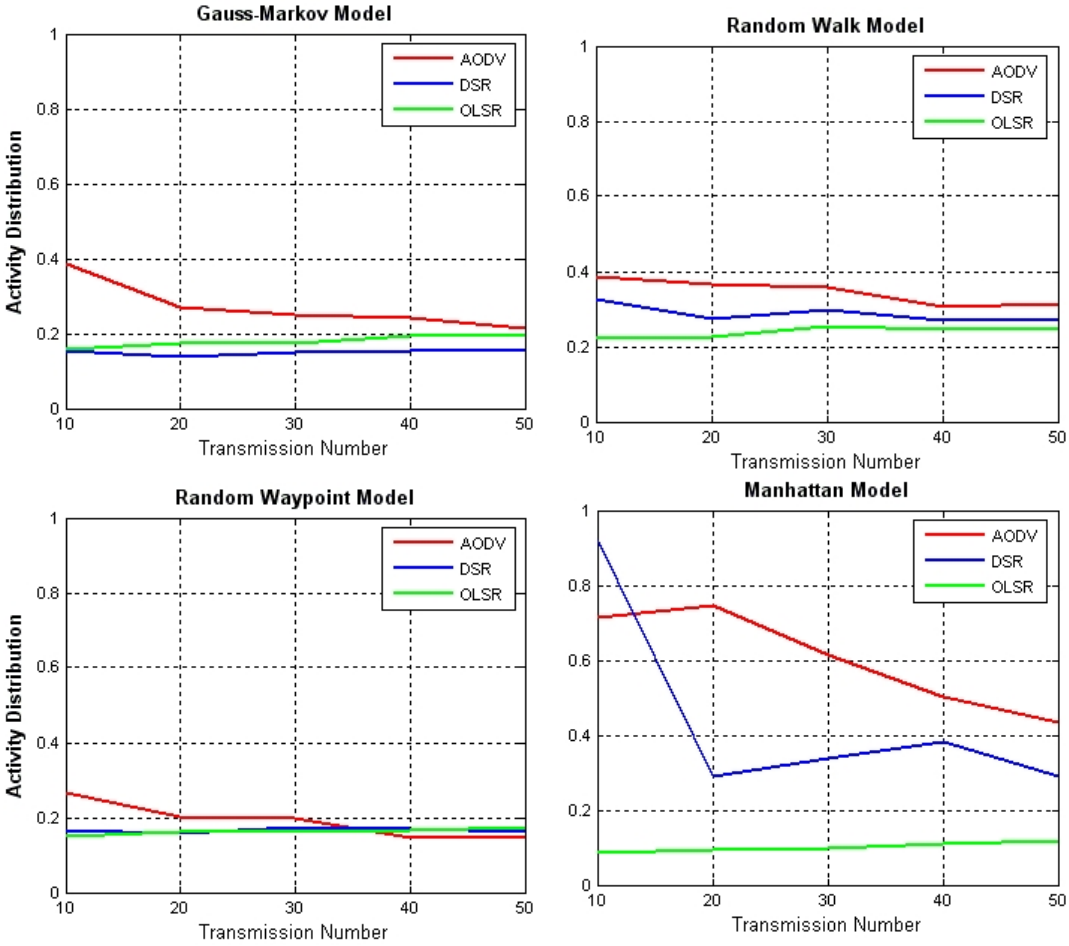


(c) Modéle Random Waypoint



(d) Modéle Manhattan

FIG. C.4 – Gigue en fonction de la charge



(c) Modéle Random Waypoint

(d) Modéle Manhattan

FIG. C.5 – Concentration de l'activité en fonction de la charge

Bibliographie

[Agh04] Khaldoun Al Agha. *Reseaux sans fil et mobiles.* Hermes science, 2004.

[Ahl86] R. Ahlswede. On multiple descriptions and team guessing. *IEEE Transactions on Information Theory*, 32 :543–549, juillet 1986.

[Apo01] J. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Proc. Visual Communication and Image Processing*, pages 392–409, janvier 2001.

[br09] A Collaboration between researchers. The ns manual. http ://www.isi.edu/nsnam/ns/doc/ns_doc.pdf, 2009.

[Chi98] Ching-Chuan Chiang. *Wireless Network Multicasting.* PhD thesis, University of California, 1998.

[eAASeDS01] Vijay Devarapalli et Ali A. Selcuk et Deepinder Sidhu. Mzr : A multicast protocol for mobile ad hoc networks. In *IEEE International Conference on Communications (ICC)*, pages 886 – 891, jun 2001.

[eAH04] F. Bai et A. Helmy. *Wireless Ad Hoc and Sensor Networks.* Kluwer Academic Publishers, 2004.

[eAKePE06] N. Normand et A. Kingston et P. Evenou. A geometry driven reconstruction algorithm for the mojette transform. *Discrete Geometry for Computer Imagery*, 4245 :122–133, octobre 2006.

[eAPeDBJ02] Yih-Chun Hu et Adrian Perrig et David B. Johnson. Ariadne : A secure on-demand routing protocol for ad hoc networks. In *Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002), septembre 2002.

[eBBLeAVePDeMV03] G. Barrenechea et B. Beferull-Lozano et A. Verma et P.L. Dragotti et M. Vetterli. Multiple description source coding and diversity routing : A joint source approach to real-time services over dense networks. In *13th International Packet Video Workshop*, avril 2003.

[eBDeBLeEBR02] K. Sanzgiri et B. Dahill et B. Levine et E. Belding-Royer. A secure routing protocol for ad hoc networks. In *International Conference on Network Protocols (ICNP)*, novembre 2002.

[eBPeNN01] J. P. Guédon et B. Parrein et N. Normand. Internet distributed image information. *Integrated Computer-Aided Engineering*, 8 :205–214, août 2001.

[eBSM04] C. Murthy et B. S. Manoj. *Ad hoc wireless networks.* Bernard M. Goodwin, 2004.

[eCEP99] Elizabeth M. Royer et Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *5th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 207 – 218, aug 1999.

[eDAeJBeRM05] Douglas S. J. De Couto et Daniel Aguayo et John Bicket et Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11 :419 – 434, 2005.

[eDAM96] David B Johnson et David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[eDBJ01] Jorjeta G. Jetcheva et David B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *2nd ACM International Symposium on Mobile and Ad-hoc Networking & Computing (MobiHOC)*, pages 33 – 44, oct 2001.

[eDEeJdJ04] Irene Fernández Díaz et Dick Epema et Jan de Jongh. Multipath routing and multiple description coding in ad-hoc networks : a simulation study. In *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 46 – 51, 2004.

[eDJeAP02] Y. Hu et D. Johnson et A. Perrig. SEAD : Secure efficient distance vector routing for mobile wireless ad hoc networks. In *4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), IEEE*, juin 2002.

[eDMeDJeYCHeJJ98] J. Broch et D.A. Maltz et D.B. Johnson et Y.-C. Hu et J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, octobre 1998.

[eEBReSRD03] C. Perkins et E. Belding-Royer et Samir R. Das. Ad hoc on-demand distance vector (aodv) routing. RFC 3561, Mobile Ad Hoc Networking Working Group, juillet 2003.

[eECeHIeJPG06] S. Hamma et E. Cizeron et H. Issaka et J.-P. Guédon. Performance evaluation of reactive and proactive routing protocol in ieee 802.11 ad hoc network. In *Proceedings of ITCom 06*, octobre 2006.

[eECeSHeBP08] J. Yi et E. Cizeron et S. Hamma et B. Parrein. Simulation and performance analysis of mp-olsr for mobile ad hoc networks. In *Proceedings of the IEEE Wireless Communication and Networking Conference*, pages 2235–2240, mars 2008.

[eECeSHeBPePL08] J. Yi et E. Cizeron et S. Hamma et B. Parrein et P. Lesage. Implementation of multipath and multiple description coding in olsr. In *Proceedings of the 4th OLSR Interop Workshop*, octobre 2008.

[eEMBR04] Ian D. Chakeres et Elizabeth M. Belding-Royer. Aodv routing protocol implementation design. In *24th International Conference on Distributed Computing Systems Workshops*, 2004.

[eEMR02] T.Y. Berger-Wolf et E. M. Reingold. Index assignment for multichannel communication under failure. In *IEEE Transactions on Information Theory*, volume 48, pages 2656–2668, octobre 2002.

[eEMReSRD02] Charles E. Perkins et Elizabeth M. Royer et Samir R. Das. Ip address autoconfiguration for ad hoc networks. DRAFT-ietf-manet-auto-conf-01.txt, 2002.

[eFFeCB06] J. Haerri et F. Filali et C. Bonnet. Performance comparison of aodv and olsr in vanets urban environments under realistic mobility patterns. In *Proceedings of the 5th IFIP Mediterranean Ad-Hoc Networking Workshop*, 2006.

[eFV03] Fabrice Theoleyre et Fabrice Valois. Topologie virtuelle pour réseaux hybrides. Technical Report 5035, INRIA Rhône-Alpes, décembre 2003.

[eGGeSB00] Chai-Keong Toh et Guillermo Guichal et Santithorn Bunchua. On-demand associativity-based multicast routing for ad hoc mobile networks (abam). In *52nd IEEE VTS Vehicular Technology Conference (VTC)*, pages 987 – 993, sep 2000.

[eGR08] H. S. V. Radha Krishna Rao et G. Radhamani. *WiMAX, A wireless tehcnology revolution.* Auerbach Publications, 2008.

[eGSAeXZeATC00] Seoung-Bum Lee et Gahng-Seop Ahn et Xiaowei Zhang et Andrew T. Campbell. INSIGNIA : An IP-based quality of service framework for mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 60(4) :374–406, 2000.

[eHA04] H. Labiod et H. Afifi. *De bluetooth à Wi-Fi.* Hermes Science, 2004.

[eHKWeWLeMG97] Ching-Chuan Chiang et Hsiao-Kuang Wu et Winston Liu et Mario Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In

IEEE Singapore International Conference on Networks, SICON'97, pages 197–211, avril 1997.

[eHL02] Hasnaa Moustafa et Houda Labiod. Srmp : A mesh-based protocol for multicast communication in ad hoc networks. In *2002 International Conference on Third Generation Wireless and Beyond*, pages 43 – 48, may 2002.

[eHWePCeKS02] V. Padmanabhan et H. Wang et P. Chou et K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *NOSSDAV*, mai 2002.

[eHWeVP03] P. Chou et H. Wang et V. Padmanabhan. Layered multiple description coding. In *Packet Video Workshop*, avril 2003.

[eICeVSeBW98] S. Basagni et I. Chlamtac et V.R. Syrotiuk et B.A. Woodward. A distance routing effect algorithm for mobility (dream). In *MOBICOM*, pages 76–84, 1998.

[eITL99] Mario Joa-Ng et I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE journal*, 17, août 1999.

[eJBeJEeMLeMS94] Andres Albanese et Johannes Blomer et Jeff Edmonds et Michael Luby et Madhu Sudan. Priority encoding transmission. In *IEEE Symposium on Foundations of Computer Science*, pages 604–612, 1994.

[eJBeVD02] Tracy Camp et Jeff Boleng et Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2 :483 – 502, 2002.

[eJK01] V.K. Goyal et J. Kovacevic. Generalized multiple description coding with correlating transforms. *IEEE. Trans. Inform. Theory*, 47(6) :2199–2224, juillet 2001.

[eJKeJc02] V. Goyal et J. Kelner et J. cevi. Multiple description vector quantization with a coarse lattice. *IEEE Transactions on Information Theory*, 48(3) :781–788, mars 2002.

[eJLeYCT99] M. Jiang et J. Li et Y. C. Tay. Cluster based routing protocol (cbrp) functional specification. draft-ietf-manet-cbrp.txt, juin 1999.

[eJLeYCTeVPeSC01] M. Jiang et J. Li et Y. C. Tay et V. Park et S. Corson. Temporally-oredered routing algorithm (tora) version 1. draft-ietf-manet-tora-spec-03.txt, juin 2001.

[eJPG97] O. Philippe et J. P. Guédon. Correlation properties of the mojette representation for non-exact image reconstruction. In *Proceedings of Picture Coding Symposium*, volume 1, pages 237–241, septembre 1997.

[eJPG98] N. Normand et J. P. Guédon. La transformée mojette : une représentation redondante pour l'image. *Comptes-Rendus de l'Académie des Sciences*, 326 :123–126, janvier 1998.

[eKReKLeVB01] R. Puri et K. Ramchandran et K. Lee et V. Bharghavan. Forward error correction (fec) codes based multiple description coding for internet video streaming and multicast. *Signal Processing : Image Communication*, 6(8) :745–762, mai 2001.

[eMCS95] A. W. Brander et Mark C. Sinclair. A comparative study of k-shortest path algorithms. In *Proc. 11th UK Performance Engineering Worksh. for Computer and Telecommunications Systems*, September 1995.

[eMG98] Tsu-Wei Chen et Mario Gerla. Global state routing : A new routing scheme for ad-hoc wireless networks. In *IEEE ICC'98*, pages 171–175, juin 1998.

[eMG01] S.-J. Lee et M. Gerla. Split multipath routing with maximally disjoint paths in ad hocnetworks. In *IEEE International Conference on Communications (ICC)*, volume 10, pages 3201–3205, juin 2001.

[eMGeCCC99] Sung-Ju Lee et Mario Gerla et Ching-Chuan Chiang. On-demand multicast routing protocol. In *Wireless Communications and Networking Conference (WCNC)*, pages 1298 – 1302, sep 1999.

[eMGeTWC00] Guangyu Pei et Mario Gerla et Tsu-Wei Chen. Fisheye state routing : A routing scheme for ad hoc wireless networks. In *ICC*, pages 70–74, 2000.

[eMGeXHeCCC99] G. Pei et M. Gerla et X. Hong et C.-C. Chiang. A wireless hierarchical routing protocol with group mobility. In *IEEE WCNC'99*, septembre 1999.

[eMMeSS07] John Hershberger et Matthew Maxel et Subhash Suri. Finding the k shortest simple paths : A new algorithm and its implementation. *ACM Transactions on Algorithms*, 3, novembre 2007.

[eMRPePS02] Zygmunt J. Haas et Marc R. Pearlman et Prince Samar. The zone routing protocol (zrp) for ad hoc networks. http ://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt, juillet 2002.

[eMSeJPG05] N. Normand et M Servières et J. P. Guédon. How to obtain a lattice basis from a discrete projected space. *Discrete Geometry for Computer Imagery*, 3429 :153–160, mars 2005.

[eMZ02] K. Weniger et M. Zitterbart. Ipv6 autoconfiguration in large scale mobile ad-hoc networks. In *European Wireless*, feb 2002.

[eNHV98] Young-Bae Ko et Nitin H. Vaidya. Location-aided routing in mobile ad hoc networks. In *ACM/IEEE Mobicom*, pages 66–75, octobre 1998.

[eNN05] J. P. Guédon et N. Normand. The mojette transform : The first ten years. *Discrete Geometry for Computer Imagery*, 3429 :79–91, avril 2005.

[eNSeAH03] F. Bai et N. Sadagopan et A. Helmy. The important framework for analyzing the impact of mobility on performance of routing for ad hoc networks. *AdHoc Networks Journal*, 1 :383–403, novembre 2003.

[ePB94] Charles E. Perkins et Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM*, pages 234–244, août 1994.

[ePJ03] T. Clausen et P. Jacquet. Optimized link state routing protocol (olsr). RFC 3526, Mobile Ad Hoc Networking Working Group, octobre 2003.

[ePMeTCeALeAQeLV01] P. Jaquet et P Mühlethaler et T. Clausen et A. Laouiti et A. Qayyum et L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the 5th IEEE International Multi Topic Conference*, pages 62–68, 2001.

[Epp99] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28 :652 – 673, 1999.

[eRNeBReHS98] E. Crawley et R. Nair et B. Rajagopalan et H. Sandick. A framework for qos-based routing in the internet. RFC 2386, Mobile Ad Hoc Networking Working Group, août 1998.

[eRP02a] M. Mohsin et R. Prakash. Ip address assignment in a mobile ad hoc network. In *MILCOM*, 2002.

[eRP02b] Sanket Nesargi et Ravi Prakash. MANETconf : Configuration of hosts in a mobile ad hoc network. In *IEEE INFOCOM*, mars 2002.

[eRQ00] Sanjay P. Ahuja et Renato Quintao. Simulation comparison of four wireless ad hoc routing protocols. In *IEEE Vehicular Technology Conference (VTC 2000)*, may 2000.

[eRSeVB99] Prasun Sinha et Raghupathy Sivakumar et Vaduvur Bharghavan. CEDAR : a core-extraction distributed ad hoc routing algorithm. In *INFOCOM (1)*, pages 202–209, 1999.

[eRT99] G. Aggelou et R. Tafazolli. Relative distance micro-discovery ad hoc routing (rdmar) protocol. draft-ietf-manet- rdmar-00.txt, septembre 1999.

[eSC81] N.S. Jayant et S.W. Christensen. Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure. *IEEE Trans. Commun.*, 29 :101–109, feb 1981.

[eSD99] A. Nasipuri et S.R. Das. On-demand multipath routing for mobile ad hoc networks. In *Eight International Conference on Computer Communications and Networks*, pages 64–70, octobre 1999.

[eSH07] E. Cizeron et S. Hamma. A multiple description coding strategy for multi-path in mobile ad hoc networks. In *Proceedings of the Second International Conference on the Latest Advances in Networks - ICLAN*, décembre 2007.

[eSHSeKN02] K. Chen et S. H. Shah et K. Nahrstedt. Cross-layer design for data accessibility in mobile ad hoc networks. *Journal on Wireless Communications*, 21 :49–75, 2002.

[eSRD01] Mahesh K. Marina et Samir R. Das. On-demand multipath distance vector routing in ad hoc networks. In *in Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[eSSH04] Chao Tian et Sheila S. Hemami. Sequential design of multiple description scalar quantizers. In *Data Compression Conference (DCC '04)*, page 32, 2004.

[eSVKeSKT04] Zhenqiang Ye et Srikanth V. Krishnamurthy et Satish K. Tripathi. A routing framework for providing robustness to node failures in mobile ad hoc networks. *Ad Hoc Networks*, 2 :87–107, 2004.

[eTC82] A. El-Gamal et T. Cover. Achievable rates for multiple descriptions. *IEEE Transactions on Information Theory*, 28 :851–857, nov 1982.

[eTG03] Valeri Naoumov et Thomas Gross. Simulation of large ad hoc networks. In *Sixth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2003)*, septembre 2003.

[eTLeNHeBMeMD99] Per Johansson et Tony Larsson et Nicklas Hedman et Bartosz Mielczarek et Mikael Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking table of contents*, pages 195–206. ACM New York, 1999.

[eVSReJD06] Sunil Kumar et Vineet S. Raghavan et Jing Deng. Medium access control protocols for ad hoc wireless networks : a survey. *Ad Hoc Networks*, 4 :326–358, 2006.

[eWDG94] M. H. MacGregor et W. D. Grover. Optimized k-shortest-paths algorithm for facility restoration. *Software - Practice and Experience*, 24(9) :823–834, septembre 1994.

[eWSeALeKC00] H. Xiao et W. Seah et A. Lo et K. Chua. A flexible quality of service model for mobile ad-hoc networks. In *IEEE Vehicular Technology Conference*, pages 445–449, may 2000.

[eXHeLMeGP01] Mario Gerla et Xiaoyan Hong et Li Ma et Guangyu Pei. Landmark routing protocol (lanmar). draft-ietf-manet- lanmar-01.txt, juin 2001.

[eYAeOEeMA05] A.C. Begen et Y. Altunbasak et O. Ergun et M.H. Ammar. Multi-path selection for multiple description video streaming over overlay networks. *Signal Processing : Image Communication*, 20(1) :39–60, janvier 2005.

[eZJH01] A. Tsirigos et Z. J. Haas. Multipath routing in the presence of frequent topological changes. *Communications Magazine, IEEE*, 39(11) :132–138, 2001.

[eZJH02] Panagiotis Papadimitratos et Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, janvier 2002.

[eZJH04] A. Tsirigos et Z. J. Haas. Analysis of multipath routing, part 2 : mitigation of the effects of frequently changing network topologies. In *IEEE Transactions on Wireless Communications*, volume 3, pages 500 – 511, mars 2004.

[eZJHeEGS02] Panagiotis Papadimitratos et Zygmunt J. Haas et Emin Gün Sirer. Path set selection in mobile ad hoc networks. In *MobiHoc '02 : Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 1–11. ACM, 2002.

[eZYeBQeJH04] Samba Sesay et Zongkai Yang et Biao Qi et Jianhua He. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Information Technology Journal*, 3, 2004.

[Gha08] H. Gharavi. Multichannel mobile ad hoc links for multimedia communications. In *Proceedings of the IEEE*, volume 96, pages 77 – 96, janvier 2008.

[Goy00] Vivek K. Goyal. Theoretical foundations of transform coding. *IEEE signal Processing Magazine*, 18(5) :9–21, 2000.

[Goy01] Vivek K. Goyal. Multiple description coding : Compression meets the network. *IEEE Signal Processing Magazine*, 18(5) :74–93, sep 2001.

[Gué09] J. P. Guédon. *The Mojette transform, theory and applications*. ISTE WILEY, janvier 2009.

[Kat78] M.B. Katz. Questions of uniqueness and resolution in reconstruction from projections. *Lecture Notes in Biomath.*, 26, 1978.

[Mer05] Rabah Meraihi. *Gestion de la qualité de service et contrôle de topologie dans les réseaux ad hoc*. PhD thesis, Ecole nationale supérieure des télécommunications, 2005.

[Müh02] Paul Mühlethaler. *802.11 et les reseaux sans fil*. Eyrolles, 2002.

[Nor97] N. Normand. *Représentation d'images et distances discrètes basées sur les éléments structurants à deux pixels.* PhD thesis, IRESTE, université de Nantes, 1997.

[Par01] B. Parrein. *Description multiple de l'Information par Transformation Mojette.* PhD thesis, Ecole Polytechnique de l'Université de Nantes, 2001.

[Phi98] O. Philippé. *Représentation d'images pour le codage conjoint source-canal sur réseaux à qualité de service.* PhD thesis, IRESTE, université de Nantes, 1998.

[Suu74] J.W. Suurballe. Disjoint paths in a network. *Networks*, 4 :125–145, 1974.

[Toh96] Chai-Keong Toh. A novel distributed routing protocol to support ad hoc mobile computing. In *IEEE 15th Annual International Phoenix Conference on Computers and Communications, IEEE IPCCC 1996*, pages 480–486, mars 1996.

[Vai93] V.A. Vaishampayan. Design of multiple description scalar quantizers. *IEEE. Trans. Inform. Theory*, 39 :821–834, may 1993.

[Wan05] C. Wang, Y. et Wu. Low complexity multiple description coding method for networked video. *Signal Processing : Image Communication*, 20(5) :447–457, June 2005.

[Zap05] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector (saodv) routing. draft-guerrero-manet-saodv-04.txt, septembre 2005.

Production scientifique

Performance evaluation of reactive and proactive routing protocol in IEEE 802.11 ad hoc network, S. Hamma et E. Cizeron et H. Issaka et J.-P. Guédon, Proceedings of ITCom 06, Boston, octobre 2006

A Multiple Description Coding strategy for Multi-Path in Mobile Ad hoc Networks , E. Cizeron et S. Hamma, Proceedings of the Second International Conference on the Latest Advances in Networks - ICLAN, Paris, decembre 2007

Simulation and Performance Analysis of MP-OLSR for Mobile Ad hoc Networks, J. Yi et E. Cizeron et S. Hamma et B. Parrein, Proceedings of the IEEE WCNC, Las Vegas, mars 2008

Implementation of Multipath and Multiple Description Coding in OLSR, J. Yi et E. Cizeron et S. Hamma et B. Parrein et P. Lesage, Proceedings of the 4th OLSR Interop Workshop, Ottawa, octobre 2008

Résumé

Les réseaux ad hoc sont un type particulier de réseaux sans fil privés de toute infrastructure fixe. Cette particularité rend le routage très problématique en cas de grande instabilité des éléments qui les composent (nœuds et liens). Le but de cette thèse est d'évaluer l'impact d'une stratégie non conventionnelle consistant à, d'une part, utiliser plusieurs routes en parallèle, d'autre part, introduire de la redondance entre les données réparties sur ces routes grâce à des méthodes de codage à description multiple. Ces méthodes permettent de transformer l'information à transmettre en un nombre défini d'éléments appelés descriptions, et tels que la perte d'un certain nombre d'entre eux n'empêche pas la reconstruction de l'information initiale. L'objectif d'une telle stratégie est de rendre chaque route moins critique, tout en veillant à modérer la redondance globale introduite. Dans ce contexte, un algorithme de sélection de routes est proposé, et différentes stratégies de répartition de descriptions sur ces routes sont étudiées. Afin d'évaluer cette approche, nous avons mis en $\frac{1}{2}$uvre de deux protocoles inspirés de mécanismes standards et incluant les idées précédemment mentionnées. Le premier, MPOLSR, est proactif. Dans ce cas, il suffit d'extraire un ensemble de routes intéressantes de l'information topologique rassemblée. Le second, TMR, est réactif. La récupération d'informations topologiques suffisamment variées nécessite alors des mécanismes différents de ceux utilisés dans les protocoles réactifs standards. Des simulations sont réalisées à l'aide de NS2 pour comparer les performances de ces propositions.

Abstract

Ad hoc networks are a special kind of wireless network with no fixed infrastructure. Because of this feature, routing is a difficult issue in case of unstability of links and nodes. The purpose of this PhD thesis is to evaluate the impact of unconventional strategy wich consists in, on one hand using several routes simultaneously, on the other hand introducing redundancy in the data spread over those routes thanks to coding methods called Multiple Description. The information to transmit is transformed in a given number of elements called descriptions. The loss of a certain number of description does not prevent from reconstructing the original data. The objective of such a strategy is thus to make every route less critical without increasing too much the amount of data transmitted in the network. In this context, an algorithm dedicated to selecting routes is proposed and different strategies for distributing descriptions on these routes are studied. In order to evaluate this approach, we have implemented two protocols, inspired from the standard routing mechanisms and including the ideas mentioned above. The first, MPOLSR is proactive. In this cas, we only need to extract routes from the information naturally gathered in nodes. The second, TMR, is reactive. Then, obtaining topological information requires to use unusual methods compared to classical reactive protocols. Simulations are realised thanks to the software NS2 in order to compare the performance of our propositions.

Mots-clés

Routage, Réseaux Ad Hoc, Multiroutes, Codage à Description Multiple, Simulations