

UNIVERSITÉ DE MONTRÉAL

ROUTAGE ET SURVIVABILITÉ
EN CONCEPTION TOPOLOGIQUE
DE RÉSEAUX À HAUT DÉBIT

RONALD BEAUBRUN

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)

NOVEMBRE 1996



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-26451-3

Canada

UNIVERSITÉ DE MONTRÉAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ROUTAGE ET SURVIVABILITÉ EN CONCEPTION
TOPOLOGIQUE DE RÉSEAUX À HAUT DÉBIT

présenté par: BEAUBRUN Ronald

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. SMITH Benjamin, Ph.D., président

Mme ROSENBERG Catherine, D. Sc., membre et directrice de recherche

M. PIERRE Samuel, Ph.D., membre et codirecteur de recherche

M. DZIONG Zbigniew, Ph.D., membre

À mon défunt père Marcel...

REMERCIEMENTS

Je tiens à remercier tous ceux qui m'ont tant moralement que financièrement supporté dans la réalisation de ce mémoire. Plus particulièrement, j'exprime ma gratitude à ma directrice de recherche, la professeure Catherine Rosenberg du département de Génie électrique et de Génie informatique, ainsi qu'à mon codirecteur, le professeur Samuel Pierre, chercheur au Centre de recherche en informatique cognitive et environnements de formation (LICEF) de la Télé-université. Ce dernier s'est particulièrement dévoué pour me procurer une aide financière adéquate et un encadrement de qualité sans lequel ce mémoire n'aurait pas vu le jour, ce qui fut pour moi une grande source de motivation. Les fructueuses discussions que j'ai eues régulièrement avec lui en rapport avec le sujet m'ont beaucoup éclairé.

Mes remerciements s'adressent également au Centre de recherche LICEF qui a su mettre à ma disposition les ressources matérielles nécessaires à l'accomplissement de mes travaux de recherche et aux membres de notre équipe de travail, dont mes collègues Sabine Kébreau et Hugues Philippeaux. Je m'en voudrais de ne pas remercier Madame Daphné Bélizaire pour ses précieux conseils, ses encouragements et sa présence morale qui s'est surtout fait sentir pendant les durs moments du périple. Enfin, je remercie tous ceux de ma famille qui n'ont jamais cessé de m'encourager et de me supporter tant financièrement que moralement, me permettant ainsi d'atteindre ce but ultime.

RÉSUMÉ

La conception topologique des réseaux distribués constitue un problème fort complexe dont la solution dépend largement de l'algorithme de routage considéré. Ce dernier doit être particulièrement robuste pour à la fois éviter la congestion du réseau et minimiser le délai moyen de chaque message, particulièrement dans le cas de réseaux à haut débit lorsque l'une quelconque des liaisons peut aléatoirement tomber en panne. Ce mémoire porte sur le routage et sur la survivabilité en conception de réseaux à haut débit.

Tout d'abord, nous avons traité le problème général tel qu'il est souvent formulé dans la littérature et présenté une nouvelle stratégie de routage qui, tout en étant simple à implémenter et rapide d'exécution, tient compte du niveau de congestion de chaque liaison pour opérer de façon efficace et produire une solution proche de l'optimum. Une telle stratégie est axée sur certains attributs de liaisons et repose fondamentalement sur deux principes, qui consistent à imposer un seuil au délai de chaque liaison et à acheminer les paquets selon les chemins les plus rapides. Elle comprend essentiellement trois étapes: la première détermine les attributs initiaux des liaisons en acheminant seulement les paquets des paires de noeuds adjacents, la seconde achemine selon les chemins les plus rapides les paquets des autres commodités et s'occupe de mettre à jour les attributs des liaisons impliquées dans les chemins correspondants, alors que la dernière étape utilise les données de l'étape précédente pour déterminer le délai moyen des paquets. L'ensemble de la stratégie est du même ordre de complexité que les

algorithmes de routage existants. Les résultats obtenus d'une telle stratégie sont en général meilleurs que ceux des autres heuristiques et proches de ceux des algorithmes de routage optimal, ce qui nous a convaincus de l'appliquer aux réseaux à haut débit.

Par ailleurs, pour être plus réaliste, nous avons considéré la défaillance possible d'une liaison pendant le routage, défaillance qui nécessite un reroutage de tous les paquets devant transiter par les chemins utilisant cette liaison. Il convient alors de déterminer les chemins de secours appropriés pour absorber de tels paquets sans pour autant congestionner les liaisons qui en font partie, et de façon à maintenir une bonne qualité de service. Tout ceci nous a amené à intégrer dans notre stratégie de routage une nouvelle méthode de survivabilité basée sur le concept de chemins disjoints de liaisons: cette méthode garantit la disponibilité d'au moins un chemin après la défaillance d'une liaison. Elle intègre, entre autres, une stratégie d'affectation de capacités qui s'assure que, par suite d'une panne, les liaisons non défaillantes soient en mesure d'absorber tout le trafic du réseau. En outre, elle permet de déterminer la variation de flot causée par la panne en question sur les liaisons non défaillantes et d'étudier ainsi le comportement de chacune de ces liaisons après la défaillance. Son application sur les réseaux GTE et USA donne lieu à des résultats comparables à ceux obtenus par d'autres méthodes connues de la littérature du domaine, ce qui s'avère satisfaisant.

Ainsi, nous avons apporté une double contribution par ce mémoire. D'abord, nous nous sommes inspirés de certains avantages des diverses classes de routage pour en concevoir une nouvelle stratégie qui implémente dynamiquement une méthode statique et qui conduit à des résultats proches de l'optimum, en évitant le plus possible les chemins

congestionnés. Ensuite, nous avons résolu une nouvelle version du problème de routage en tenant compte de la possibilité de défaillance des liaisons pendant l'acheminement des paquets. Dans le même ordre d'idée, nous avons proposé une nouvelle méthode de survivabilité intégrée à notre stratégie de routage. Les résultats obtenus ont confirmé l'efficacité d'une telle méthode.

ABSTRACT

Topological network design is a very complex problem. Its efficiency largely depends on the routing strategy used in operating the network. The objective of such a strategy is to minimize the average packet delay, subject to multicommodity constraints. Indeed, routing procedures play an important role in the design of distributed networks. This thesis deals with routing through such networks by considering first the global problem, next by taking account the impact of high speed to the routing, and then by integrating survivability to the routing strategy.

This thesis contains several important aspects. First, it presents a new routing strategy based on certain links attributes and path delay. Numerical results show that this procedure is computationally efficient for minimizing the packet average delay, which motivates us to apply it on high speed computer networks.

Moreover, most formal routing algorithms implicitly assume a perfect network whose components are always functional, and only determine the flow assignment for no failure. In reality, both links and nodes, although generally highly reliable, can occasionally fail. Such failures can significantly damage the quality of service offered by the network. This leads us to integrate in our routing strategy a new survivability method that explicitly takes into account the effect of link failure, which guarantees that at least one route is available between each node pair after any link failure. Results show that networks using this method are always survivable and generally meet performance requirements.

TABLE DES MATIÈRES

	Page
DÉDICACE.....	iv
REMERCIEMENTS.....	v
RÉSUMÉ.....	vi
ABSTRACT.....	ix
TABLE DES MATIÈRES.....	x
LISTE DES TABLEAUX.....	xiv
LISTE DES FIGURES.....	xvii
LISTE DES ANNEXES.....	xix
 CHAPITRE I: INTRODUCTION.....	 1
1.1 Définitions et concepts de base.....	2
1.2 Éléments de problématique.....	6
1.3 Objectifs de recherche.....	8
1.4 Résultats escomptés.....	10
1.5 Plan du mémoire.....	10
 CHAPITRE II: CONCEPTION TOPOLOGIQUE	
DES RÉSEAUX DISTRIBUÉS.....	13
2.1 Paramètres fondamentaux.....	13
2.1.1 Attributs de liaisons.....	14
2.1.2 Exigences de trafic.....	14
2.1.3 Fiabilité.....	16

2.1.4	Stratégies de routage.....	17
2.1.5	Délai moyen de paquet.....	18
2.1.6	Coût des liaisons.....	20
2.2	Formulation et analyse du problème.....	21
2.3	Décomposition en sous-problèmes.....	24
2.3.1	Routage des paquets et affectation de flots.....	24
2.3.2	Affectation de capacités.....	25
2.3.3	Affectation de flots et de capacités.....	28
2.4	Analyse des techniques de résolution.....	29
2.4.1	Méthodes conventionnelles.....	31
2.4.2	Approches d'Intelligence Artificielle (IA).....	34
2.5	Contributions spécifiques.....	36

CHAPITRE III: ROUTAGE DES PAQUETS

	DANS LES RÉSEAUX DISTRIBUÉS.....	39
3.1	Routage et modèle OSI.....	39
3.2	Caractérisation du flot de multicommodités.....	42
3.3	Caractérisation des procédures de routage.....	47
3.3.1	Routage statique.....	47
3.3.2	Routage adaptatif.....	48
3.3.3	Routage optimal.....	49
3.4	Nouvelle approche de routage basée sur les attributs de liaisons.....	55
3.4.1	Fondements de l'approche.....	56
3.4.2	Énoncé de l'algorithme.....	60
3.4.3	Complexité algorithmique.....	63

3.4.3.1	Complexité au meilleur cas.....	63
3.4.3.2	Complexité au pire cas.....	64
3.4.4	Détails d'implémentation.....	66
3.5	Caractéristiques des réseaux à haut débit.....	67
3.5.1	Exigences du haut débit.....	68
3.5.2	Latence et délai de propagation des paquets.....	69
3.5.3	Détermination du délai de paquet.....	71
3.5.4	Contrôle de flot et de congestion.....	72
3.6	Routage à haut débit.....	73
3.6.1	Impact du débit élevé.....	74
3.6.2	Adaptabilité de notre approche de routage.....	76
3.7	Résultats de mise en oeuvre.....	77
3.7.1	Résultats obtenus sur des réseaux à débit moyen.....	78
3.7.2	Résultats obtenus à haut débit.....	87
3.8	Analyse des résultats et conclusion.....	89

CHAPITRE IV: INTÉGRATION DES ASPECTS

	DE ROUTAGE ET DE SURVIVABILITÉ.....	94
4.1	Fiabilité et survivabilité des réseaux.....	95
4.1.1	Considérations de fiabilité.....	95
4.1.2	Considérations de survivabilité.....	97
4.2	Survivabilité et routage.....	98
4.2.1	Hypothèses.....	98
4.2.2	Modèle d'intégration.....	100
4.3	Nouvelle méthode d'intégration de routage et de survivabilité.....	104

4.3.1	Variation de flot.....	104
4.3.2	Affectation de capacités.....	110
4.4	Implémentation et mise en oeuvre.....	112
4.4.1	Étapes d'implémentation.....	112
4.4.2	Procédures de mise en oeuvre.....	115
4.5	Analyse des résultats.....	117
4.5.1	Comportement d'un réseau à haut débit en cas de défaillance d'une liaison.....	121
4.5.2	Comparaison de résultats.....	127
CHAPTER V: CONCLUSION.....		137
5.1	Contributions et synthèse des résultats.....	137
5.2	Limitations des travaux et orientations de recherches.....	139
BIBLIOGRAPHIE.....		141

LISTE DES TABLEAUX

	Page
2.1 Options de capacités et leurs coûts respectifs.....	21
3.1 Coordonnées cartésiennes des noeuds.....	79
3.2 Options de capacités.....	79
3.3 Résultats comparatifs de routage pour un trafic de 10 paq/s.....	80
3.4 Résultats comparatifs de routage pour un trafic de 15 paq/s.....	81
3.5 Résultats comparatifs de routage pour un trafic de 20 paq/s.....	81
3.6 Résultats comparatifs de routage pour un trafic de 25 paq/s.....	81
3.7 Résultats comparatifs de routage pour un trafic de 30 paq/s.....	82
3.8 Résultats comparatifs de routage pour un trafic de 35 paq/s.....	82
3.9 Résultats comparatifs de routage pour un trafic de 40 paq/s.....	82
3.10 Résultats comparatifs de routage pour un trafic de 45 paq/s.....	83
3.11 Résultats comparatifs de routage pour un trafic de 50 paq/s.....	83
3.12 Matrice des options de capacités.....	84
3.13 Comparaison des algorithmes de routage.....	86
3.14 Options de capacités à haut débit.....	87
3.15 Attributs de liaisons à haut débit.....	88
4.1 Représentation des chemins.....	108
4.2 Variation de flot après défaillance de la liaison (4, 5).....	109
4.3 Variation de flot après défaillance de la liaison (2, 3).....	110

4.4	Attributs de liaisons sans considération de survivabilité.....	118
4.5	Attributs de liaisons avec considération de survivabilité (État E_0).....	119
4.6	Flot et délai après défaillance respective des liaisons (1, 5) et (9, 13).....	120
4.7	Numérotation des liaisons.....	122
4.8	Comparaison de délai du réseau GTE (Cas 1).....	131
4.9	Comparaison de délai du réseau USA pour $1/\mu = 430$ bits/paq (Cas 1).....	131
4.10	Options de capacités des réseaux GTE et USA.....	132
4.11	Capacités de liaisons en kbps des réseaux GTE et USA.....	132
4.12	Comparaison de délai du réseau GTE (Cas 2).....	134
4.13	Comparaison de délai du réseau USA pour $1/\mu = 430$ bits/paq (Cas 2).....	135
AI.1	Résultats comparatifs de routage pour un trafic de 10 paq/s.....	159
AI.2	Résultats comparatifs de routage pour un trafic de 15 paq/s.....	159
AI.3	Résultats comparatifs de routage pour un trafic de 20 paq/s.....	160
AI.4	Résultats comparatifs de routage pour un trafic de 25 paq/s.....	160
AI.5	Résultats comparatifs de routage pour un trafic de 30 paq/s.....	160
AI.6	Résultats comparatifs de routage pour un trafic de 35 paq/s.....	161
AI.7	Résultats comparatifs de routage pour un trafic de 40 paq/s.....	161
AI.8	Résultats comparatifs de routage pour un trafic de 45 paq/s.....	161
AI.9	Résultats comparatifs de routage pour un trafic de 50 paq/s.....	162
AI.10	Coordonnées des noeuds.....	162
AI.11	Attributs de liaisons selon les trois heuristiques.....	163

AI.12	Évolution des paramètres des algorithmes de routage optimal.....	164
AI.13	Attributs de liaisons proches de l'optimalité des algorithmes de routage optimal.....	165
AI.14	Coordonnées des noeuds du réseau	165
AII.1	Détermination du chemin principal et du chemin de secours.....	183

LISTE DES FIGURES

	Page
1.1 Structure d'un réseau téléinformatique typique.....	2
2.1 Matrice de trafic d'un réseau de 6 noeuds.....	15
2.2 Formulation générale du problème de conception topologique	22
2.3 Formulation du problème de routage.....	25
2.4 Formulation du problème d'affectation de capacités.....	27
2.5 Formulation du problème d'affectation de flots et de capacités.....	30
3.1 Le modèle de référence OSI.....	41
3.2 Sous-réseau de transport (6 noeuds, 7 liaisons).....	43
3.3 Conservation de flot au noeud 1.....	43
3.4 Espace convexe.....	45
3.5 Fonction convexe.....	46
3.6 Sous-réseau de transport (10 noeuds, 14 liaisons).....	79
3.7 Variations du délai moyen de paquet en fonction du niveau de trafic.....	84
3.8 Sous-réseau de transport (8 noeuds, 12 liaisons).....	85
3.9 Configuration d'un réseau à haut débit (20 noeuds, 30 liaisons).....	89
4.1 Réseau avec liaisons défaillantes (7 noeuds, 8 liaisons).....	107
4.2 Réseau à haut débit avec liaisons défaillantes (20 noeuds, 30 liaisons).....	116
4.3 Variation relative de flot sur les liaisons 1, 2, 3.....	123
4.4 Variation moyenne de la charge du réseau.....	125

4.5	Variation du délai moyen de paquet.....	126
4.6	Réseau GTE (12 noeuds, 25 liaisons).....	128
4.7	Réseau USA (26 noeuds, 41 liaisons).....	129
AII.1	Variation relative de flot sur les liaisons 4, 5, 6.....	189
AII.2	Variation relative de flot sur les liaisons 7, 8, 9.....	189
AII.3	Variation relative de flot sur les liaisons 10, 11, 12.....	190
AII.4	Variation relative de flot sur les liaisons 13, 14, 15.....	190
AII.5	Variation relative de flot sur les liaisons 16, 17, 18.....	191
AII.6	Variation relative de flot sur les liaisons 19, 20, 21.....	191
AII.7	Variation relative de flot sur les liaisons 22, 23, 24.....	192
AII.8	Variation relative de flot sur les liaisons 25, 26, 27.....	192
AII.9	Variation relative de flot sur les liaisons 28, 29, 30.....	193

LISTE DES ANNEXES

	Page
ANNEXE I: RÉSULTATS DÉTAILLÉS DU CHAPITRE III.....	151
AI.1 Principales fonctions de la nouvelle stratégie de routage.....	151
AI.2 Résultats comparatifs des heuristiques.....	159
AI.3 Résultats détaillés obtenus des algorithmes de routage.....	162
AI.4 Coordonnées des noeuds du réseau à haut débit.....	165
 ANNEXE II: RÉSULTATS DÉTAILLÉS D'INTÉGRATION DES ASPECTS	
DE ROUTAGE ET DE SURVIVABILITÉ.....	167
AII.1 Principales fonctions d'intégration de routage et de survivabilité.....	167
AII.2 Chemin principal et chemin de secours de chaque paire de noeuds	183
AII.3 Comportement des liaisons d'un réseau à haut débit	
en cas de défaillance.....	188

CHAPITRE I

INTRODUCTION

Durant la dernière décennie, les réseaux de télécommunications se sont développés à un rythme vertigineux. Cette croissance est due non seulement à un besoin d'échanger des informations, mais aussi à un besoin de consulter à distance des bases de données souvent réparties sur plusieurs ordinateurs, ou encore de partager la puissance de calcul de processeurs spécialisés. Ce mémoire porte précisément sur les *réseaux téléinformatiques*, en mettant l'emphasis sur les aspects de routage et de survivabilité en conception de réseaux à haut débit. En guise d'introduction, ce chapitre présente un sommaire des divers aspects de tels réseaux. Dans un premier temps, nous définirons les concepts de base du domaine, démarche essentielle à une bonne compréhension et à une modélisation adéquate d'un réseau à haut débit. Dans un deuxième temps, nous esquisserons des éléments de la problématique qui nous préoccupe, puis formulerons les objectifs de recherche. Pour finir, nous préciserons la nature des résultats escomptés, avant d'indiquer les grandes lignes du mémoire.

1.1 Définitions et concepts de base

Un *réseau téléinformatique* est un ensemble d'équipements informatiques ou noeuds reliés entre eux par des liaisons de communications (paires torsadées, câbles coaxiaux, fibre optique, satellite, faisceaux hertziens) et permettant l'échange d'informations entre utilisateurs géographiquement dispersés (Pierre, 1991). Un tel réseau effectue essentiellement deux fonctions: le *transport d'informations* et le *traitement* de ces mêmes informations. Ceci amène à considérer un réseau téléinformatique typique comme une structure hiérarchisée à deux niveaux: un *sous-réseau de transport*, et un ou plusieurs *réseaux locaux* permettant l'accès aux ressources disponibles sur l'ensemble de l'infrastructure. La Figure 1.1 en donne une illustration.

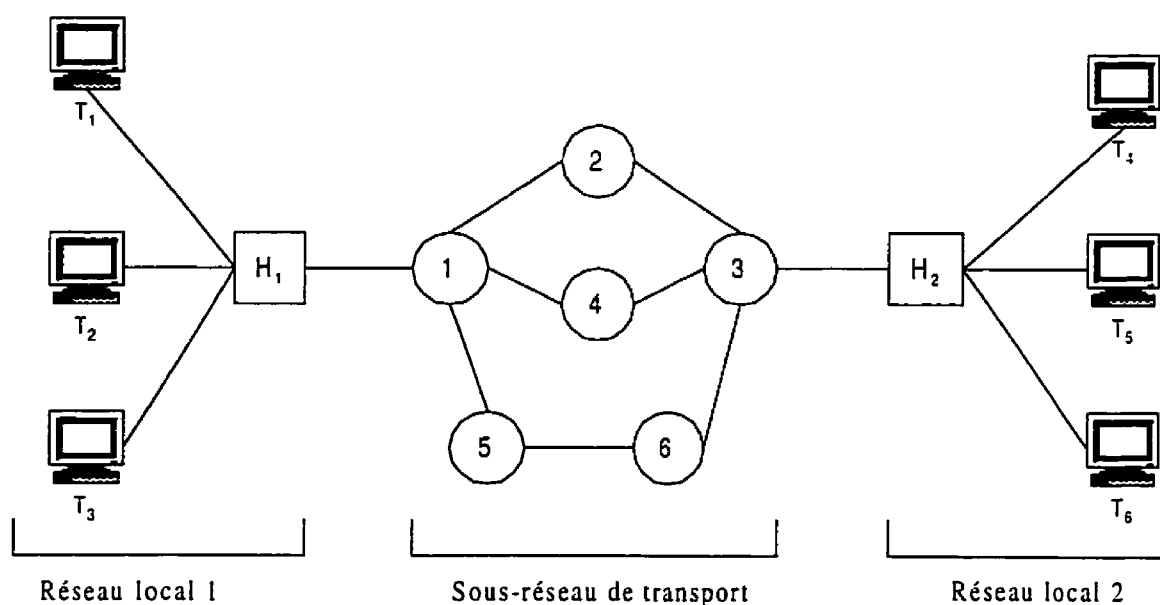


Figure 1.1 Structure d'un réseau téléinformatique typique

De par sa structure et pour des raisons historiques, un réseau d'accès local, ou tout simplement un réseau local a toujours été considéré comme un système centralisé, dans le sens qu'un ordinateur central se charge, non seulement de distribuer les droits de transmission, mais aussi d'effectuer les traitements exigés par les stations ou postes de travail jouant le rôle de terminaux d'accès. Chaque terminal T_i constitue un équipement d'entrée-sortie qui facilite le dialogue entre l'ordinateur central et un utilisateur du réseau. Cet ordinateur central, communément appelé *hôte* et noté H_i , se charge de transmettre les informations d'un noeud à un autre, permettant ainsi aux utilisateurs d'accéder aux ressources du réseau.

Quant au sous-réseau de transport, il constitue un exemple typique de système distribué. En effet, il n'y existe pas de hiérarchie formelle entre les différents noeuds, par le fait même que ces derniers possèdent des fonctions et des positions hiérarchiquement équivalentes. Ainsi, les commutateurs ou noeuds de commutation déterminent collectivement l'ordre dans lequel ils peuvent émettre. Un tel réseau relie deux ou plusieurs ordinateurs hôtes et achemine les informations sous forme de séquences binaires entre utilisateurs de réseaux locaux distants. Généralement, il est modélisé par un graphe non orienté $G = (N, A)$, où N désigne l'ensemble des noeuds ou commutateurs et A l'ensemble des liaisons de communications.

Supposons qu'un utilisateur du poste de travail T_i de la figure 1.1 veuille communiquer avec celui du poste T_j . Alors, H_i vérifie que T_i a le droit d'émettre et transmet l'information à véhiculer au noeud 1 du sous-réseau de transport. Ce dernier se charge du transport de bout en bout de cette information qui doit se rendre au prochain

ordinateur hôte H_2 , avant d'être acheminée vers la station T_4 . Il convient de mentionner que la technologie utilisée pour transférer les informations d'un bout à l'autre du sous-réseau de transport joue un rôle prépondérant dans la transmission des messages. On parle alors de *techniques de commutation*.

Il existe plusieurs techniques de commutation (Pierre, 1995). La *commutation de circuit* consiste essentiellement à déterminer préalablement un chemin physique ou circuit entre un noeud émetteur et un noeud récepteur pour la transmission d'un message (Girard, 1990). Autrement dit, pendant la transmission, un circuit est réservé au message en question, de sorte qu'aucune autre communication ne puisse l'emprunter. Un tel procédé fait perdre le bénéfice de transmettre simultanément d'autres messages par ce circuit, ce qui conduit à une sous-utilisation du sous-réseau de transport.

La *commutation de messages* consiste essentiellement à acheminer les messages graduellement de noeud en noeud, d'une source à une destination, sans réservation préalable de chemin physique entre l'émetteur et le récepteur. Par cette technique, un message provenant d'un commutateur doit être correctement reçu et stocké au noeud intermédiaire, avant d'être acheminé au commutateur suivant. De toute évidence, un tel procédé risque de saturer la mémoire tampon des noeuds intermédiaires et de ralentir considérablement la vitesse de transmission des messages.

Pour y remédier, on a pensé à découper chaque message en segments de taille plus petite, appelés *paquets*, munis d'une adresse de destination et d'un numéro de séquence. Ces paquets sont alors transmis indépendamment de noeud en noeud au travers du sous-réseau de transport, jusqu'à la destination où ils sont regroupés pour former le message

originel. C'est le principe de la *commutation de paquets*. Ainsi, le *sous-réseau de transport* constitue un réseau distribué à commutation de paquets.

L'émergence des réseaux à haut débit et le besoin d'une communication de plus en plus rapide ont conduit à deux nouvelles techniques: la commutation par *relais de trames* et la commutation par *relais de cellules*. La première consiste à découper l'information à transmettre en blocs de taille variable, appelés *trames* et munis d'un numéro de voie logique et des indications de service (Pierre, 1995). Cette approche suppose que les liaisons de transmission sont très fiables et que les trames ne sont ni corrompues, ni perdues, de sorte que le contrôle d'erreur ne s'effectue qu'aux noeuds source et destination, ce qui diminue évidemment le temps de traitement aux noeuds intermédiaires.

Quant à la commutation par relais de cellules, elle constitue une forme hybride de commutation de circuits et de commutation de paquets (Pierre, 1995). En effet, par cette technique, le message est découpé en blocs de taille fixe et restreinte, appelés *cellules*, ce qui permet d'augmenter la rapidité de la commutation. Cette technique est bien adaptée aux transfert d'informations de type varié (données, son, images fixes ou animées) et nécessitant un débit élevé. La commutation par relais de cellules est principalement implantée dans les RNIS à large bande. Nonobstant les différences intrinsèques entre les trois concepts (paquets, cellules, trames), le terme *paquet* sera retenu par la suite pour désigner indifféremment aussi bien les trames que les cellules.

La *configuration topologique* d'un réseau téléinformatique se réfère à l'emplacement géographique des noeuds et des liaisons d'un tel réseau. La détermination

de l'ensemble des liaisons interconnectant ces noeuds au meilleur coût, en respectant une série de contraintes, fait l'objet du *dimensionnement* ou de la *conception topologique* du réseau. Cette dernière fait appel à un certain nombre de paramètres qu'il importe de définir.

Alors que la quantité maximale d'informations pouvant être acheminée sur une liaison constitue sa *capacité*, le *flot* de cette liaison représente la quantité effective d'informations qui y circule par unité de temps. D'autre part, le *trafic* entre une paire de noeuds désigne le nombre moyen de paquets par unité de temps échangés entre ces noeuds. Un *chemin* représente une séquence de liaisons partant d'une source pour aboutir à une destination. Une *stratégie de routage* détermine alors un ensemble de règles permettant d'acheminer les paquets entre chaque source et chaque destination, ce qui conduit à la détermination du *délai moyen* d'un paquet, c'est-à-dire le temps moyen pris par ce paquet pour traverser le sous-réseau de transport. Quant à la *survivabilité* d'un réseau, elle se réfère à la capacité de ce réseau à maintenir ses performances en termes de qualité de service lors d'une éventuelle panne d'un noeud ou d'une liaison. On parle de *reroutage* de paquets quand il s'agit de rediriger par d'autres chemins tous les paquets devant transiter par un chemin dont une liaison devient défectueuse.

1.2 Éléments de problématique

La conception topologique des réseaux distribués constitue un problème fort complexe dont la solution dépend largement de l'algorithme de routage considéré. Ce

dernier doit être particulièrement robuste pour à la fois éviter la congestion du réseau et minimiser le délai de chaque message. Toutefois, la conception d'un algorithme performant de routage demeure un problème complexe d'optimisation d'une fonction non linéaire, en l'occurrence le délai moyen des paquets, sujet éventuellement à un ensemble de contraintes. En effet, pour un réseau de n noeuds et pour des raisons qui seront exposées plus loin dans ce mémoire, ce problème induit $n^2(n - 1)$ équations de conservation de flot (Schwartz, 1987). L'objectif d'une stratégie de routage consiste alors à minimiser le délai moyen en respectant toutes ces contraintes et tel que le flot de chaque liaison soit positif et inférieur à sa capacité, ce qui se révèle particulièrement laborieux à résoudre.

Il existe toutefois des méthodes exactes de programmation mathématique qui se basent sur certaines propriétés de la fonction de délai moyen pour résoudre le problème de routage à l'optimum (Bertsekas & Gallager, 1987; Fratta *et al.*, 1974). Néanmoins, la mise en oeuvre de ces méthodes nécessite des efforts de calcul considérables, ce qui amène à recourir à des heuristiques de routage permettant de déterminer en des temps de calcul raisonnables les chemins des paquets, en respectant évidemment toutes les contraintes.

Par ailleurs, les caractéristiques spécifiques des réseaux à haut débit rendent le problème de routage encore plus difficile, puisqu'il convient de modéliser adéquatement les liaisons de tels réseaux lors la conception de l'algorithme de routage. Si maintenant nous devons considérer que, pendant l'acheminement des paquets, une liaison quelconque du réseau peut aléatoirement tomber en panne, le problème se complique encore

davantage. En effet, la défaillance d'une liaison entraîne le reroutage de tous les paquets devant transiter par les chemins utilisant cette liaison. Ces paquets doivent alors emprunter d'autres chemins pour se rendre à leur destination respective (Gavish & Neuman, 1992). Il importe alors de déterminer ces chemins de façon à éviter la congestion des liaisons qui en font partie et à maintenir les performances du réseau.

À l'analyse, les méthodes exactes de programmation mathématique se révèlent inadéquates pour résoudre le problème dans sa globalité, tenant compte à la fois du débit élevé et de la possibilité de défaillance de certaines composantes. En outre, la littérature intégrant le routage à haut débit et la survivabilité est très limitée: certains chercheurs (Baransel *et al.*, 1995; Khasnabish, 1993) mettent seulement l'emphasis sur le routage à haut débit, alors que d'autres comme Gavish et Neuman (1992) traitent du routage avec possibilité de défaillance d'une liaison sans toutefois tenir compte de la possibilité de débit élevé des liaisons. Il s'avère donc opportun d'intégrer une méthode efficace de survivabilité à la modélisation des liaisons de haut débit lors de la conception d'un algorithme de routage. C'est là l'orientation que nous privilégions dans ce mémoire.

1.3 Objectifs de recherche

Les procédures de routage peuvent se classer en trois catégories: les stratégies de routage statique, celles de routage adaptatif et celles de routage optimal. Malgré leur simplicité d'implémentation et leur rapidité d'exécution, les heuristiques de routage statique ne tiennent nullement compte du niveau de trafic dans le réseau (Schwartz &

Stern, 1980) et, de ce fait, ne garantissent pas une bonne solution au problème. D'autre part, les procédures de routage adaptatif tiennent compte du niveau de trafic, mais la mise en mémoire en chaque noeud de l'état courant du réseau peut conduire à la saturation éventuelle du tampon de ces noeuds (Kershenbaum, 1993). Quant aux algorithmes de routage optimal, ils ne sont pratiquement pas utilisés, à cause de leur complexité et de leur temps d'exécution relativement long. Tout ceci nous amène à nous demander s'il est possible de concevoir un algorithme de routage qui, tout en étant simple à implémenter et rapide d'exécution, tient compte du niveau de congestion de chaque liaison pour opérer de façon efficace et produire une solution proche de l'optimum. Il s'agit là de notre premier objectif de recherche.

La considération de possibilité de défaillance amène à déterminer une bonne méthode de reroutage des paquets, méthode qui calcule entre autres la variation de flot des liaisons non défaillantes. Dans cet ordre d'idée, Gavish et Neuman (1992) ont proposé une méthode de calcul de variation de flot puissante et générale. Toutefois, les caractéristiques intrinsèques d'une telle méthode donnent lieu à des expressions complexes de variation de flot et peuvent dans certains cas conduire à l'interruption de la communication entre certaines paires de noeuds (Gavish & Neuman, 1992). Nous visons en conséquence à concevoir une nouvelle méthode de survivabilité qui permet d'abord de choisir les capacités pouvant absorber la variation de flot induite par la défaillance, ensuite de dévier habilement le flot de la liaison en panne et de déterminer par une formule simple la variation de flot sur les liaisons non défaillantes. Une telle méthode

peut être intégrée à notre algorithme de routage pour donner lieu à une stratégie efficace d'intégration des aspects de routage et de survivabilité.

1.4 Résultats escomptés

Nous pouvons escompter plusieurs séries de résultats. D'abord, par rapport au problème de base, nous nous attendons à produire une solution sous-optimale. Par contre, l'efficacité de calcul de notre stratégie peut s'avérer satisfaisante, ce qui nous amène à penser que, d'une part, pour un même temps d'exécution, notre stratégie de routage peut conduire à de meilleurs résultats que ceux fournis par d'autres heuristiques connues et que, d'autre part, pour un temps d'exécution nettement plus faible, les résultats de notre stratégie peuvent bien se comparer à ceux des algorithmes classiques de routage optimal. En ce qui a trait au routage à haut débit, nous nous attendons à obtenir des résultats qui tiennent compte des caractéristiques intrinsèques des liaisons de haut débit. En dernier lieu, nous visons à mettre en oeuvre une méthode d'intégration de routage et de survivabilité qui, en plus de pouvoir s'adapter aux réseaux à haut débit, exploite certaines faiblesses de la méthode de Gavish et Neuman (1992) pour produire des résultats qui lui sont à toutes fins pratiques comparables.

1.5 Plan du mémoire

La suite du mémoire est organisée en quatre chapitres. Le chapitre 2 établit une revue sélective de la littérature du domaine de la conception topologique, en mettant

l'accent sur les concepts fondamentaux et les paramètres importants à considérer. Cela permet de formuler le problème de conception topologique de la façon la moins restrictive possible, d'entamer la décomposition du problème global en sous-problèmes par un choix judicieux de paramètres à considérer, d'analyser les techniques de résolution les plus répandues et d'établir nos contributions à la littérature.

Le chapitre 3 traite du routage des paquets au travers d'un réseau distribué et présente une nouvelle stratégie basée sur certains attributs de liaisons et sur la notion de délai de chemin, et qui, dans ses principales caractéristiques, se situe dans la lignée des algorithmes de *déviations de flot* (Fratta *et al.*, 1974) et de *Bertsekas-Gallager* (1987). Les principes de base d'un tel algorithme y sont établis, sa complexité évaluée et les résultats d'implémentation analysés, ce qui permet de juger de sa performance et de son efficacité en termes de temps de calcul et d'espace mémoire requis pour sa mise en oeuvre. Ce chapitre caractérise aussi les réseaux à haut débit et, à partir de l'analyse de l'impact du débit élevé sur le routage, montre que notre algorithme peut s'adapter à ce type de réseaux. Les détails d'implémentation et de résultats de mise en oeuvre y sont également présentés.

Le chapitre 4 considère un modèle de réseau plus réaliste et généralise le problème de routage, en y intégrant l'aspect de survivabilité. Il présente les caractéristiques spécifiques d'un modèle général d'intégration, avant d'introduire une nouvelle méthode basée sur la notion de chemins disjoints. Il en décrit alors les différentes étapes d'implémentation, ainsi que les procédures de mise en oeuvre.

L'analyse des résultats et une comparaison de ces résultats avec ceux fournis par d'autres méthodes y sont aussi entreprises.

En guise de conclusion, nous synthétisons au chapitre 5 les résultats obtenus. Cette synthèse est complétée par une discussion sur les limites des méthodes proposées et par quelques suggestions d'orientations de recherches.

CHAPITRE II

CONCEPTION TOPOLOGIQUE

DES RÉSEAUX DISTRIBUÉS

La conception topologique des réseaux distribués a souvent été abordée selon deux types d'approche: l'approche *heuristique* qui donne lieu en principe à des solutions sous-optimales et que l'on peut appliquer à des réseaux de grande taille; et l'approche classique basée sur la *programmation mathématique* qui, en général, conduit à une solution optimale, mais requiert beaucoup de temps de calcul quand elle s'applique à des réseaux de grande taille. Ce chapitre présente le problème dans son ensemble, par une revue sélective de la littérature du domaine. Dans un premier temps, nous expliciterons les paramètres fondamentaux qui servent à formuler le problème de la façon la moins restrictive possible. Puis, nous entamerons la décomposition du problème global en sous-problèmes par un choix judicieux de paramètres à considérer. Cela nous conduira finalement à exposer et à analyser les techniques de résolution les plus répandues, et à établir nos contributions à la littérature de la conception topologique.

2.1 Paramètres fondamentaux

La conception topologique se réfère à un certain nombre de paramètres qui représentent des données du problème, des contraintes à satisfaire ou des variables à

optimiser. Ce sont: les attributs de liaisons, les exigences de trafic, la fiabilité, la stratégie de routage, le délai moyen des paquets et le coût total des liaisons. Nous nous proposons dans ce qui suit d'explicitier ces paramètres.

2.1.1 Attributs de liaisons

Chaque liaison l comporte un certain nombre d'attributs qui seront utilisés pendant tout le processus de conception. Ce sont (Pierre, 1994):

- le flot f_l qui désigne la quantité effective d'informations circulant sur cette liaison;
- la capacité C_l qui désigne la quantité maximale d'informations que peut acheminer cette liaison, avec $f_l \leq C_l$;
- l'utilisation U_l de la liaison l qui représente le rapport du flot à la capacité;
- le coût $d_l(C_l)$ de la liaison l qui est une fonction de la capacité de cette liaison.

2.1.2 Exigences de trafic

Dans un réseau à commutation de paquets, le trafic entre une paire de noeuds i et j correspond au nombre de paquets échangés entre ces noeuds par unité de temps. De ce fait, le trafic entre toutes les paires de noeuds se traduit par une matrice dite de trafic et notée Γ , dont chaque élément γ_{ij} représente le nombre moyen de paquets par seconde échangés entre le noeud i et le noeud j . Toutefois, cette matrice est inconnue et ne peut être estimée de façon précise lors de la conception du réseau. En effet, elle dépend de

plusieurs paramètres du réseau, difficiles à évaluer et susceptibles d'être modifiés avec le temps. Ainsi, certains chercheurs (Kershenbaum, 1993; Legault, 1994), pour déterminer cette matrice, supposent que le trafic entre deux sites est inversement proportionnel à la distance séparant ces deux sites. D'autres (Gerla & Kleinrock, 1977; Pierre, 1991) définissent cette matrice aléatoire par:

$$\Gamma = \rho \bar{\Gamma} \quad (2.1)$$

où $\bar{\Gamma}$ représente une matrice de base connue et ρ un facteur d'échelle variable dépendant du niveau de trafic. Toutefois, dans certains cas, la conception topologique est insensible aux variations du trafic, ce qui justifie l'utilisation du trafic moyen dans la plupart des problèmes de conception. En ce qui nous concerne, nous supposons que cette matrice est symétrique et donnée. La Figure 2.1 constitue une illustration d'une matrice de trafic associée à un réseau de 6 noeuds.

$$\Gamma = \begin{bmatrix} 0 & 40 & 15 & 10 & 20 & 25 \\ 40 & 0 & 30 & 20 & 25 & 30 \\ 15 & 30 & 0 & 15 & 20 & 15 \\ 10 & 20 & 15 & 0 & 15 & 20 \\ 20 & 25 & 20 & 15 & 0 & 25 \\ 25 & 30 & 15 & 20 & 25 & 0 \end{bmatrix}$$

Figure 2.1 Matrice de trafic d'un réseau de 6 noeuds

2.1.3 Fiabilité

La fiabilité d'un réseau fait appel à plusieurs concepts de la théorie des graphes. En effet, le *degré* d'un graphe $G = (N, A)$, noté d_G , peut se déterminer à partir du degré $d(i)$ de chaque noeud du graphe, le degré d'un noeud i désignant le nombre de liaisons incidentes à ce noeud. Le degré du graphe d_G est alors donné par:

$$d_G = \min_{i \in N} [d(i)] \quad (2.2)$$

Par ailleurs, la *connexité* se réfère à la notion de chemin. Deux chemins sont dits *disjoints d'arcs* s'ils n'ont aucun arc en commun; ils sont dits *disjoints de noeuds* s'ils n'ont aucun noeud en commun, sauf le noeud source et le noeud destination. De ce fait, il existe deux types de connexité: la connexité au sens des noeuds et la connexité au sens des arcs. Un graphe est dit *K-connexe* au sens des arcs si toutes les paires de noeuds sont reliées par K chemins disjoints d'arcs. La connexité au sens des arcs d'un réseau se note C_a . De même, ce graphe est dit *K-connexe* au sens des noeuds si toutes les paires de noeuds sont reliées par K chemins disjoints de noeuds. La connexité au sens des noeuds se note alors C_n . On peut alors montrer que:

$$C_n \leq C_a \leq d_G$$

Cette relation implique que la connexité au sens des noeuds est plus sévère que les deux autres paramètres. D'où l'opportunité d'adopter celle-ci comme mesure de fiabilité (Pierre, 1991).

De ce fait, une structure d'arbre de degré 1 constitue une solution économique, mais s'avère très peu fiable. Par contre, un réseau complet, malgré sa haute fiabilité,

conduit à des coûts prohibitifs dus à la redondance de ses liaisons. Ainsi, pour des réseaux de petite ou de moyenne taille (10 à 20 noeuds), on peut adopter la 2-connexité. Toutefois, si la taille du réseau devient grande, il est recommandé d'augmenter les exigences de fiabilité (la 3-connexité, par exemple).

À part la connexité, d'autres paramètres peuvent être aussi utilisés pour l'évaluation de la fiabilité. Il s'agit d'abord du *degré de connexité* des noeuds, paramètre qui détermine la stabilité physique du réseau: il désigne le nombre moyen de noeuds que l'on peut enlever sans compromettre le degré de connexité attendu de ce réseau. Quant au degré de connexité des liaisons, il représente la contribution moyenne de chaque liaison à la connexité du réseau (Newport & Varshney, 1989).

2.1.4 Stratégies de routage

L'opération par laquelle les paquets s'acheminent d'un noeud i à un noeud j , en choisissant un chemin parmi plusieurs, constitue le routage. Une bonne stratégie de routage utilise les éléments de la matrice de trafic, ainsi que la configuration topologique. Aussi doit-elle pouvoir rapidement s'adapter à tout changement du niveau de trafic dans le réseau, de façon à minimiser le délai moyen des paquets. La plupart des techniques actuelles de routage se basent sur le plus court chemin selon une métrique donnée pour sélectionner le chemin que chaque paquet doit emprunter.

Même s'il existe dans la littérature plusieurs stratégies de routage (Bertsekas & Gallager, 1987; Gavish & Neuman, 1989; Schwartz & Stern, 1980), la plus utilisée demeure le routage statique où les paquets sont acheminés sur des chemins statiques, ne

variant pas selon le trafic supporté par le réseau. En effet, cette stratégie est facile à implémenter et permet l'évaluation directe du flot et du délai. Le routage adaptatif, par contre, est complexe et requiert des simulations pour évaluer le flot et le délai. Toutefois, dans certains cas, les résultats déterminés selon le routage adaptatif sont proches de ceux obtenus selon le routage statique.

2.1.5 Délai moyen de paquet

Le délai moyen d'un paquet constitue une mesure importante de performance en conception topologique et représente le paramètre à optimiser lors de la conception d'un algorithme de routage. Il est donné par (Kleinrock, 1976):

$$T_{moy} = \sum_{\substack{i, j \\ i \neq j}} \frac{\gamma_{ij}}{\gamma} T_{ij} \quad (2.3)$$

où:

- γ_{ij} représente le trafic moyen entre les noeuds i et j ;
- T_{ij} le délai moyen d'attente et de transmission d'un paquet du noeud i au noeud j ;
- γ la charge totale du réseau.

L'application de la *règle de Little* à ce modèle conduit au résultat suivant:

$$T_{moy} = \sum_{l=1}^m \frac{\lambda_l}{\gamma} T_l \quad (2.4)$$

où:

- λ_l représente le nombre de paquets traversant la liaison l par unité de temps;
- m le nombre total de liaisons du réseau; et

- T_l le temps d'attente et de transmission d'un paquet sur la liaison l .

Cette expression, établie par Kleinrock (1976), est simple et générale. Toutefois, les termes λ_l et T_l ne peuvent être évalués qu'à partir d'un certain nombre d'hypothèses qui sont:

- arrivée de Poisson des paquets;
- longueur des paquets exponentiellement distribuée;
- capacité de la mémoire tampon des noeuds infinie;
- liaisons de transmission très fiables;
- temps de traitement des paquets négligeable;
- indépendance entre les temps interarrivées et les temps de service sur chaque liaison.

Ces hypothèses conduisent au *modèle de Jackson*, où chaque liaison se comporte comme une file d'attente de type $M/M/1$ (Kerashenbaum, 1993). Dans ce cas bien précis, le délai de transmission et d'attente d'un paquet sur la liaison l est donné par:

$$T_l = \frac{1}{\mu C_l - \lambda_l} \quad (2.5)$$

avec:

- $1/\mu$ la taille moyenne des paquets, en bits/paquet;
- C_l la capacité de la liaison l , en bits/seconde ou *bps*;
- λ_l le nombre de paquets par seconde traversant la liaison l .

D'autre part, le terme λ_l peut être exprimé en fonction de f_l , car:

$$f_l = \frac{\lambda_l}{\mu} \quad (2.6)$$

Combinant les relations (2.4), (2.5) et (2.6), nous aboutissons à l'expression suivante du délai moyen d'un paquet:

$$T_{\text{moy}} = \frac{1}{\gamma} \sum_{l=1}^m \frac{f_l}{C_l - f_l} \quad (2.7)$$

2.1.6 Coût des liaisons

Généralement, le coût total des liaisons représente la fonction à minimiser (fonction objective) en conception topologique et dépend essentiellement des capacités utilisées. Dans ce cas, puisque le coût de la liaison l de capacité C_l est $d_l(C_l)$, si le réseau comporte m liaisons, le coût total D des liaisons de communications est déterminé par :

$$D = \sum_{l=1}^m d_l(C_l) \quad (2.8)$$

L'efficacité des algorithmes de conception dépend en grande partie de la forme des $d_l(C_l)$ qui, dans la plupart des cas, sont des valeurs discrètes, dépendant des capacités de liaisons. Cependant, pour des raisons d'efficacité de calcul, il est commode d'approximer les coûts discrets par des valeurs continues durant une phase initiale d'optimisation pour finalement discrétiser ces valeurs durant les phases de raffinement. Le Tableau 2.1 illustre une table d'options de capacités et de leurs coûts respectifs.

Tableau 2.1. Options de capacités et leurs coûts respectifs

Capacité (kbps)	Coût fixe (\$/mois)	Coût variable (\$/mois/km)
9.6	650	0.40
19.2	850	2.50
50.0	850	7.50
100.0	1700	10.00
230.4	2350	30.00
460.8	4700	60.00
921.6	9400	120.00
1843.2	18800	240.00

2.2 Formulation et analyse du problème

Il existe plusieurs façons de formuler le problème de conception topologique. En général, ces formulations correspondent à des choix différents des mesures de performance, des variables de conception ou des contraintes. Nous adoptons à la Figure 2.2 une formulation très générale qui est équivalente à celle présentée par Pierre (1991).

Ce problème d'optimisation, tel que formulé, est véritablement un problème topologique, car la dimension et l'étendue du réseau influencent en grande partie son coût et sa performance. De ce fait, la localisation des commutateurs, que l'on peut déterminer à partir de cartes géographiques, représente les données de base du problème de dimensionnement. Il convient toutefois de mentionner que cette localisation peut être abordée différemment selon les chercheurs. En effet, certains concepteurs de réseaux utilisent le système de coordonnées cartésiennes pour localiser les noeuds, alors que

d'autres préfèrent l'utilisation de coordonnées géographiques, telles que la latitude et la longitude (Kershenbaum, 1993).

Étant donné:

- la localisation des commutateurs;
- le trafic moyen γ_{ij} entre chaque paire de noeuds (i, j) du réseau, avec $i, j = 1, 2, \dots, n$ et $i \neq j$;
- la matrice de coût (d_l) , $l = 1, 2, \dots, m$;

Minimiser

- le coût total D des liaisons de communications;

Sujet à

- la contrainte de fiabilité (K -connexité);
- la contrainte de délai moyen des paquets ($T_{moy} \leq T_{max}$);

Par choix

- de la configuration topologique;
 - des capacités de liaisons;
 - de la stratégie de routage.
-

Figure 2.2 Formulation générale du problème de conception topologique

Quant au trafic supporté par le réseau, il constitue aussi une donnée importante du problème, puisque la charge du réseau en dépend grandement. Par contre, il s'agit d'un paramètre très difficile à déterminer. En effet, le trafic moyen entre chaque paire de noeuds du réseau γ_{ij} varie de façon aléatoire selon le jour de la semaine et selon l'instant considéré de la journée. C'est d'ailleurs pourquoi, dans la plupart des cas, le réseau est conçu pour supporter le trafic moyen de l'instant où le trafic est le plus intense de la journée.

On se rend alors compte que rien qu'avec la localisation des commutateurs et la matrice de trafic, les complications sont telles que le problème d'optimisation paraît déjà très difficile. À cela, ajoutons le coût des liaisons de communications comportant un coût fixe et un coût variable dépendant de la longueur et de la capacité de chaque liaison. Et, puisque ces capacités sont discrètes, on obtient des coûts discrets, tel qu'il est illustré au Tableau 2.1.

De façon générale, on cherche à optimiser une fonction objective sous une certaine variable, en tenant compte des contraintes du problème de conception. La fonction objective correspond au coût total des liaisons de communications, alors que les contraintes se rapportent à la fiabilité et au délai T_{moy} de paquet. Ainsi, il n'est pratiquement pas possible d'optimiser à la fois le coût du réseau et le délai moyen des paquets. Toutefois, rien ne nous empêche d'établir un certain compromis entre ces deux paramètres ou de les traiter de façon séparée. En pratique, ce problème de planification s'avère non moins complexe à résoudre dans sa globalité. C'est d'ailleurs pourquoi la plupart des chercheurs procèdent à sa décomposition en différents sous-problèmes.

2.3 Décomposition en sous-problèmes

Une façon d'aborder le problème de conception topologique consiste à subdiviser le problème global en trois sous-problèmes: le routage (qui détermine l'affectation de flots aux liaisons), l'affectation de capacités, et l'affectation simultanée de flots et de capacités aux liaisons de communications. Cette section traite de ces différents sous-problèmes.

2.3.1 Routage des paquets et affectation de flots

Dans un réseau de n noeuds, il existe $n(n-1)$ paires de source/destination et chaque paire (i, j) représente une *commodité* k . Alors, si nous désignons par n_{sd} le nombre total de commodités, nous obtenons:

$$n_{sd} = n(n-1) \quad (2.9)$$

On se rend compte que chaque commodité k contribue au flot d'une liaison l du réseau. Par conséquent, si l'on note par f_l^k le flot dû à la commodité k sur la liaison l , alors le flot total de cette liaison constitue un flot de *multicommodités* et s'exprime par la relation:

$$f_l = \sum_{k=1}^{n_{sd}} f_l^k \quad (2.10)$$

Dans un réseau distribué, le routage des paquets est étroitement lié à l'affectation de flots aux liaisons de communications. En effet, un bon algorithme de routage cherche à déterminer le vecteur flot optimal f pour lequel le délai T_{moy} de chaque paquet est minimal. Ainsi, à partir du développement du modèle de Kleinrock (1976), le problème de routage se formule de la façon indiquée à la Figure 2.3.

Étant donné:

- la configuration topologique (n noeuds et m liaisons);
- la matrice de trafic $\Gamma = (\gamma_{ij})$;
- le vecteur des capacités de liaisons $C = (C_1, C_2, \dots, C_m)$;

Minimiser

- le délai moyen des paquets;

Sujet à

- $f = (f_1, f_2, \dots, f_m)$ est un vecteur flot de multicommodités;
- f satisfait aux exigences de la matrice de trafic Γ ;
- $f \leq C$;

Par choix de

- f .
-

Figure 2.3 Formulation du problème de routage

2.3.2 Affectation de capacités

L'affectation de capacités aux liaisons de communications constitue une phase importante de la conception d'un réseau, car elle permet de déterminer la capacité de chaque liaison qui, tout en permettant de respecter un seuil de délai maximal, minimise le

coût total de ce réseau. Une technique mathématique très puissante de résolution d'un tel problème d'optimisation avec contraintes demeure la *relaxation lagrangienne*. Cette dernière consiste à introduire un certain nombre de contraintes dans la fonction objective et donne lieu à une relaxation du problème original, ce qui aboutit à un nouveau problème appelé *sous-problème lagrangien*. Ce dernier intègre dans sa formulation un ensemble de variables, appelées *multiplieurs de Lagrange* qui, une fois déterminées, produit une borne inférieure du problème original. La formulation du problème d'affectation de capacités est présentée à la Figure 2.4.

Si l'on suppose que les capacités sont continues et qu'elles sont affectées aux liaisons de façon indépendante, alors le coût de la liaison l de capacité C_l sera linéaire et donné par:

$$d_l(C_l) = d_l^* C_l + d_{l0} \quad (2.11)$$

où d_{l0} représente le coût d'installation et d_l^* un coût de base variant selon la capacité de la liaison l . En outre, par la relaxation lagrangienne, la valeur optimale de la capacité de la liaison l est donnée par la relation (Gerla & Kleinrock, 1977):

$$C_l = f_l + \frac{\sum_{j=1}^m \sqrt{d_j f_j}}{\gamma T_{max}} \sqrt{\frac{f_l}{d_l}} \quad (2.12)$$

où γ désigne le trafic total supporté par le réseau et T_{max} le délai maximal admis. Nous en déduisons le coût optimal du réseau qui est alors donné par:

$$D = \sum_{l=1}^m \left[d_l f_l + d_{l0} + \frac{\left(\sum_{j=1}^m \sqrt{d_j f_j} \right)^2}{\gamma T_{\max}} \right] \quad (2.13)$$

Étant donné:

- la configuration topologique (n noeuds et m liaisons);
- la matrice de trafic I ;
- le vecteur flot f ;

Minimiser

- le coût total des capacités de liaisons;

Sujet à

- $T_{\text{moy}} \leq T_{\text{max}}$
- $f < C$;

Par choix de

- C .

Figure 2.4 Formulation du problème d'affectation de capacités

Si l'on devient plus réaliste en considérant des valeurs discrètes de capacités, c'est-à-dire que chaque liaison a une vitesse de transmission sélectionnée à partir d'un ensemble donné de capacités (Tableau 2.1), le problème devient alors beaucoup plus complexe et la relaxation lagrangienne ne peut plus s'y appliquer. Dans ce cas, on fait appel à des algorithmes de programmation dynamique. Deux de ces algorithmes sont présentés par Kershenbaum (1993).

Le premier algorithme, de complexité exponentielle, porte le nom de *fusion en série* (serial merge) et détermine des valeurs de capacités quasi-optimales. Deux concepts y sont introduits: la *dominance* et les *solutions partielles*. Un couple de délai-coût, noté (t_i, d_i) domine un autre couple (t_j, d_j) si l'on a: $t_i < t_j$ et $d_i < d_j$. Une solution partielle constitue alors un sous-ensemble de liaisons auxquelles sont assignées des capacités. Ainsi, la fusion en série se base sur la dominance parmi les solutions partielles et sur certaines contraintes du problème, telles que l'utilisation U_l et le délai moyen T_{moy} . Quant au second algorithme, il constitue une version améliorée de la relaxation lagrangienne en tentant, dans un premier temps, de minimiser le coût des liaisons sans tenir compte de la contrainte de délai, puis en choisissant parmi les capacités disponibles celles qui déterminent le meilleur délai. De complexité polynomiale, il est qualifié de *glouton* et, tout comme la fusion en série, produit une solution sous-optimale.

2.3.3 Affectation de flots et de capacités

Du fait de l'optimisation simultanée du flot et de la capacité de chaque liaison, ce problème admet plusieurs minima locaux qui rendent difficile l'obtention de la

solution exacte. Si l'on suppose que les coûts de liaisons sont linéaires, on peut déterminer pour un vecteur flot f une expression de la capacité optimale (à partir de la formulation de l'affectation de capacités) en termes de f . Dans ce cas, la fonction de coût $D(f)$ est concave et comporte plusieurs minima locaux. Une modification de l'algorithme de déviation de flot peut être utilisée pour déterminer un optimum local, optimum qui dépendra de la sélection du flot initial (Gerla & Kleinrock, 1977). Un meilleur estimé de la solution globale peut être obtenu en choisissant au hasard plusieurs valeurs de flot initial. Le problème d'affectation de flots et de capacités peut être formulé de la façon indiquée à la Figure 2.5.

2.4 Analyse des techniques de résolution

Puisque les méthodes exactes ne peuvent s'appliquer au dimensionnement de réseaux qui dépassent une certaine taille, on doit se tourner vers des heuristiques ou utiliser l'expérience du domaine pour obtenir des solutions sous-optimales. Analysons maintenant les *méthodes conventionnelles* de dimensionnement de réseaux, avant de considérer les récentes approches s'inspirant de l'*Intelligence Artificielle (IA)*.

Étant donné:

- la configuration topologique;
- la matrice de trafic Γ ;
- le coût des capacités de liaisons $d_l(C_l)$;

Minimiser

- le coût total des capacités;

Sujet à

- f est un vecteur flot de multicommodités;
- f satisfait aux exigences de la matrice de trafic Γ ;
- $T(f, C) \leq T_{max}$
- $f < C$;

Par choix de:

- f
 - C
-

Figure 2.5 Formulation du problème d'affectation de flots et de capacités

2.4.1 Méthodes conventionnelles

La structure générale d'opération d'une bonne heuristique de dimensionnement est la suivante: on génère une topologie initiale et l'on vérifie si cette topologie obéit aux contraintes de fiabilité et de délai. Si l'une de ces contraintes n'est pas respectée, on génère une autre topologie. Quand une topologie réalisable est obtenue, on calcule son coût. Et ce processus se répète jusqu'à obtenir un minimum local. La clé de développement d'une bonne heuristique demeure la compréhension de la structure du problème à résoudre, ce qui aide à identifier les caractéristiques d'une bonne solution, ainsi que les paramètres ayant le plus d'impact (les plus significatifs) sur cette solution.

La méthode d'*échange de liaisons* constitue l'une des premières heuristiques de dimensionnement de réseaux (Gerla *et al.*, 1974). Son principe de fonctionnement est simple: il consiste essentiellement à ajouter des liaisons au réseau, à en enlever, ou à effectuer les deux à partir d'une solution initiale, de façon à améliorer le coût de la topologie courante, jusqu'à atteindre un coût minimal. Cette méthode, quoique très simpliste, requiert une exploration exhaustive de tous les échanges possibles, consommant ainsi beaucoup de temps pour des réseaux de plus de 20 noeuds (Chattopadhyay *et al.*, 1989). De plus, le manque d'information sur les liaisons à échanger conduit à des perturbations arbitraires et la solution dépend de la topologie initiale générée. Ainsi, lorsque la topologie initiale est mauvaise, on a de fortes chances d'aboutir à une mauvaise solution. Dans le cas contraire, on peut trouver une solution proche de l'optimum.

Une transformation de l'*échange de branches* a amené à la méthode de *saturation de coupe*, proposée par Gerla (1974). Cet algorithme identifie d'abord les diverses coupes du réseau et tente d'optimiser à la fois son débit et son temps de réponse, par la saturation de ces coupes, ou par l'augmentation de la capacité des liaisons qui en font partie. De cette façon, la méthode de saturation de coupe aboutit à des solutions sous-optimales. Toutefois, il lui faut partir d'une bonne topologie initiale, ce qui n'est nullement garanti.

Il est clair que chacune de ces deux méthodes comporte des restrictions. La méthode d'échanges de liaisons constitue une approche que l'on pourrait qualifier d'*aveugle*, puisque le choix de la topologie initiale et des règles de perturbation n'est guidé par aucun principe solidement établi. Dans le même ordre d'idée, même si l'algorithme de saturation de coupe constitue une amélioration considérable d'une telle technique, le routage et, par conséquent, l'affectation de flots n'y sont que peu efficaces. De plus, une telle méthode s'applique seulement aux réseaux 2-connexes et suppose que toutes les liaisons comportent des capacités identiques, ce qui à notre avis manque de réalisme. Par ailleurs, il importe aussi de trouver un moyen efficace de générer de bonnes topologies initiales et, en considérant une plus grande classe de critères pour l'addition et le retrait de liaisons, en exécutant plus d'une insertion ou de retrait par itération, l'algorithme peut être modifié de manière à résoudre des problèmes plus généraux de conception topologique.

La méthode de saturation de coupe a toutefois inspiré certains chercheurs du domaine. Sapir (1978), dans son mémoire de maîtrise, s'est basé sur celle-ci pour développer un algorithme encore plus général de conception dont la solution finale est

indépendante de la topologie initiale. Quant à Newport et Varshney (1989), ils tiennent compte des conditions de survivabilité du réseau en se définissant des mesures de degré de connexité. Cette modification implique l'augmentation des procédures d'ajout et de retrait de liaisons et inclut des connaissances sur les indices de décomposition de noeud et de liaison.

D'autre part, Gerla (1973) a proposé l'algorithme *d'élimination de liaisons à coût concave* qui s'applique lorsque les coûts discrets peuvent être raisonnablement approximés par des courbes concaves. Cette méthode part d'une configuration topologique fortement connexe et élimine, à chaque itération, les liaisons les moins économiques jusqu'à trouver un minimum local. Toutefois, même si cette méthode élimine efficacement les liaisons les moins économiques, elle ne permet pas d'insérer de nouvelles liaisons et, de ce fait, peut conduire à une mauvaise solution.

Parallèlement, se sont développées des méthodes *stochastiques* que Zanakis et Evans (1981) ont adopté pour leur simplicité, leur vitesse d'exécution et leur robustesse. Toutefois, les résultats produits par de telles méthodes dépendent largement de la structure du modèle utilisé et sont sensibles à la taille du réseau. Chattopadhyay *et al.* (1989), quant à eux, combinent la méthode de *séparation et évaluation progressive* à un algorithme de théorie des graphes pour faire l'affectation de capacités aux liaisons. De son côté, Hyppolite (1994) applique le *recuit simulé* à la conception topologique. Cette technique produit de bons résultats, moyennant un choix minutieux des paramètres de recuit (Pierre *et al.*, 1995). Dans le même ordre d'idée, Legault (1994) a utilisé avec un certain succès un algorithme génétique pour résoudre le même problème. La porte est

donc bien ouverte pour un foisonnement d'approches s'inspirant des techniques d'*Intelligence Artificielle (IA)*.

2.4.2 Approches d'Intelligence Artificielle (IA)

Plus récemment, les chercheurs se sont tournés vers l'Intelligence Artificielle en vue d'améliorer les solutions émanant des techniques conventionnelles. Dans cet ordre d'idée, Kershenbaum (1993) applique la *logique floue* pour déterminer les règles de perturbations à partir d'une topologie initiale générée par l'algorithme *MENTOR* (MEsh Network Topology Optimization and Routing). La détermination de cette topologie initiale conduit au calcul des performances du réseau qui sont: le coût, la fiabilité et le délai. De ce calcul est construite une fonction floue recouvrant toutes les valeurs normalisées possibles des critères de performance. L'étape suivante s'occupe de faire le classement des topologies à partir des objectifs et des alternatives de décision, ce qui permet d'examiner l'impact des changements dans les critères de décision sur la topologie finale (Kershenbaum & Robinson, 1991). Il convient de mentionner que *MENTOR* s'adapte particulièrement aux réseaux utilisant des capacités relativement élevées et adopte plusieurs hypothèses de relaxation donnant lieu à beaucoup de simplifications, en particulier celle qui consiste à poser que le coût d'une liaison est une fonction croissante de sa longueur. En fait, chaque liaison a un coût fixe et un coût par unité de longueur qui n'est pas nécessairement et strictement proportionnel à sa longueur.

Par ailleurs, Dutta et Mitra (1993) ont proposé une méthode baptisée *approche hybride* qui combine la rigueur d'une approche exacte combinatoire à la flexibilité d'une

approche heuristique. En effet, cette méthode utilise des algorithmes puissants d'optimisation pour résoudre certains sous-problèmes et des heuristiques flexibles pour en résoudre d'autres qui ne peuvent être formulés de façon mathématique. Toutefois, pour conserver sa flexibilité, la composante heuristique doit être explicitée et séparée de la composante algorithmique de façon à raffiner convenablement le problème. Un *tableau noir* gère la structure globale du système et détient les structures de données communes accessibles par une série de modules, dont certains constituent des modules de reconnaissance de topologies, d'autres supportent un système de *maintenance de vérité* (TMS pour Truth Maintenance System) ou un système de *retour-arrière* (DDB pour Dependency Directed Backtracking), d'autres enfin s'occupent de la détection d'erreurs et de l'ordonnancement du tableau noir.

Quant à Pierre (1991), il a proposé un système intelligent, appelé *SIDRO* (Système Intelligent de Dimensionnement de Réseaux d'Ordinateurs), basé sur une hiérarchie de règles heuristiques et sur l'apprentissage inductif, et qui intègre quatre modules exécutant les tâches suivantes: interaction avec l'utilisateur, génération de topologie initiale, génération d'exemples et apprentissage inductif. Un tel système vise essentiellement à appliquer des transformations locales ou perturbations à une bonne topologie initiale dans le but de réduire le coût total des liaisons ou de ramener sous un certain seuil le délai moyen. Ces perturbations permettent d'ajouter, d'enlever ou de substituer des liaisons, et sont réalisées à partir d'opérateurs intégrés dans des règles dites de perturbation. La solution produite par *SIDRO* dépend de la dimension de l'espace de recherche implicitement contrôlé par l'ensemble des contraintes, plus spécifiquement par

la contrainte de délai. En effet, lorsque le délai maximal T_{max} spécifié par l'utilisateur est élevé et qu'un grand nombre de topologies satisfait cette contrainte, l'espace de recherche tend à s'agrandir. Dans le cas contraire, l'espace de recherche et le nombre de topologies faisables sont donc réduits.

Actuellement, les résultats montrent que les techniques d'IA sont les plus performantes dans la résolution du problème de conception topologique (Legault, 1994; Pierre, 1991). Mais, ce sont des techniques complexes qui font appel à des connaissances approfondies en *systèmes d'exploitation*, en *base de données*, voire en *systèmes à base de connaissances*, chacune de ces disciplines représentant un vaste domaine de recherche. Ainsi, dans la sélection d'une méthode ou d'un algorithme de dimensionnement, des critères comme l'efficacité de calcul, la flexibilité et la possibilité de solutions alternatives multiples sont souvent plus importants que la quasi-optimalité. Trois éléments clés doivent être considérés dans un tel processus: le choix d'une bonne topologie initiale, une stratégie efficace d'affectation de flots et de capacités, et un ensemble de critères à la fois efficaces et faciles à mettre en œuvre pour effectuer les perturbations.

2.5 Contributions spécifiques

La conception topologique de réseaux distribués constitue un problème fort complexe d'optimisation dont diverses avenues de solutions ont été présentées. La formulation générale adoptée à la Figure 2.2 minimise le coût total des liaisons de communications en respectant les contraintes de fiabilité et de délai. Dans ce mémoire,

nous mettons particulièrement l'emphasis sur le délai moyen T_{moy} de paquet, délai qui peut être optimisé par un algorithme efficace de routage (en tenant évidemment compte du fait que ce dernier constitue une variable du problème de conception topologique). Il s'avère alors important d'intégrer dans le processus de conception une bonne stratégie de routage qui, tout en étant simple à implanter et rapide d'exécution, tient compte du niveau de congestion de chaque liaison pour opérer de façon efficace et produire une solution proche de l'optimum en termes de délai.

Malgré leur simplicité d'implémentation et leur rapidité d'exécution, les stratégies de routage statique ne tiennent nullement compte du niveau de trafic dans le réseau (Schwartz & Stern, 1980) et, de ce fait, ne peuvent garantir une bonne solution à un tel problème. D'autre part, les procédures de routage adaptatif tiennent compte du niveau de trafic, mais nécessite la mise en mémoire en chaque noeud de l'état courant du réseau, ce qui conduit à la saturation éventuelle du tampon de ces noeuds (Kershenbaum, 1993). Quant aux algorithmes de routage optimal, ils ne sont pratiquement pas utilisés, à cause de leur complexité et de leur temps d'exécution relativement long. Tout ceci nous amène à concevoir une stratégie de routage axée sur certains attributs de liaisons et qui repose fondamentalement sur deux principes tout à fait originaux, permettant de réduire efficacement la congestion dans le réseau. La manière de déterminer les attributs de liaisons, ainsi que celle d'acheminer les paquets constituent autant de paramètres nouveaux qui déterminent l'efficacité d'un tel algorithme. De plus, son implémentation intègre un module d'affectation de capacités qui lui permet de résoudre conjointement les

problèmes d'affectation de flots et d'affectation de capacités, en cherchant à minimiser le délai moyen T_{moy} des paquets. Il s'agit là de notre première contribution à la littérature.

Notre seconde contribution résulte de la conception d'une nouvelle méthode de survivabilité des réseaux, méthode qui sera intégrée dans la stratégie de routage nouvellement conçue. Une telle méthode garantit la disponibilité d'au moins un chemin après la défaillance d'une quelconque liaison. De plus, elle intègre une stratégie efficace d'affectation de capacités qui s'assure que, par suite d'une panne, les liaisons non défaillantes soient en mesure d'absorber tout le trafic du réseau. Cette stratégie nous amène en outre à déterminer, par une formule plus compacte que celles présentées dans la littérature (Gavish & Neuman, 1992), la variation de flot causée par la panne en question sur les liaisons non défaillantes, ce qui nous permet d'étudier le comportement de chaque liaison après la défaillance.

Ainsi, nous apportons une double contribution par ce mémoire. Au chapitre 3, nous présenterons notre nouvelle stratégie de routage, alors qu'au chapitre 4, nous exposerons notre méthode d'intégration de routage et de survivabilité, méthode qui permettra de résoudre une nouvelle version du problème de routage.

CHAPITRE III

ROUTAGE DES PAQUETS

DANS LES RÉSEAUX DISTRIBUÉS

L'efficacité et la performance d'un réseau distribué dépendent en grande partie de l'algorithme de *routage* ou d'*acheminement* des paquets qui y est implanté. Un tel algorithme doit être conçu de façon à éviter la congestion du réseau et surtout à minimiser le délai de chaque message. Ce chapitre traite du routage des paquets au travers d'un réseau distribué. Dans un premier temps, il en définit les aspects fondamentaux, passe en revue les algorithmes existants et expose une nouvelle stratégie basée sur certains attributs de liaisons et sur la notion de délai de chemin. Dans un deuxième temps, il caractérise les réseaux à haut débit et, à partir de l'analyse de l'impact du débit élevé sur le routage, montre que notre stratégie peut s'adapter à ce type de réseaux. Il se termine par une analyse des résultats produits par le nouvel algorithme de routage, ce qui permet de conclure sur sa performance et son efficacité.

3.1 Routage et modèle OSI

Reprenons l'exemple de la Figure 1.1, en supposant qu'un message est émis du réseau local 1 au réseau local 2. Dans ce cas, H_1 achemine ce message au noeud 1 du sous-réseau de transport qui le découpe en paquets de taille variable (1000 à 5000 bits), et

munit chacun de ces paquets d'un numéro d'ordre et de l'adresse du noeud destination, en l'occurrence le noeud 3 qui donne accès au réseau local 2. Après ce traitement, le noeud 1 consulte une table dite d'acheminement contenant le prochain noeud à visiter par chaque paquet. Ainsi, chaque paquet utilise le chemin enregistré dans la table d'acheminement pour atteindre les prochains noeuds jusqu'à parvenir au noeud destination 3. L'ensemble de ces opérations constitue l'acheminement ou le routage des paquets.

Il convient de mentionner que l'échange d'informations entre deux noeuds représente une tâche fort complexe, pouvant être décomposée en plusieurs tâches élémentaires. Ce qui conduit à mettre en place une structure basée non seulement sur des principes d'ordre fonctionnel, mais aussi sur des concepts de hiérarchie et de décentralisation, de façon à gérer efficacement le flux d'informations entre deux entités quelconques du réseau (Rudnianski, 1986).

Pour une topologie donnée de réseau, il est souvent commode de considérer l'architecture de ce réseau comme un modèle conceptuel structuré en couches. Une telle architecture intègre un certain nombre de protocoles de communications permettant l'interconnexion et la coopération d'un ensemble de systèmes. Le modèle *OSI (Open Systems Interconnection)*, mis au point par l'*ISO (International Standard Organization)* et comportant sept niveaux hiérarchiques ou couches, constitue un modèle de référence. La Figure 3.1 en donne une illustration.

Dans ce mémoire, nous n'élaborons pas sur le rôle et les multiples fonctions de chacune des couches du modèle. Nous nous contentons simplement d'en distinguer deux catégories (Tanenbaum, 1988):

- les *couches de haut niveau* comprenant les couches *Application*, *Présentation*, *Session* et *Transport*;
- celles de *bas niveau* regroupant les couches *Réseau*, *Liaison* et *Physique*.

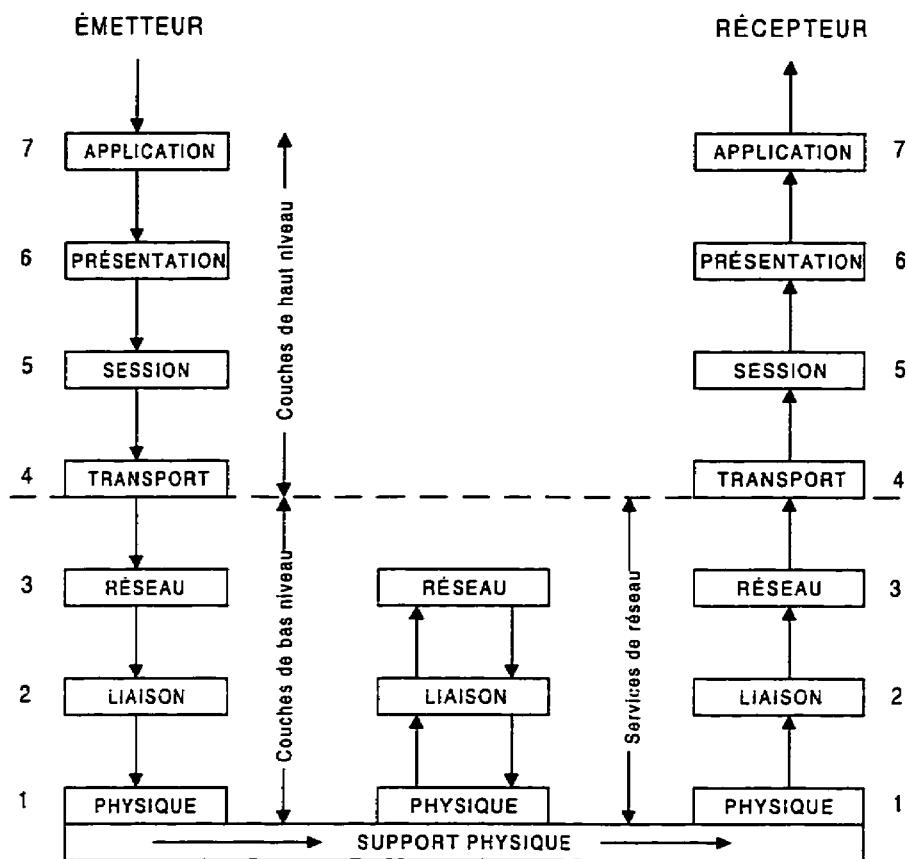


Figure 3.1 Le modèle de référence OSI

Il importe de mentionner que, logiquement, l'acheminement de l'information dans un réseau distribué est une fonction de la couche Réseau du modèle OSI, fonction qui fait appel en principe aux services des couches de plus bas niveau. Cela s'avère très intéressant pour l'informaticien, qui est plutôt préoccupé par l'aspect logique du routage

et qui doit mettre en oeuvre toute une série de protocoles permettant à deux entités quelconques du réseau de se communiquer. Cependant, la conception d'un algorithme performant de routage demeure avant tout un problème complexe d'optimisation, faisant référence au flot de multicommodités. Nous nous proposons d'en expliciter les propriétés fondamentales.

3.2 Caractérisation du flot de multicommodités

L'étude du délai dans un système distribué d'informations conduit à considérer un tel système comme un réseau de files d'attente où, à chaque noeud c , les paquets, une fois traités, restent en attente de la prochaine liaison disponible. Il se forme alors une file d'attente à chaque liaison qui rentre à ce noeud. En outre, le flot d'informations à chaque commutateur c du réseau doit être conservé, commodité par commodité, ce qui permet de formuler pour chaque commodité (i, j) du réseau l'équation de conservation de flot au noeud c (Schwartz, 1987):

$$\sum_{p=1}^n f_{pc}^{ij} - \sum_{q=1}^n f_{cq}^{ij} = \begin{cases} -r_{ij}, & \text{si } c = i \\ +r_{ij}, & \text{si } c = j \\ 0, & \text{sinon} \end{cases} \quad (3.1)$$

où r_{ij} désigne le trafic moyen en *bits/seconde* (*bps*) entre les noeuds i et j du réseau, et f_{pc}^{ij} le flot dû à la commodité (i, j) sur la liaison (p, c) .

Considérons le sous-réseau de transport de la Figure 3.2 et supposons que la commodité représentée par γ_3 doit se rendre du noeud 1 au noeud 3. Alors, du noeud 1, les paquets attendent que les liaisons $(1, 2)$, $(1, 4)$ et $(1, 5)$ deviennent disponibles pour

transiter par chacune d'entre elles et produire respectivement les flots f_{12}^{13} , f_{14}^{13} , f_{15}^{13} .

D'après (3.1), l'équation de conservation de flot au noeud 1 pour la commodité (1, 3)

s'écrit alors:

$$r_{13} = f_{12}^{13} + f_{14}^{13} + f_{15}^{13} \quad (3.2)$$

où r_{13} représente le trafic en bps entre les noeuds 1 et 3. La Figure 3.3 en donne une illustration.

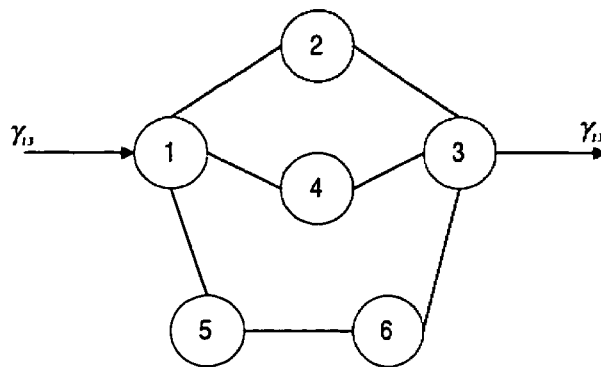


Figure 3.2 Sous-réseau de transport (6 noeuds, 7 liaisons)

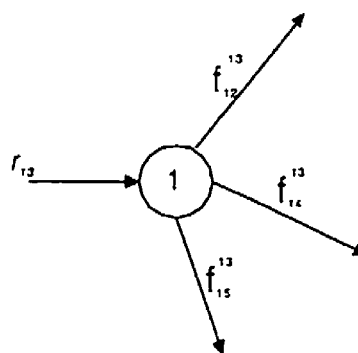


Figure 3.3 Conservation de flot au noeud 1

D'autre part, (3.1) constitue une équation de contrainte qui doit être vérifiée à chaque noeud et pour chacune des n_{sd} commodités du réseau. Pour un réseau de n noeuds, il existe alors $n * n_{sd}$ équations, constituant les contraintes du problème. Et, nous basant sur les résultats de la relation (2.9), nous aboutissons à $n^2(n-1)$ équations de contrainte. Dès lors, l'objectif d'une stratégie de routage consiste à minimiser le délai moyen T_{moy} des paquets du réseau, en déterminant les valeurs de flot f_l^k satisfaisant toutes les contraintes et telles que $f_l^k > 0$, et $f_l < C_l$. Il convient alors de préciser les propriétés mathématiques de la fonction T_{moy} à minimiser, propriétés pouvant nous guider dans la recherche d'une solution optimale.

Considérons un problème d'optimisation (P) dont la fonction objective est donnée par la relation suivante:

$$z = \sum_{i=1}^n a_i x_i \quad (3.3)$$

où les x_i représentent les n variables à optimiser et les a_i désignent leurs coefficients respectifs. Alors, si toutes les contraintes de (P) sont linéaires, ce dernier sera considéré comme un problème de programmation linéaire. Dans ce cas, il existe des techniques d'optimisation mathématique permettant de résoudre un tel problème dans un temps raisonnable. Par contre, à cause de la non linéarité de la fonction objective T_{moy} , le problème de routage appartient à une classe de problèmes plus difficiles, faisant appel à des techniques plus puissantes de résolution.

Déterminons le minimum d'une fonction de deux variables définie par $f(x_1, x_2)$ et soumise à une série de contraintes. Si l'ensemble χ de toutes les solutions réalisables appartiennent à un polygone, on dit que cet ensemble forme un espace convexe. Autrement dit, pour deux points quelconques a et b de χ , tous les points de la droite (a, b) se retrouvent à l'intérieur de la région formée par χ . La Figure 3.4 présente un espace convexe. Il est alors intéressant de se rendre compte que l'on peut passer d'une solution réalisable à une autre à l'intérieur d'un espace convexe, sans quitter pour autant la région réalisable. Ce constat favorise l'utilisation de méthodes simples, itératives pour la recherche d'une solution optimale.

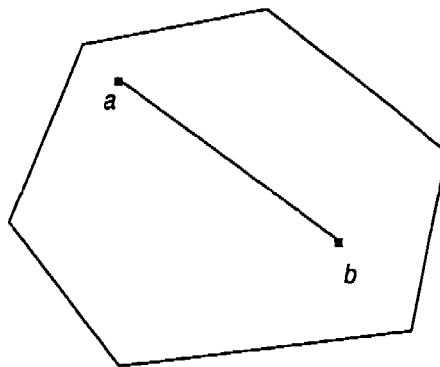


Figure 3.4 Espace convexe

Dans le même ordre d'idée, considérons deux points $A = (a, f(a))$ et $B = (b, f(b))$ d'une courbe décrite par la fonction $f(x)$. Désignons par (A, B) la droite reliant ces deux points. La fonction $f(x)$ est dit *convexe* si et seulement si, pour toute valeur c comprise

entre a et b , le point $C = (c, f(c))$ se situe au-dessous de la droite (A, B) . Algébriquement, une fonction convexe f vérifie la propriété suivante:

$$(1 - \lambda)f(a) + \lambda f(b) \geq f((1 - \lambda)a + \lambda b), \quad 0 \leq \lambda \leq 1 \quad (3.4)$$

La Figure 3.5 illustre une telle fonction.

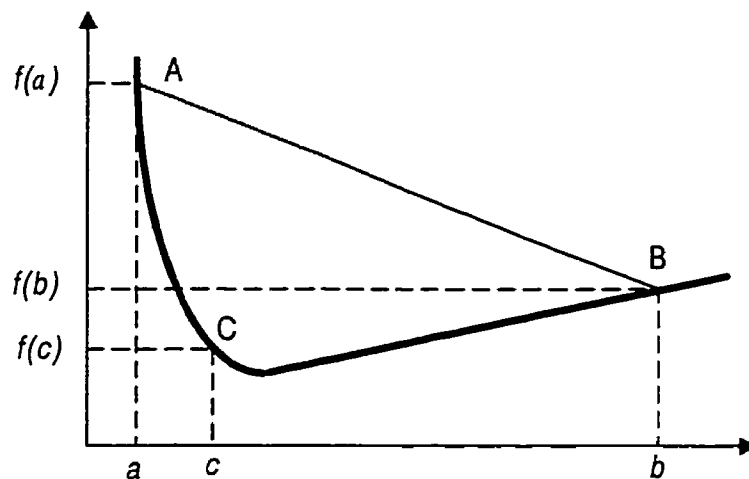


Figure 3.5 Fonction convexe

Remarquons qu'une fonction convexe comporte un minimum global qui peut être déterminé par une recherche locale. Cette dernière consiste à déterminer une solution réalisable et à se déplacer dans la direction de décroissance (direction de descente) de la fonction objective, jusqu'à atteindre la solution optimale, ce qui permet d'éviter l'exploration de tout l'espace de recherche. Il est intéressant de mentionner que la fonction T_{moy} donnée par la relation (2.7) est convexe (Bertsekas & Gallager, 1987), ce qui sous-entend qu'un minimum local représente aussi un minimum global. Plusieurs algorithmes utilisent ce résultat pour traiter le problème de routage de façon optimale.

3.3 Caractérisation des procédures de routage

Les procédures de routage peuvent être classées selon leurs objectifs de performance (Legault, 1994; Schwartz, 1987). Nous en distinguons trois grandes catégories: les procédures de *routage statique*, dont le choix du chemin entre deux noeuds i et j du réseau peut être calculé d'avance; celles de *routage adaptatif*, qui tiennent compte de l'état courant du réseau pour prendre les bonnes décisions; et celles qui se basent sur les propriétés mathématiques de la fonction objective T_{moy} pour résoudre le problème de façon optimale. Nous nous proposons de les passer en revue.

3.3.1 Routage statique

Le routage statique détermine d'avance tous les chemins entre chaque source et chaque destination selon des métriques statiques, telles que la longueur des liaisons ou le nombre de liaisons intermédiaires (*hops*) entre toutes les paires source/destination. Cette procédure de routage, dite de plus courte distance, est largement utilisée: elle s'avère simple à implémenter et minimise le nombre de liaisons ou la distance du chemin à parcourir par l'information. Par contre, elle ne fait aucune considération du temps d'attente et de traitement des paquets à l'entrée de chaque liaison, ce qui peut conduire à la congestion de certains noeuds ou de certaines liaisons du réseau.

3.3.2 Routage adaptatif

Les stratégies de routage adaptatif exigent, pendant tout le fonctionnement du réseau, une mise en mémoire du niveau de congestion en chaque noeud, de façon à tenir compte de tout changement de trafic. L'adaptation du niveau de trafic peut se réaliser de façon centralisée, isolée ou distribuée.

Les algorithmes de routage avec adaptation centralisée font appel à un noeud spécial, appelé *centre de contrôle du routage* (RCC pour *Routing Control Center*) pour gérer l'acheminement des paquets de chacune des commodités. L'échange d'informations entre le RCC et les autres noeuds se fait de façon ordonnée et comporte plusieurs étapes. Périodiquement, chaque commutateur transmet au RCC des informations sur son état (longueur de chaque file d'attente, trafic traité depuis le dernier rapport, occupation de mémoire). Ce dernier se base sur ces informations pour évaluer les chemins et renvoie les résultats de son évaluation à chaque noeud. Ce processus paraît idéal dans la mesure où le RCC possède tous les renseignements nécessaires à une bonne prise de décision. Par contre, toute l'information étant concentrée en un noeud, le réseau devient très vulnérable à toute panne reliée au RCC.

Quant au routage avec adaptation isolée, il consiste à rediriger le flot d'informations provenant d'un noeud vers la liaison ayant la file d'attente la plus courte. Le routage par inondation en constitue un cas particulier. Par cette technique, chaque noeud connaît le noeud destination du message et, à la réception de chaque paquet, il en envoie une copie à chacun des noeuds voisins, sauf au noeud source. Le réseau est alors inondé de paquets qui, éventuellement, arriveront à destination. Malgré sa simplicité, le

roulage par inondation est largement utilisé, spécifiquement dans les réseaux dont la topologie change rapidement (réseaux de radios mobiles) et dans les réseaux militaires. Peu économique, mais efficace et fiable, cette stratégie permet d'atteindre un correspondant n'importe où, pourvu qu'il existe un chemin permettant de l'atteindre. Cependant, l'enregistrement de copies multiples de chaque message à chaque noeud peut éventuellement conduire à la saturation de certains noeuds et de certaines liaisons. Une stratégie qui se rapproche du roulage par inondation, c'est l'*acheminement aléatoire* par lequel les paquets sont acheminés suivant un critère d'allure aléatoire, indépendant de leur destination qu'ils finiront par atteindre.

Les procédures de roulage avec adaptation distribuée constituent un compromis entre les algorithmes de roulage centralisé et ceux de roulage isolé. Dans ce cas, tous les commutateurs jouent le même rôle et les informations concernant l'état du réseau sont directement échangées entre noeuds adjacents, ce qui permet à chacun de ces noeuds de mettre à jour sa table de roulage en fonction de l'état du réseau. Cette stratégie s'avère plus robuste que les précédentes.

3.3.3 Routage optimal

La troisième classe de procédures comporte les algorithmes utilisant les techniques d'optimisation de fonctions non linéaires pour déterminer de façon optimale les chemins de toutes les commodités de la matrice de trafic. Ce sont des méthodes itératives qui réduisent le délai des paquets en redirigeant, à chaque itération, une partie du flot des chemins les plus chargés vers les moins congestionnés, ce qui conduit

généralement à l'acheminement d'une même information sur plusieurs chemins. Nous présentons deux algorithmes classiques de routage optimal: la méthode de *déviaton de flot* (Fratta *et al.*, 1974) et l'algorithme de Bertsekas-Gallager (Bertsekas & Gallager, 1987).

La méthode de déviaton de flot

La méthode de déviaton de flot (*FD* pour *Flow Deviation*) constitue un cas particulier d'une méthode générale (*Frank-Wolfe method*) d'optimisation de fonctions non linéaires, avec contraintes convexes (Bertsekas & Gallager, 1987). En supposant indépendante l'arrivée des paquets et la longueur des messages exponentiellement distribuée, cette procédure détermine la direction de descente la plus rapide de la fonction objective pour aboutir à la solution optimale. Autrement dit, à chaque itération, l'algorithme détermine un ensemble de chemins offrant la meilleure amélioration marginale et y fait dévier une quantité appropriée du trafic provenant des chemins préexistants. Évidemment, cela réduit monotonement le délai T_{moy} à chaque itération, jusqu'à aboutir à un minimum global. Il convient de noter qu'à un minimum f^* , $T(f^*)$ est nécessairement stationnaire: pour une légère perturbation ∂f telle que $(f^* + \partial f)$ est un vecteur flot de multicommodités, le délai augmente (Kershenbaum, 1993), ce qui se traduit par: $T(f^* + \partial f) \geq T(f^*)$. Ces principes conduisent à l'énoncé de l'algorithme FD dont les principales étapes s'énoncent comme suit:

Étape 0: Détermination du flot initial

Soit $f^{(0)}$ un vecteur flot initial réalisable. Posons $p = 0$;

Étape 1: Calcul des dérivées partielles du délai par rapport au flot pour chaque liaison

Affecter à chaque liaison l du réseau la longueur suivante:

$$L_l^{(p)} = \left[\frac{\partial T}{\partial f_l} \right]_{f=f^{(p)}}, \quad \forall l = 1, 2, \dots, m \quad (3.5)$$

Étape 2: Calcul du vecteur flot $\Phi^{(p)}$

Déterminer le flot de chaque liaison correspondant au plus court chemin entre chaque paire source/destination, avec comme métrique $L_l^{(p)}$ de la relation (3.5) de l'étape 1. Soit $\Phi^{(p)}$ ce vecteur. Alors, $\Phi^{(p)}$ fait partie d'un ensemble convexe.

Étape 3: Détermination de λ_p

Trouver la valeur de λ (avec $0 \leq \lambda \leq 1$) qui minimise le délai correspondant à la combinaison convexe de $\Phi^{(p)}$ et de $f^{(p)}$. Soit λ_p cette valeur. Alors, l'expression

$$T \left[(1 - \lambda_p) \Phi^{(p)} + \lambda_p f^{(p)} \right], \quad 0 \leq \lambda_p \leq 1 \quad \text{est minimale.}$$

Étape 4: Mise à jour des flots de liaisons

Poser:

$$\mathbf{f}^{(p+1)} = \left[(1 - \lambda_p) \Phi^{(p)} + \lambda_p \mathbf{f}^{(p)} \right], \quad 0 \leq \lambda_p \leq 1$$

Étape 5: Condition d'arrêt

Soit ε un seuil très faible, proche de zéro. Si $|T(\mathbf{f}^{(p+1)}) - T(\mathbf{f}^{(p)})| \leq \varepsilon$, arrêter:

$\mathbf{f}^{(p)}$ est optimal au seuil ε .

Sinon, incrémenter p et aller à l'étape 1.

Même si la méthode FD converge vers le délai optimal, il n'en demeure pas moins que cette convergence est lente, surtout dans le cas des grands réseaux, ce qui a amené les chercheurs Bertsekas et Gallager (1987) à adopter la méthode de projection au problème de routage, de façon à accélérer la convergence vers l'optimalité.

L'algorithme de Bertsekas-Gallager

Proche de la solution optimale, l'algorithme de Bertsekas-Gallager (BG) détermine une approximation de la dérivée seconde du délai par rapport au flot et utilise cette approximation comme facteur de correction sur la quantité de flot à déplacer. Synthétisons les différentes étapes de cet algorithme.

Étape 1: Calcul du délai incrémental initial

Ce délai correspond, comme pour l'algorithme de déviation de flot, à la dérivée partielle du délai par rapport au flot. Alors, pour une liaison l , il s'obtient comme suit:

$$L_l = \frac{\partial T_l}{\partial f_l} = \frac{C_l}{(C_l - f_l)^2} \quad (3.6)$$

Le délai incrémental initial correspond à celui où le flot de chaque liaison f_l est nul. Dans ce cas, il est donné par:

$$L_l = 1/C_l$$

Étape 2: Calcul de la longueur de chaque chemin

Acheminer les paquets des commodités selon les chemins les plus courts, avec L_l comme métrique. Déterminer alors la longueur L_p de chaque chemin.

Étape 3: Calcul de la longueur améliorée de chaque chemin

Déterminer les nouvelles longueurs de liaisons à partir de la mise à jour des valeurs de flot. Calculer alors la longueur améliorée L_p^* de chaque chemin.

Étape 4: Détermination de la longueur correspondant à la dérivée seconde du délai

Déterminer la dérivée seconde S_l de T_{moy} par rapport à f_l , ce qui donne:

$$S_i = \frac{2 C_i}{(C_i - f_i)^3} \quad (3.7)$$

La longueur H_p d'un chemin correspond à la somme des dérivées secondes de toutes les liaisons faisant partie d'un tel chemin.

Étape 5: Détermination de la portion de flot à dévier

Pour chaque commodité, la quantité de flot à déplacer du chemin préexistant au chemin courant est donnée par l'expression:

$$\delta = \frac{\alpha (L_p - \hat{L}_p)}{H_p} \quad (3.8)$$

où α représente un pas. Pour un réseau de n noeuds et m liaisons, il est recommandé de choisir (Kershenbaum, 1993):

$$\frac{1}{n} \leq \alpha \leq \frac{2m}{n^2} \quad (3.9)$$

Étape 6: Condition d'arrêt

Si δ devient très faible, calculer le délai et arrêter. Sinon, mettre à jour le flot de chaque liaison et aller à l'étape 3.

La vitesse de convergence de cet algorithme dépend en grande partie de la valeur de α . Évidemment, une grande valeur de α augmente la quantité de flot à dévier à chaque itération, mais ne garantit pourtant pas une bonne vitesse de convergence. Il s'avère donc

important de déterminer la valeur de α qui optimise la vitesse de convergence. Et, même si l'utilisation de la dérivée seconde de la fonction objective améliore généralement la vitesse de convergence vers le délai optimal, le réajustement de la longueur associée à chaque liaison, pour chacune des commodités, peut conduire à une complexité élevée. Il est alors évident que les techniques à base de déviation de flot ne sont efficaces que pour les petits réseaux, où le nombre de commodités n_{sd} est relativement faible.

Ainsi, les heuristiques de routage statique demeurent simples à implémenter et rapides d'exécution, mais elles ne tiennent nullement compte du niveau de trafic dans le réseau, ce qui ne garantit pas du tout un bon délai. Dans le même ordre d'idée, les procédures de routage adaptatif tiennent compte du niveau de trafic par la mise en mémoire en chaque noeud de l'état courant du réseau, ce qui peut conduire à la saturation éventuelle de la mémoire de ces noeuds. Quant aux algorithmes de routage optimal, ils ne sont pratiquement pas utilisés, à cause de leur temps d'exécution qui est relativement long. Tout ceci nous amène à nous demander s'il est possible de concevoir un algorithme de routage qui, tout en étant simple à implémenter et rapide d'exécution, tient compte du niveau de congestion de chaque liaison pour opérer de façon efficace et produire une solution proche de l'optimale. C'est l'objet de la prochaine section.

3.4 Nouvelle approche de routage basée sur les attributs de liaisons

Nous nous proposons de présenter un nouvel algorithme de routage, axé sur la détermination du flot f_l , de la capacité C_l et du délai T_l de chacune des liaisons du réseau.

D'abord, nous établirons les fondements d'un tel algorithme, puis nous en donnerons les grandes lignes avant d'en évaluer la complexité. Pour finir, nous présenterons les résultats obtenus, résultats que nous comparerons à ceux fournis par les algorithmes concurrents.

3.4.1 Fondements de l'approche

Nous nous basons sur deux principes d'ordre intuitif pour éviter la congestion et minimiser le plus possible le délai des paquets. Ce sont:

- Imposer un seuil au délai de chaque liaison;
- Acheminer toujours les paquets selon le chemin le plus rapide.

Le premier principe limite le délai T_l de chacune des liaisons à une valeur maximale. Implicitement, le délai moyen T_{moy} se doit aussi d'être limité, selon la relation (2.4) de proportionnalité entre T_l et T_{moy} . Quant au deuxième principe, il permet de minimiser éventuellement le délai moyen de paquet. L'application de ces principes au problème de routage implique l'utilisation efficace des données du problème.

Par chemin le plus rapide, nous entendons le chemin comprenant les liaisons dont le délai total d'attente et de transmission entre une source et une destination est le plus faible. Aussi s'avère-t-il important de déterminer le délai de chaque liaison. C'est là l'originalité de la première étape de notre approche, qui utilise des éléments bien spécifiques de la matrice de trafic, ainsi que la topologie et la table des options de capacités pour déterminer tous les paramètres dont l'algorithme aura initialement besoin pour être exécuté.

Considérant que deux noeuds sont adjacents s'ils sont reliés par une liaison unique, nous déduisons que le nombre de paires de noeuds adjacents d'un réseau est équivalent au nombre total m de liaisons de ce réseau. Ainsi, la première étape de notre stratégie de routage permet de déterminer les paramètres conduisant au calcul des attributs de chaque liaison, en acheminant le flot de toute commodité (i, j) telle que le noeud i est adjacent au noeud j sur la liaison (i, j) . Par cette étape, le trafic des m commodités dont la source est adjacente à la destination est alors acheminé. Un tel procédé permet, d'une part, de réduire la charge du réseau et, d'autre part, de déterminer les valeurs initiales des attributs de chaque liaison, tels que le flot f_i , la capacité C_i et le délai T_i . Ces valeurs sont mises à jour au fur et à mesure du déroulement de l'algorithme.

Un exemple simple peut illustrer notre propos. Supposons que l'on veuille transmettre un paquet de 1000 bits d'un noeud i à un noeud j . S'il existe un chemin direct entre ces deux noeuds, nous pouvons transmettre tout le flot le long de cette liaison et la contribution totale par rapport au reste du réseau demeure 1000 bits. Par contre, si nous devons passer par des noeuds intermédiaires p et q , ce paquet suivra le chemin formé par les liaisons (i, p) , (p, q) et (q, j) . Ainsi, le paquet contribue à un débit effectif de 3000 bits au réseau, ce qui aura pour conséquence d'augmenter la charge d'un tel réseau et éventuellement de le congestionner.

D'autre part, l'acheminement direct des paquets conduit à un flot initial sur chaque liaison. Ce flot, exprimé en paquets/seconde (paq/s), peut être converti en bps en multipliant le nombre de (paq/s) par $(1/\mu)$ qui représente la taille moyenne des paquets. Cette affectation de flots conduit aussi à une affectation de capacités en choisissant, de la

table des options de capacités, une capacité C_l telle que $f_l < C_l$. Connaissant f_l , C_l , ainsi que la taille moyenne des paquets ($1/\mu$), nous pouvons appliquer la relation (2.5) pour déterminer le délai d'attente et de transmission d'un paquet sur chaque liaison.

Toutefois, il convient d'être prudent dans le choix de la capacité de la liaison l . En effet, si C_l est choisie trop proche de f_l , le délai T_l peut devenir assez grand. Par exemple, pour un f_l de 9.0 kbps, il est tout à fait normal, en utilisant le Tableau 3.2, de choisir une capacité de 9.6 kbps. Dans ce cas, le délai d'un paquet de 1000 bits sur la liaison l , calculé à partir de la relation (2.5), sera de 1.667 s, ce qui est relativement élevé et qui peut augmenter considérablement T_{moy} tel que celui-ci est exprimé par la relation (2.4). Pour éviter ces cas pathologiques et, du même coup, augmenter la puissance de notre approche, nous y ajoutons un seuil de délai maximal $T_{l_{\text{max}}}$ sur chaque liaison l . Autrement dit, la capacité C_l doit être choisie de façon à respecter à la fois les contraintes:

$$\begin{cases} f_l < C_l \\ T_l \leq T_{l_{\text{max}}} \end{cases} \quad (3.10)$$

Reprenons l'exemple précédent avec le même flot de 9.0 kbps. Pour un $T_{l_{\text{max}}}$ de 100 ms, notre algorithme détermine une capacité de 19.2 kbps, conduisant à un délai T_l de 98.0 ms. Ce qui contribue à maintenir T_{moy} dans des limites plus raisonnables.

Le choix de $T_{l_{\text{max}}}$ dépend de la performance et de la qualité de service que l'on veut obtenir du réseau. En outre, ce choix doit implicitement tenir compte du niveau de trafic de chaque liaison du réseau, car une liaison doit être d'autant plus rapide que le niveau de trafic y est élevé. De ce fait, nous choisissons $T_{l_{\text{max}}}$ inversement proportionnel

au trafic moyen du réseau. En outre, la performance et la qualité de service sont prises en considération par une constante de proportionnalité K . Ainsi, combinant tous ces paramètres, nous aboutissons à la relation suivante définissant le délai maximal sur chaque liaison l :

$$T_{l_{\max}} = \frac{K}{\bar{\gamma}} \quad (3.11)$$

où K représente une constante de qualité de service et:

$$\bar{\gamma} = \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n \gamma_{ij} \quad (3.12)$$

représente le trafic moyen du réseau. Dans le cas d'un trafic uniforme, nous obtenons:

$$\bar{\gamma} = \gamma_{ij} \quad (3.13)$$

Alors, pour un trafic uniforme de 10 paq/s et pour $K = 1$, nous limitons le délai de chacune des liaisons à 100 ms/paq. Si nous fonctionnons à haut débit et que le trafic, toujours uniforme, passe à 100 paq/s, nous nous rendons compte que le délai maximal admis sur chaque liaison devient 10 ms pour chaque paquet. Autrement dit, à haut débit, le délai d'attente et de transmission sur chaque liaison doit être très faible. Enfin, la performance est d'autant meilleure que $T_{l_{\max}}$ est faible.

Rappelons que nous n'avons acheminé que le trafic de m commodités jusqu'à présent. En supposant une matrice de trafic symétrique ($\gamma_{ij} = \gamma_{ji}$), nous pouvons seulement manipuler la triangulaire supérieure de Γ . Dans ce cas, il reste $\{n(n-1)/2 - m\}$ commodités à considérer. C'est le rôle d'une seconde étape.

En utilisant les attributs de liaisons calculés à la première étape, nous pouvons acheminer le trafic de la $(m+1)^{\text{e}}$ commodité de Γ (premier élément dont la source est non adjacent à la destination) selon le chemin le plus rapide. Dans ce cas, l'ajout de ce trafic sur les liaisons faisant partie d'un tel chemin augmente le flot, éventuellement la capacité (en tenant compte du seuil de délai maximal par liaison) et change le délai de chacune de ces liaisons. En conséquence, une mise à jour des attributs de toute liaison faisant partie du chemin le plus rapide s'avère nécessaire. Cette phase est répétée jusqu'à l'acheminement du trafic de toutes les commodités de Γ . On peut alors vérifier que le vecteur flot f satisfait aux exigences de la matrice Γ . Il ne reste plus qu'à déterminer T_{moy} par l'application de la relation (2.7). Ce calcul est fait dans une troisième étape.

On se rend compte qu'implicitement notre algorithme respecte toutes les contraintes et utilise toutes les données disponibles, de façon à bien choisir les chemins des commodités et minimiser ainsi le délai moyen de chaque paquet. En outre, l'acheminement des paquets de chaque commodité le long du chemin le plus rapide contribue à éviter le plus possible les chemins congestionnés.

3.4.2 Énoncé de l'algorithme

L'énoncé de notre algorithme suppose que chaque liaison (i, j) du réseau est bidirectionnelle et symétrique, de sorte que nous puissions manipuler seulement la triangulaire supérieure de la matrice de trafic Γ . Alors, pour un réseau de n noeuds et de m liaisons, notre heuristique de routage comporte trois étapes que nous allons expliciter.

Soit T_{ij}^p le délai provisoire de la liaison (i, j) , f_{ij}^p , et λ_{ij}^p les flots provisoires et C_{ij}^p la capacité provisoire sur cette liaison (i, j) .

Étape 1: Détermination du délai provisoire de chaque liaison

- a) Parcourir la matrice de trafic Γ ;
- b) Tant que le noeud i est adjacent au noeud j , choisir l'élément γ_{ij} de Γ et effectuer les opérations suivantes:

$$\lambda_{ij}^p \leftarrow \gamma_{ij}$$

$$f_{ij}^p \leftarrow \frac{\lambda_{ij}^p}{\mu}$$

$$C_{ij}^p \leftarrow f(T_{ij_{max}})$$

$$T_{ij}^p \leftarrow \frac{1}{\mu [C_{ij}^p - f_{ij}^p]}$$

Étiqueter l'élément γ_{ij} courant et considérer γ_{ij} suivant.

Étape 2: Routage du trafic des commodités restantes selon le délai provisoire de liaison

Tant que $[n(n-1)/2 - m] > 0$ (il reste encore des commodités à considérer), effectuer les opérations suivantes:

- a) Choisir l'élément γ_{sr} de Γ non étiqueté: (s, t) forme un chemin

$$\Pi_{sr}$$

b) Déterminer le plus court chemin entre s et t , avec comme

métrique T_{ij}^p pour tout $(i, j) \in \Pi_\pi$

c) Mise à jour des attributs de chaque liaison $(i, j) \in \Pi_\pi$

$$\lambda_{ij}^p \leftarrow \lambda_{ij}^p + \gamma_\pi$$

$$f'_{ij}^p \leftarrow \frac{\lambda_{ij}^p}{\mu}$$

$$C_{ij}^p \leftarrow f(T_{ij_{max}})$$

$$T_{ij}^p \leftarrow \frac{1}{\mu [C_{ij}^p - f'_{ij}^p]}$$

$$T_{ij}^p \leftarrow T_{ij}^p$$

Étiqueter l'élément γ_π de Γ .

d) Continuer le parcours de Γ

$$\Rightarrow m \leftarrow m + 1$$

Étape 3: Calcul des attributs de liaisons et du délai moyen de paquet

Pour chaque liaison $(i, j) \in A$, l'ensemble des liaisons du réseau, effectuer les opérations suivantes:

$$f_{ij} \leftarrow f_{ij}^p$$

$$C_{ij} \leftarrow C_{ij}^p$$

Déterminer alors le délai moyen de paquet par la relation suivante:

$$T_{moy} = \frac{1}{\gamma} \sum_{(i,j) \in A} \frac{f_{ij}}{C_{ij} - f_{ij}}, \text{ avec } \gamma = \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n \gamma_{ij}$$

3.4.3 Complexité algorithmique

La complexité algorithmique de notre approche peut être déterminée par le calcul de la complexité de chacune des trois étapes. Évidemment, cette complexité est fonction de la topologie: elle dépend du nombre n de noeuds et du nombre m de liaisons du réseau. Il s'avère donc important de considérer d'abord le meilleur cas, ensuite le pire cas pour notre étude de complexité.

3.4.3.1 Complexité au meilleur cas

Le meilleur cas est représentatif d'un réseau complet, comportant m_{max} liaisons, où m_{max} est donné par:

$$m_{max} = \frac{n(n-1)}{2} \quad (3.14)$$

Dans ce cas, l'algorithme n'exécute pas l'étape 2: la complexité est uniquement déterminée par les opérations des étapes 1 et 3. En tenant compte de la symétrie de Γ , la première étape parcourt $n(n-1)/2$ éléments et effectue 5 opérations de durée indépendante des paramètres du réseau sur chacun de ces éléments. Alors, le nombre total d'opérations est déterminé par la relation:

$$f_1(n) = \frac{5n(n-1)}{2} \quad (3.15)$$

Quant à la troisième étape, elle effectue deux opérations d'affectation sur les m_{max} liaisons, puis le calcul du délai moyen T_{moy} qui est essentiellement une somme sur les m_{max} liaisons. Dans ce cas, le nombre total d'opérations effectuées à cette étape est donné par:

$$f_3(n) = 2 m_{max} + m_{max}$$

Ce qui peut s'exprimer en fonction de n par:

$$f_3(n) = \frac{3n(n-1)}{2} \quad (3.16)$$

Ainsi, dans le meilleur cas, le nombre total d'opérations effectuées par notre algorithme est donné par:

$$f(n) = f_1(n) + f_3(n)$$

Ce qui donne:

$$f(n) = 4n(n-1) \quad (3.17)$$

Nous déduisons que, dans le meilleur cas, notre algorithme de routage est d'ordre $O(n^2)$.

3.4.3.2 Complexité au pire cas

Le pire cas correspond à celui où le terme $[n(n-1)/2-m]$ est maximal. Autrement dit, le nombre m de liaisons doit être minimal et donné par:

$$m_{min} = n - 1$$

Ce qui est caractéristique d'une configuration topologique en arbre. Dans ce cas, l'algorithme exécute les trois étapes.

L'étape 1 parcourt la matrice de trafic Γ et effectue 5 opérations sur chacun des m éléments. Alors, le nombre total d'opérations exigé par cette étape est donné par:

$$f_1(n) = 5 (n - 1) \quad (3.18)$$

L'étape 2 manipule $[n(n-1)/2 - m_{min}]$ éléments de Γ . En fonction de n , ce nombre d'éléments est égal à $(n-1)(n-2)/2$. Sur chacun de ces éléments, l'algorithme vérifie si l'élément en question est étiqueté (opération a), détermine le plus court chemin entre p et q (opération b) et exécute 5 opérations d'affectation et une d'étiquetage, avant de passer à l'élément suivant de Γ . En tenant compte que, pour une structure en arbre, l'algorithme de plus court chemin est d'ordre $O(n)$ (Evans & Minieka, 1992), le nombre total d'opérations exécuté par l'étape 2 est donné par:

$$f_2(n) = (1 + n + 5 + 1) * (n - 1)(n - 2)/2$$

D'où:

$$f_2(n) = \frac{n^3 + 4n^2 - 19n + 14}{2} \quad (3.19)$$

Quant à la troisième étape, elle effectue, comme dans le meilleur cas, 2 opérations d'affectation sur les m_{min} liaisons et le calcul du délai moyen T_{moy} . Par conséquent, le nombre total d'opérations effectuées par cette étape est donné par:

$$f_3(n) = 2 m_{min} + m_{min}$$

Ce qui peut s'exprimer en fonction de n par:

$$f_3(n) = 3n - 3 \quad (3.20)$$

Ainsi, dans le pire cas, le nombre total d'opérations effectuées par notre algorithme est déterminé par la relation:

$$f(n) = f_1(n) + f_2(n) + f_3(n)$$

Ce qui donne:

$$f(n) = \frac{n^3 + 4n^2 - 3n - 2}{2} \quad (3.21)$$

Donc, la complexité dans le pire cas est d'ordre $O(n^3)$. En conclusion, la complexité de notre algorithme de routage est polynomiale et comprise entre $O(n^2)$ et $O(n^3)$, ce qui se compare bien aux algorithmes de routage existants.

3.4.4 Détails d'implémentation

Même si l'interface de notre programme paraît conviviale, nous passons plutôt en revue les divers modules de traitement. Au départ, l'utilisateur doit rentrer la configuration topologique ou préciser les fichiers où sont enregistrées les coordonnées des nœuds et les liaisons du réseau. Par ailleurs, la table des options de capacités, ainsi que la matrice de trafic sont incorporées dans le programme. Utilisant les données disponibles, ce dernier crée des structures de liaisons où sont principalement enregistrés le flot, la capacité et le délai de chaque liaison.

Initialement, le flot de chaque liaison et le délai moyen de chaque paquet sont nuls. L'étape 1 recherche alors les paires de nœuds directement connectées et, pour

chacune d'entre elles, achemine les paquets de telles commodités sur la liaison adjacente. Elle en calcule ainsi le flot et, à partir du seuil de délai, détermine la capacité minimale satisfaisant à cette contrainte, ce qui permet de déterminer le délai de chaque liaison, donné par la relation (2.5).

L'exécution de la deuxième étape exige un test portant sur le nombre maximal de liaisons (calculé à partir de la relation 3.14) et le nombre effectif de liaisons du réseau (précisé par l'utilisateur ou enregistré dans un fichier). Si ces paramètres s'avèrent égaux, l'algorithme passe directement à l'étape 3 et calcule le délai moyen de chaque paquet par l'application de la relation (2.7). Dans le cas contraire, les attributs de liaisons déterminés à l'étape 1 sont considérés comme provisoires et le routage des paquets se fait selon l'algorithme de plus court chemin avec comme métrique T_l , le délai d'attente et de transmission d'un paquet sur la liaison l . Toutefois, l'application de cet algorithme demeure dynamique, car la valeur de T_l doit être mise à jour au fur et à mesure que les paquets transitent par la liaison l . C'est d'ailleurs pourquoi la complexité de notre algorithme peut atteindre $O(n^3)$ dans le pire cas. Les détails des principales fonctions de traitement sont présentés en Annexe I, section AI.1.

3.5 Caractéristiques des réseaux à haut débit

L'avènement de la fibre optique a permis d'accroître considérablement la largeur de bande des liaisons de communications et conduit ainsi à l'émergence des réseaux à

haut débit. Ces derniers comportent des caractéristiques spécifiques que nous nous proposons ici de passer en revue.

3.5.1 Exigences du haut débit

L'augmentation de la puissance de traitement des ordinateurs et la croissance rapide du volume d'informations traitées et enregistrées ont beaucoup motivé l'utilisation de réseaux à haut débit. En effet, les équipements et commutateurs issus des progrès de l'intégration à très grande échelle (VLSI) permettent actuellement de faire de la commutation à très grande vitesse, de l'ordre de terrabits (10^{12} bits) par seconde (Cidon *et al.*, 1995). Dans ce cas, des vitesses de transmission trop faibles conduisent éventuellement à une sous-utilisation de la capacité de traitement de ces noeuds. Pour y remédier, l'introduction de la fibre optique dans l'infrastructure des réseaux constitue une solution idéale, capable de répondre à la fois au débit élevé et au faible délai de transmission. Car, en plus de permettre aux noeuds interconnectés de fonctionner à distance avec des temps de réponse très courts et des débits pouvant dépasser le Gbps (Vetter *et al.*, 1993), la fibre optique offre la même qualité de service qu'un bus d'ordinateur.

Les réseaux à haut débit peuvent supporter 3 types distincts de trafic: *isochrone*, *synchrone* et *asynchrone* (Vetter *et al.*, 1993). Un trafic est de type isochrone quand il requiert un délai de transmission constant et des conditions de transfert en temps réel. Le transfert d'images animées correspond généralement à ce type de trafic. Quant au trafic synchrone, il renvoie essentiellement à des unités de données transmises selon des

intervalles de temps réguliers, comme la transmission de grands fichiers comportant des images fixes. Enfin, le trafic asynchrone provient de tout ce qui arrive au réseau de façon aléatoire ou sporadique, comme les données tapées à partir d'un terminal.

Chaque type de trafic comporte ses propres exigences en termes de délai et de largeur de bande. Par exemple, la voix et les autres applications en temps réel exigent un délai de transmission faible et limité dans le temps; elles tolèrent la perte d'une certaine quantité d'informations, sans dégradation de la qualité de service. Par contre, le trafic asynchrone des données n'admet pas de perte d'informations, mais accepte un délai variable et plus long et, en cas de surcharge du réseau, peut être retransmis plus tard. Quant aux services de type vidéo (dont le trafic est isochrone), ils exigent un débit élevé et un faible délai de transmission, de façon à garder l'intégrité de l'information et offrir ainsi une bonne qualité de service. Pour la suite, nous considérons un trafic de type asynchrone.

3.5.2 Latence et délai de propagation des paquets

Supposons que l'on veuille transmettre un fichier d'un mega-octet sur une liaison de 500 km, dont la capacité est de 1 Gbps. On s'attend évidemment à ce que chaque paquet de 1000 bits d'un tel fichier ne mette que 1 μ s pour traverser cette liaison. Mais, il en est tout autre: malgré la largeur de bande élevée d'une liaison de haut débit, le temps de transfert d'un paquet sur cette liaison peut être relativement long. Cela n'est nullement relié à la technologie, mais plutôt à la vitesse limitée de la lumière sur la liaison. En effet, on peut bien vouloir augmenter la vitesse de transmission de chaque paquet, mais on ne

peut nullement dépasser la vitesse de propagation c de la lumière dans le vide ($3.0 * 10^8$ m/s). Dès lors, il importe de définir la *latence* ou le délai de propagation d'un paquet par le temps que prend ce dernier pour se munir d'énergie nécessaire et suffisante pour traverser une liaison (Kleinrock, 1992). Pour une liaison l quelconque, nous exprimons ce délai par τ_l , où τ_l est supérieur au temps de propagation c de la lumière dans le vide. Toutefois, dans le but de simplifier le calcul du délai de propagation à travers la fibre optique, nous supposons que les paquets voyagent à la vitesse de la lumière. Par conséquent, le délai de propagation de chaque paquet sur une liaison l s'exprime par :

$$\tau_l = \frac{L_l}{c} \quad (3.22)$$

où L_l représente la longueur de la liaison l .

Si nous exprimons cette longueur en km, la latence du paquet est simplement donnée par:

$$\tau_l = \frac{10^{-5}}{3} L_l \quad (3.23)$$

Par exemple, le temps de propagation d'un paquet sur une liaison de 300 km est de 1.0 ms, si l'on suppose que ce paquet voyage à la vitesse de la lumière. Nous nous rendons compte que ce délai est indépendant de la capacité de la liaison.

D'autre part, le délai de transmission d'un paquet sur une liaison l dépend de la taille de ce paquet et de la capacité de la liaison en question. Ce délai s'exprime par:

$$T_{l,r} = \frac{1}{\mu C_l} \quad (3.24)$$

où $1/\mu$ représente la taille moyenne des paquets, exprimée en bits par paquet (bits/paq) et C_l la capacité de la liaison en bps. Ainsi, à haut débit, l'augmentation de la capacité contribue à diminuer le délai de transmission et ne comporte aucun effet sur le délai de propagation, ce qui rend le délai de transmission très faible par rapport au délai de propagation.

3.5.3 Détermination du délai de paquet

À haut débit, le temps moyen d'un paquet sur une liaison correspond à la somme des délais d'attente, de transmission et de propagation de ce paquet sur une telle liaison. On passe alors d'une catégorie de réseaux où l'on est surtout limité par la capacité des liaisons à une autre, principalement limitée par la latence des paquets. Tout compte fait, la communauté scientifique (Kleinrock, 1992; Hwang *et al.*, 1995) suppose l'arrivée poissonnienne et une distribution exponentielle des paquets sur les liaisons de haut débit, ce qui donne lieu au modèle de Jackson où chaque liaison se comporte comme une file d'attente classique de type M/M/1. Dans ce cas, la détermination du temps total T_l du paquet sur une liaison l peut être obtenu par l'ajout du délai de propagation τ_l au temps d'attente et de transmission donné par la relation (2.5) (Kleinrock, 1992). Ainsi, le délai T_l d'attente, de transmission et de propagation d'un paquet sur cette liaison l , est donné par la relation:

$$T_l = \frac{1}{\mu C_l - \lambda_l} + \tau_l \quad (3.25)$$

où:

- τ_l , donné par la relation (3.23), représente le délai de propagation du paquet sur la liaison l ;
- λ_l représente le flot de cette liaison en paq/s;
- C_l sa capacité en bps.

L'application de la relation (2.4) permet alors de déterminer le délai moyen de chaque paquet d'un réseau à haut débit. Un tel délai est donné par:

$$T_{\text{moy}} = \frac{1}{\gamma} \sum_{l=1}^m \lambda_l \left[\frac{1}{\mu C_l - \lambda_l} + \tau_l \right] \quad (3.26)$$

3.5.4 Contrôle de flot et de congestion

Malgré la largeur de bande élevée de ses liaisons, un réseau à haut débit est susceptible de *congestion*. En effet, lorsque ces liaisons ne sont pas efficacement utilisées, les informations provenant des noeuds selon un rythme très élevé peuvent vouloir emprunter certaines liaisons dont le tampon est déjà surchargé, ce qui risque d'augmenter indéfiniment la longueur des files d'attente aux noeuds menant à ces liaisons. D'où l'engorgement éventuel de ces noeuds et la congestion possible du réseau. Il importe alors d'assurer un contrôle efficace du flot émanant de chaque source, de façon à éviter de telles anomalies.

Il existe plusieurs attributs de contrôle de flot dont les plus importants sont: le contrôle de type *préventif* qui utilise des règles permettant tout simplement d'éviter la congestion et le contrôle de type *réactif* qui tente de résoudre le problème de congestion

seulement lorsqu'il survient (Baransel *et al.*, 1995). En général, les méthodes réactives utilisent des boucles de rétroaction qui permettent à chaque paire de noeuds adjacents de s'échanger sur l'état de leur niveau de congestion. Mais, à haut débit, le délai de propagation est trop élevé, ce qui rend l'information de rétroaction inutile et qui nous amène à considérer les méthodes de type préventif pour contrôler efficacement la congestion.

Les procédures de contrôle de flot préventif se basent soit sur le débit, soit sur la qualité de service, pour contrôler globalement la congestion du réseau. Dans le cas de la transmission de données, ces mécanismes permettent d'assigner à chaque noeud une quantité maximale d'informations à transmettre. Ainsi, les paquets ou cellules en surplus peuvent être retardés et retransmis plus tard, de façon à respecter un partage équitable de la capacité d'engorgement des noeuds. Par contre, si le trafic est de type multimédia, le contrôle de flot tient plutôt compte de la qualité de service, en octroyant seulement à ce type de trafic une portion de la bande passante disponible (Baransel *et al.*, 1995), ce qui limite considérablement la quantité d'informations en transit dans le réseau.

3.6 Routage à haut débit

Il existe plusieurs techniques de routage spécifiques aux réseaux à haut débit (Baransel *et al.*, 1995; Khasnabish, 1993); mais rares sont celles qui considèrent le délai de chaque liaison comme étant composé des délais d'attente, de transmission et de propagation, tel qu'exprimé par la relation (3.25). Cette section vise à montrer que notre

algorithme de routage peut facilement s'adapter aux réseaux à haut débit. Mais d'abord, analysons l'impact du débit élevé sur le routage.

3.6.1 Impact du débit élevé

Les réseaux à haut débit permettent non seulement de transférer des informations de manière très rapide, mais surtout d'intégrer sur un même support la voix, les données, les images fixes et animées pour faire du *multimédia*. Il s'avère alors intéressant d'étudier l'impact du débit élevé sur la détermination des chemins de ces informations multimédias.

Le débit élevé se réfère à la fois à la quantité d'informations circulant sur chacune des liaisons du réseau et à la largeur de bande de ces liaisons. Autrement dit, un réseau à haut débit est normalement conçu pour acheminer beaucoup d'informations en des temps extrêmement courts. Toutefois, même si la largeur de bande élevée des liaisons permet le transport de beaucoup d'informations, elle ne garantit pas toujours la bonne réception de ces informations en des temps très courts. À cause du taux élevé d'arrivée des paquets, l'engorgement des liaisons devient crucial, pouvant aboutir à des files d'attente de longueur quasiment infinie. C'est là l'importance d'un bon algorithme de routage qui doit non seulement permettre au réseau d'absorber tout le trafic qui lui est soumis, mais surtout d'éviter des engorgements par une bonne gestion de la largeur de bande des liaisons.

Par ailleurs, à cause de la rapidité intrinsèque de traitement des noeuds de commutation, nous négligeons le temps de traitement aux noeuds (Cidon *et al.*, 1995).

Cela nous amène à nous concentrer surtout sur la complexité de l'algorithme de routage qui doit être rapide d'exécution, et sur le choix minutieux des chemins évitant la congestion. En effet, si cet algorithme s'avère complexe et exige un calcul élaboré aux noeuds, et si les chemins ne sont pas méticuleusement déterminés, le flux de paquets arrivant à très grande vitesse aura tendance à congestionner tout le réseau.

D'autre part, les capacités élevées des liaisons de haut débit contribue à rendre très faible (sinon négligeable) le délai de transmission par rapport au délai de propagation des paquets. En tenant compte de la vitesse constante de propagation des paquets, nous déduisons que ce délai de propagation dépend essentiellement de la longueur de la liaison. Ainsi, selon la longueur de la liaison, le temps de transit du paquet peut être plus ou moins court, ce qui n'est nullement le cas des réseaux à faible débit. De ce fait, un algorithme de routage pour les réseaux à haut débit doit tenir compte de la longueur de chacune des liaisons du réseau.

Ce point est particulièrement intéressant et peut influencer le routage des paquets, surtout dans le cas de notre algorithme qui utilise le chemin le plus rapide pour acheminer les paquets. En effet, pour un réseau à faible débit, les paquets empruntent le chemin constitué des liaisons où le temps d'attente et de transmission est très faible. Maintenant que le délai de propagation devient dominant, nous devons admettre que les paquets empruntent les chemins dont la longueur est la plus faible possible. Ainsi, à haut débit, le temps de traitement et de commutation doit être nul d'une part, et d'autre part, les paquets ont tendance à emprunter les liaisons l avec un C_l élevé et un L_l faible.

3.6.2 Adaptabilité de notre algorithme à haut débit

D'abord, nous devons préciser que notre méthode de routage adaptée aux réseaux à haut débit comporte les mêmes opérations de base que l'algorithme utilisé pour les réseaux à faible débit présenté à la section 3.4. La méthode reste simple et toutes les contraintes d'un problème classique de routage y sont respectées. De plus, la faible complexité algorithmique nous amène à négliger le temps de traitement aux noeuds, d'autant plus qu'à haut débit, ces noeuds sont extrêmement rapides.

D'autre part, le délai maximal imposé sur chaque liaison doit être très faible, de façon à éviter un temps de transit trop long et encombrer ainsi la liaison. Nous avons déjà proposé qu'un tel délai soit choisi inversement proportionnel au trafic moyen (relation (3.11)). À haut débit, le trafic moyen étant très élevé, la contrainte de délai devient encore plus sévère. Par exemple, si le trafic est uniforme de 150 paq/s, avec une constante de qualité de service $K = 1.0$, nous obtenons, par l'application de la relation (3.11), un délai maximal de 6.7 ms sur chaque liaison. Il importe de déterminer les bonnes valeurs de capacités qui permettent de respecter cette contrainte. Ainsi, notre algorithme de routage permet de maintenir le délai sur chaque liaison dans des limites raisonnables.

L'aspect le plus important réside dans la détermination des chemins qui tiennent compte de tous les paramètres caractéristiques du débit élevé et ayant un impact sur le routage. Notre algorithme en tient compte par la détermination du délai total sur chaque liaison, utilisant la relation 3.25. Dès lors, les paquets choisissent les chemins les plus rapides, évitant les liaisons congestionnées.

Pour la mise en oeuvre de notre approche, nous considérons un réseau de 20 noeuds et de 30 liaisons, représenté à la Figure 3.9 et dont les coordonnées sont spécifiées en Annexe I, section AI.4. Nous supposons une matrice de trafic uniforme de 150 paq/s et utilisons les options de capacités indiquées au Tableau 3.14. En outre, nous utilisons une constante de qualité de service $K = 0.6$, qui limite le délai sur chaque liaison à 4.0 ms et nous fixons à nouveau la taille moyenne des paquets ($1/\mu$) à 1000 bits (Baransel *et al.*, 1995). En principe, l'acheminement des paquets et l'affectation de flots se réalisent suivant le principe exposé à la section 3.4. D'abord, nous déterminons toutes les paires de noeuds adjacents pour y acheminer les trafics correspondants, ce qui permet de déterminer respectivement le flot, la capacité et les délais provisoires d'attente, de transmission et de propagation sur chaque liaison. Cette détermination d'attributs provisoires de liaisons permet d'acheminer le trafic restant selon les chemins les plus rapides. L'application de cet algorithme à ce problème conduit à un délai moyen très faible (moins de 6.0 ms par paquet), ce qui témoigne de son efficacité et de son adaptabilité aux réseaux à haut débit.

3.7 Résultats de mise en oeuvre

Dans cette section, nous exposons, dans un premier temps, les résultats obtenus par l'application de notre algorithme à des réseaux à débit peu élevé et, dans un deuxième temps, nous présentons les résultats obtenus d'un réseau à haut débit.

3.7.1 Résultats obtenus sur des réseaux à débit moyen

Pour la mise en oeuvre de notre approche sur des réseaux à débit moyen, nous utilisons des patrons de trafic relativement peu élevés, pouvant atteindre un maximum de 50 paq/s entre une paire de source/destination. Les résultats sont regroupés en deux séries: d'abord, nous comparons la performance de notre algorithme à celle des heuristiques statiques de routage, telles que le routage de plus courte distance et le routage à nombre minimal de hops; ensuite, nous nous mesurons aux méthodes classiques de routage optimal, de façon à nous situer par rapport à la solution optimale.

Pour notre première série de tests, nous considérons un réseau de 10 noeuds et de 14 liaisons (illustré à la Figure 3.6), dont les coordonnées cartésiennes sont représentées au Tableau 3.1. En outre, nous supposons une matrice de trafic uniforme et utilisons les options de capacités indiquées au Tableau 3.2. Il convient alors de rappeler que, dans ce mémoire, chaque algorithme de routage réalise intrinsèquement une affectation de capacités.

Le premier test utilise un trafic uniforme de 10 paq/s. Pour un tel trafic, nous calculons d'abord le délai moyen T_{moy} de chaque paquet, qui constitue le paramètre à minimiser. Ensuite, nous choisissons deux paires de source/destination et déterminons pour chacune d'entre elles le chemin suivi, ainsi que le temps de traversée du réseau. Toutes ces opérations sont exécutées pour le routage de plus courte distance, pour le routage à nombre minimal de hops et pour notre approche, ce qui nous permet de comparer les résultats déterminés par ces trois heuristiques.

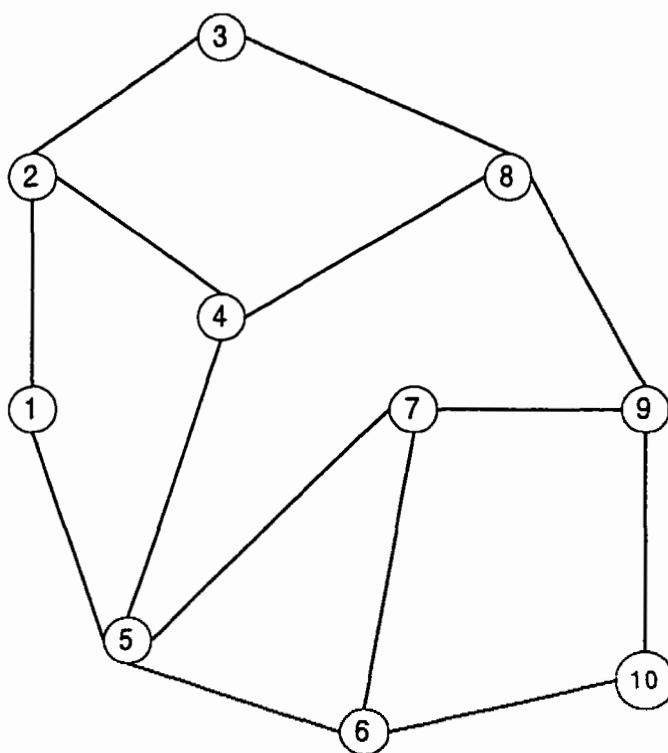


Figure 3.6 Sous-réseau de transport (10 noeuds et 14 liaisons)

Tableau 3.1. Coordonnées cartésiennes des noeuds

Noeud	1	2	3	4	5	6	7	8	9	10
Abscisse	20	20	40	40	30	55	60	70	85	85
Ordonnée	60	85	100	70	35	25	60	85	60	30

Tableau 3.2 Options de capacités

C_l (kbps)	9.6	19.2	50.0	100.0	230.4	460.8	921.6	1843.2
-----------------	-----	------	------	-------	-------	-------	-------	--------

Par exemple, pour un trafic uniforme de 10 paq/s, nous obtenons un délai moyen de 59.9 ms selon le routage de plus courte distance. De plus, selon cette même technique, un paquet partant du noeud 2 pour se rendre au noeud 10 emprunte le chemin composé des noeuds 2, 1, 5, 6, 10 et le délai associé à ce chemin est de 193.3 ms. Cette détermination de chemin et de délai est répétée pour la commodité (3, 6). Le Tableau 3.3 résume les résultats obtenus selon les trois heuristiques en question.

Tableau 3.3. Résultats comparatifs de routage pour un trafic de 10 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	$T_{chemin1}$ (ms)	S_2/D_2	Chemin2	$T_{chemin2}$ (ms)
Plus courte distance	59.9	2-10	2-1-5-6-10	193.3	3-6	3-2-1-5-6	152.3
Nombre minimal de hops	38.6	2-10	2-4-8-9-10	84.9	3-6	3-2-1-5-6	97.2
Nouvelle approche	32.4	2-10	2-1-5-6-10	40.4	3-6	3-2-1-5-6	43.2

Nous répétons toutes les opérations résumées au Tableau 3.3, mais en considérant d'autres matrices de trafic (toujours uniforme), de façon à analyser le comportement de chacune des heuristiques face au changement de trafic. Les divers résultats sont présentés dans les tableaux 3.4 à 3.11 ainsi qu'en Annexe I, section AI.2. Nous nous rendons compte que, dans tous les cas étudiés, notre approche produit de meilleurs résultats que les autres heuristiques. La Figure 3.7 illustre les variations du délai moyen T_{moy} en fonction du niveau de trafic considéré, ce pour chacune des heuristiques de routage considérées aux tableaux précédents. Il est donc évident que la nouvelle approche proposée donne en général de meilleurs résultats, comparée aux deux autres.

Tableau 3.4 Résultats comparatifs de routage pour un trafic de 15 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	26.6	2-10	2-1-5-6-10	27.5	3-6	3-2-1-5-6	38.2
Nombre minimal de hops	74.0	2-10	2-4-8-9-10	76.0	3-6	3-2-1-5-6	228.2
Nouvelle approche	22.2	2-10	2-3-8-4-5-6-10	29.0	3-6	3-8-4-5-6	18.6

Tableau 3.5 Résultats comparatifs de routage pour un trafic de 20 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	37.5	2-10	2-1-5-6-10	85.8	3-6	3-2-1-5-6	76.2
Nombre minimal de hops	32.8	2-10	2-4-8-9-10	134.5	3-6	3-2-1-5-6	37.8
Nouvelle approche	14.8	2-10	2-1-5-6-10	30.8	3-6	3-2-1-5-6	23.4

Tableau 3.6 Résultats comparatifs de routage pour un trafic de 25 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	18.3	2-10	2-1-5-6-10	54.4	3-6	3-2-1-5-6	27.7
Nombre minimal de hops	20.3	2-10	2-4-8-9-10	44.9	3-6	3-2-1-5-6	54.2
Nouvelle approche	10.2	2-10	2-4-8-9-10	15.9	3-6	3-8-9-10-6	19.3

Tableau 3.7 Résultats comparatifs de routage pour un trafic de 30 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin 2	T_{chemin2} (ms)
Plus courte distance	11.3	2-10	2-1-5-6-10	13.7	3-6	3-2-1-5-6	19.1
Nombre minimal de hops	37.0	2-10	2-4-8-9-10	38.0	3-6	3-2-1-5-6	114.1
Nouvelle approche	11.3	2-10	2-4-8-9-10	21.7	3-6	3-2-1-5-6	23.4

Tableau 3.8 Résultats comparatifs de routage pour un trafic de 35 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	21.7	2-10	2-1-5-6-10	19.4	3-6	3-2-1-5-6	38.4
Nombre minimal de hops	63.5	2-10	2-4-8-9-10	23.9	3-6	3-2-1-5-6	184.4
Nouvelle approche	8.1	2-10	2-1-5-6-10	9.6	3-6	3-8-4-5-6	9.6

Tableau 3.9 Résultats comparatifs de routage pour un trafic de 40 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	18.7	2-10	2-1-5-6-10	42.9	3-6	3-2-1-5-6	38.1
Nombre minimal de hops	16.3	2-10	2-4-8-9-10	67.2	3-6	3-2-1-5-6	18.9
Nouvelle approche	6.3	2-10	2-3-8-9-10	10.4	3-6	3-2-1-5-6	10.5

Tableau 3.10 Résultats comparatifs de routage pour un trafic de 45 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	56.7	2-10	2-1-5-6-10	62.3	3-6	3-2-1-5-6	55.0
Nombre minimal de hops	19.7	2-10	2-4-8-9-10	13.3	3-6	3-2-1-5-6	32.0
Nouvelle approche	6.4	2-10	2-4-8-9-10	8.2	3-6	3-8-9-10-6	10.7

Tableau 3.11 Résultats comparatifs de routage pour un trafic de 50 paq/s

Heuristiques de routage	T_{moy} (ms)	S_1/D_1	Chemin1	T_{chemin1} (ms)	S_2/D_2	Chemin2	T_{chemin2} (ms)
Plus courte distance	9.2	2-10	2-1-5-6-10	27.2	3-6	3-2-1-5-6	13.9
Nombre minimal de hops	9.9	2-10	2-4-8-9-10	22.4	3-6	3-2-1-5-6	27.1
Nouvelle approche	8.7	2-10	2-4-8-9-10	9.4	3-6	3-8-9-10-6	14.0

Ces résultats nous ont alors motivés à comparer notre approche aux algorithmes classiques de routage optimal, tels que la méthode FD et l'algorithme BG. Cependant, les données du problème ne sont pas les mêmes qu'auparavant: nous considérons cette fois-ci un réseau de 8 noeuds et de 12 liaisons, formé de quelques grandes villes américaines et représenté à la Figure 3.8. Les coordonnées des noeuds sont extraites de *Telecommunications Network Design Algorithms* de Kershenbaum (1993) et représentées en Annexe I, section AI.3. En outre, nous supposons une matrice de trafic uniforme de 2 paq/s avec $1/\mu=1000$ bits/paq, et des options de capacités indiquées au Tableau 3.12. Pour

fins de comparaison, nous déterminons, en plus du délai moyen de paquet, le temps d'exécution, le nombre d'itérations exécutées, ainsi que la somme des capacités utilisées par chaque algorithme.

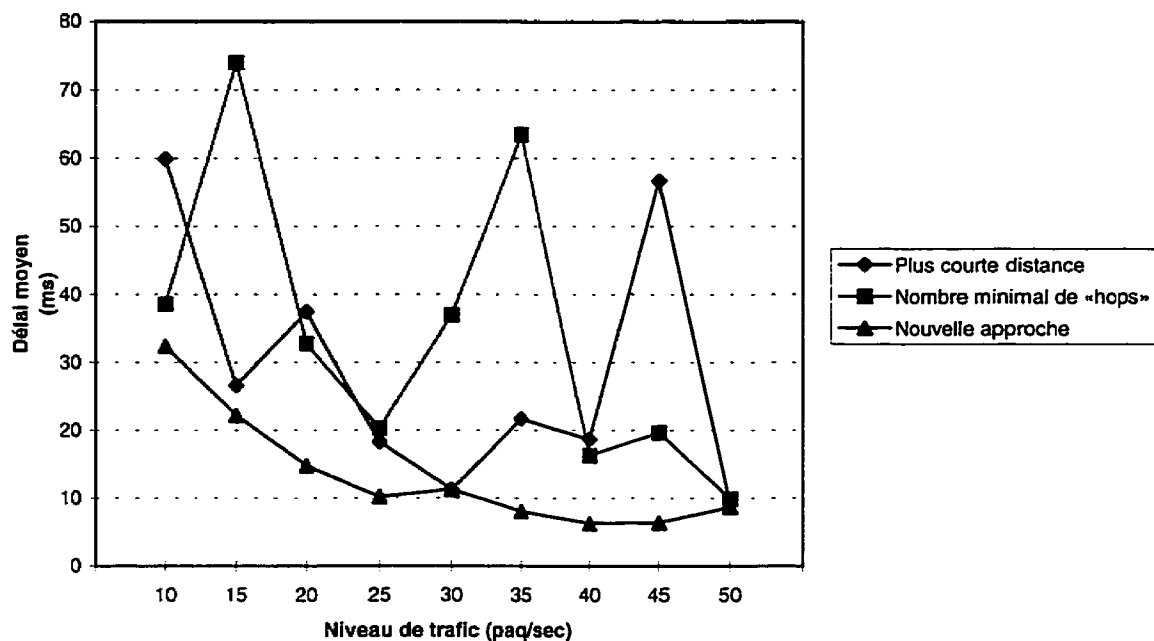


Figure 3.7 Variations du délai moyen de paquet en fonction du niveau de trafic

Tableau 3.12 Matrice des options de capacités

C_i (kbps)	9.6	19.2	28.8	38.4	48.0	57.6	67.2	76.8
-----------------	-----	------	------	------	------	------	------	------

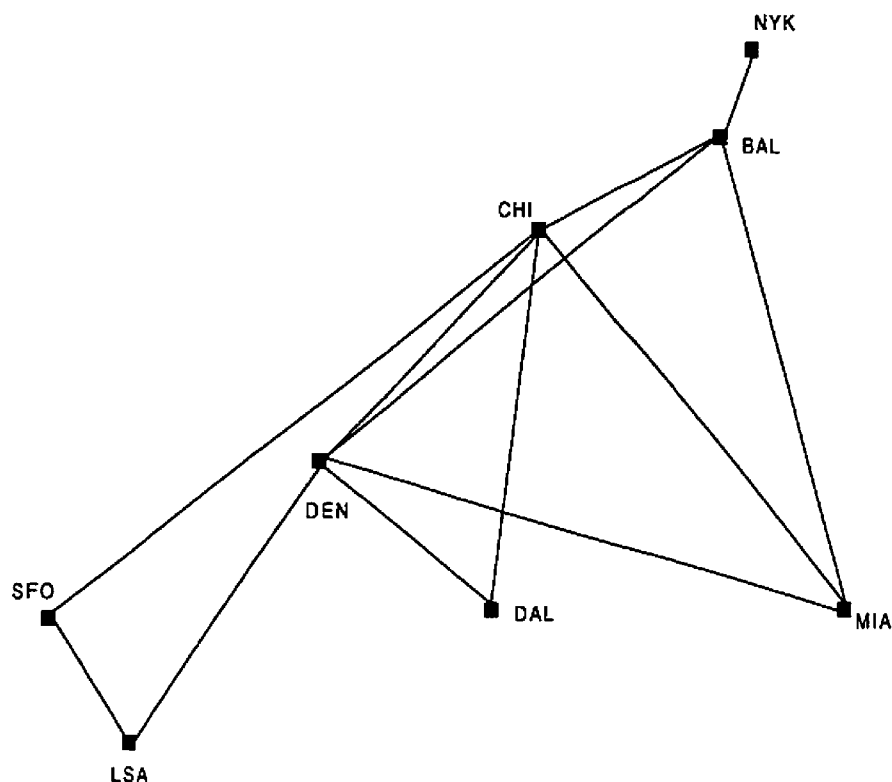


Figure 3.8 Sous-réseau de transport (8 noeuds, 12 liaisons)

Dans le but de mesurer le temps d'exécution, nous avons intégré un chronomètre dans le programme implantant les algorithmes de routage et exécuté ce programme sur un ordinateur de type *IBM PS/1 486SX*. Le Tableau 3.13 résume les résultats déterminés d'une part par les heuristiques et, d'autre part, par les algorithmes de routage optimal. Les résultats détaillés sont présentés en Annexe I, section A1.3. Nous nous rendons compte que le routage des paquets selon notre approche aboutit à un délai meilleur que celui fourni par les heuristiques concurrentes et très proche de celui calculé par les techniques de routage optimal, ce qui confirme la bonne qualité de notre algorithme.

Tableau 3.13 Comparaison des algorithmes de routage

Algorithme de routage	Nombre d'itérations	Temps d'exécution (s)	T_{moy} (s)	Capacité totale (kbps)
Plus courte distance	1	0.110	0.795	240.0
Nombre minimal de hops	1	0.110	0.627	249.6
Nouvelle approche	1	0.110	0.202	297.6
Déviaton de flot	14	5.54	0.190	240.0
Bertsekas-Gallager	16	13.94	0.187	240.0

Par ailleurs, la capacité totale utilisée par le réseau correspond à la somme des capacités de liaisons et, de ce fait, caractérise le coût total des liaisons de communications (section 2.1.6). S'il est vrai que, pour le réseau de la Figure 3.8, notre algorithme utilise des capacités plus élevées que les autres stratégies, il est aussi vrai que cette augmentation globale de la largeur de bande est largement compensée par le gain obtenu en termes de délai. Par rapport au temps d'exécution, les trois heuristiques comportent la même rapidité, mais demeurent nettement plus rapides que les algorithmes de routage optimal. Toutefois, il convient de mentionner que, tels qu'implantés, ces derniers ne conduisent pas tout à fait à la solution optimale à cause des options discrètes de capacités et du choix des paramètres utilisés. Il en résulte d'ailleurs un faible écart dans la détermination du délai moyen de paquet.

3.7.2 Résultats obtenus à haut débit

Notre algorithme de routage est à nouveau implanté en Turbo C, version 3.0. En fait, les principales fonctions de traitement demeurent les mêmes que pour les réseaux à faible débit. La différence majeure réside dans la détermination du délai sur chaque liaison qui, à haut débit, comporte à la fois les temps d'attente, de transmission et de propagation, tels que exprimés par la relation (3.25).

Nous appliquons alors notre algorithme au réseau de la Figure 3.9, en supposant une matrice de trafic uniforme de 150 paq/s, en utilisant les options de capacités du Tableau 3.14 et une taille moyenne de 1000 bits par paquet et en limitant le délai sur chaque liaison à 4.0 ms. Il en résulte un délai moyen de 5.7113 ms par paquet. Les résultats complets des attributs de liaisons de chacune d'entre elles sont présentés au Tableau 3.15 qui indique, entre autres, le flot, la capacité, les délais de transmission et de propagation, ainsi que le délai total sur chaque liaison, ce dernier étant composé des délais d'attente, de transmission et de propagation.

Tableau 3.14 Options de capacités à haut débit

Signal	DS-1	DS-1C	DS-2	2 DS-2	12 DS-1	4 DS-2	DS-3	2 DS-3	4 DS-3	DS-4
C_l (Mbps)	1.544	3.152	6.312	12.624	18.528	25.248	44.736	89.472	178.94	274.18

Tableau 3.15 Attributs de liaisons à haut débit

Liaison	Flot (Mbps)	Capacité (Mbps) (Signal)	$T_{\text{transmission}}$ (ms)	$T_{\text{propagation}}$ (ms)	T_{total} (ms)
(1, 3)	4.200	6.312 DS-2	0.1584	0.3804	0.8539
(1, 4)	5.550	6.312 DS-2	0.11584	0.3779	1.6902
(1, 5)	8.850	12.624 2 DS-2	0.0792	0.4249	0.6899
(2, 3)	2.850	3.152 DS-1C	0.3173	0.3480	3.6593
(2, 6)	13.950	18.528 12 DS-1	0.0540	0.5336	0.7520
(2, 7)	12.300	12.624 2 DS-2	0.0792	0.5336	3.6200
(3, 7)	2.850	3.152 DS-1C	0.3173	0.3167	3.6279
(4, 11)	3.900	6.312 DS-2	0.1584	0.6549	1.0695
(4, 18)	6.150	12.624 2 DS-2	0.0792	1.3858	1.5403
(5, 9)	10.050	12.624 2 DS-2	0.0792	0.3727	0.7612
(5, 11)	4.050	6.312 DS-2	0.1584	0.5270	0.9691
(6, 8)	2.700	3.152 DS-1C	0.3173	0.3887	2.6011
(6, 12)	11.850	12.624 2 DS-2	0.0792	0.5069	1.7989
(7, 9)	11.850	12.624 2 DS-2	0.0792	0.2635	1.5555
(8, 10)	3.450	6.312 DS-2	0.1584	0.3308	0.6802
(8, 13)	3.750	6.312 DS-2	0.1584	0.5207	0.9110
(9, 13)	11.400	12.624 2 DS-2	0.0792	0.3145	1.1315
(10, 12)	11.550	12.624 2 DS-2	0.0792	0.1863	1.1174
(10, 15)	9.450	12.624 2 DS-2	0.0792	0.4167	0.7317
(11, 14)	6.000	6.312 DS-2	0.1584	0.2357	3.4408
(12, 16)	8.400	12.624 2 DS-2	0.0792	0.6194	0.8561
(13, 14)	9.450	12.624 2 DS-2	0.0792	0.3399	0.6550
(14, 17)	5.850	6.312 DS-2	0.1584	0.3727	2.5372
(14, 18)	5.850	6.312 DS-2	0.1584	0.6346	2.7991
(15, 17)	5.850	6.312 DS-2	0.1584	0.3005	2.4650
(15, 20)	2.850	3.152 DS-1C	0.3173	0.7683	4.0796
(16, 20)	5.100	6.312 DS-2	0.1584	0.3598	1.1849
(17, 19)	1.200	1.544 DS-1	0.6477	0.5590	3.4660
(18, 19)	9.000	12.624 2 DS-2	0.0792	0.3436	0.6195
(19, 20)	5.700	6.312 DS-2	0.1584	0.6719	2.3058
Délai moyen des paquets $T_{\text{moy}} = 5.7113 \text{ ms}$					

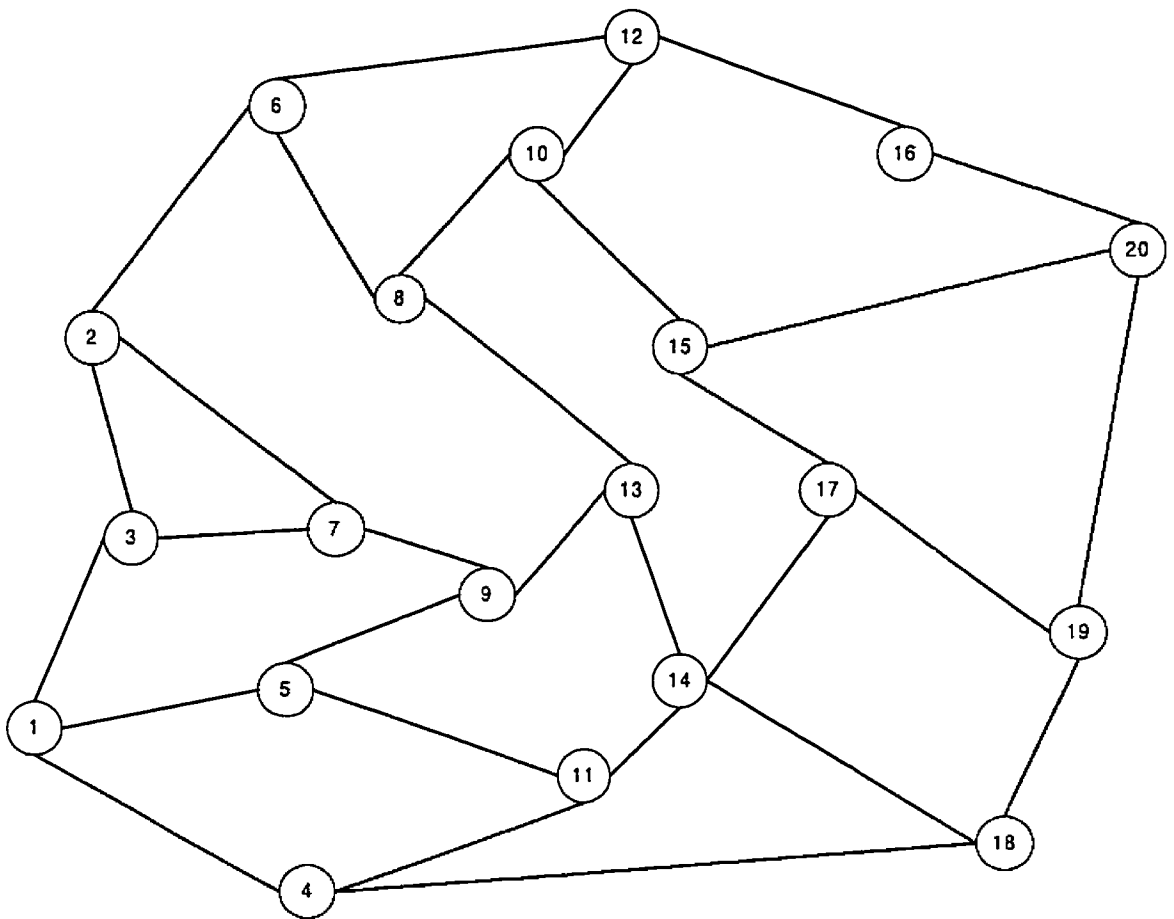


Figure 3.9 Configuration d'un réseau à haut débit (20 noeuds, 30 liaisons)

3.8 Analyse des résultats et conclusion

Comme nous l'avons montré, la première étape de notre approche réduit considérablement le nombre de paquets dans le réseau par l'acheminement direct du trafic de certaines commodités. Par conséquent, cette réduction de flot contribue à minimiser la charge du réseau. D'autre part, l'ajout d'un seuil au délai de chaque liaison nous assure d'être le plus proche possible de l'optimalité, indépendamment de la matrice de trafic, ce qui d'ailleurs apparaît très clairement dans les divers tableaux comparatifs présentés

précédemment. L'inconvénient majeur réside dans le choix des capacités du réseau de la Figure 3.8, capacités qui sont légèrement plus élevées lorsqu'elles sont déterminées par notre approche. Cet inconvénient est toutefois compensé par l'amélioration obtenue du délai moyen (par rapport aux 2 premiers algorithmes) et du temps d'exécution (par rapport à FD et BG).

Quant au routage de plus courte distance, il ne fait aucune considération du flot ou du trafic sur les liaisons. C'est plutôt une méthode *gloutonne* se basant simplement sur la notion de distance entre les noeuds pour acheminer le trafic des commodités. Cette simplicité rend toutefois l'algorithme très rapide, mais ne conduit à une bonne solution que pour certains patrons de trafic. Par exemple, pour un trafic uniforme de 30 paq/s, le routage de plus courte distance aboutit à un délai de 11.3 ms, alors que pour une matrice de 50 paq/s, ce même algorithme induit un délai de 9.2 ms. Mais, ce ne sont là que des résultats plutôt conjoncturels, car il suffit de passer de 50 paq/s à 45 paq/s pour voir le délai atteindre 56.7 ms, ce qui montre une certaine instabilité dans la technique et qui prouve que le plus court chemin ne constitue pas nécessairement le chemin le moins congestionné.

Par ailleurs, en considérant que chaque liaison est de longueur unitaire, le routage de plus courte distance devient équivalent au routage à nombre minimal de hops. Dans ce cas, ces deux heuristiques présentent les mêmes caractéristiques. Autrement dit, le routage à nombre minimal de hops ne garantit pas non plus une bonne solution au problème. Par exemple, pour un trafic de 50 paq/s, il détermine un délai moyen de 9.9 ms par paquet, alors que, dans les mêmes conditions mais pour un trafic de 15 paq/s, ce

même algorithme conduit à un délai de 74.0 ms. Ainsi, même si l'on pense diminuer le délai d'attente et de transmission en acheminant les paquets par le nombre minimal de liaisons intermédiaires, les résultats obtenus dépendent en grande partie de la matrice de trafic considérée.

L'étude comparative des résultats des trois différentes heuristiques de routage montre que, pour un même niveau de trafic, le délai moyen calculé selon notre approche est toujours meilleur que celui déterminé par les deux autres algorithmes. D'autre part, pour un même niveau de trafic, le temps pris par un paquet pour se rendre d'une source à une destination est généralement plus faible si le chemin est déterminé par notre algorithme que s'il est choisi par des heuristiques statiques. Enfin, nous nous rendons compte que le délai varie beaucoup pour ces dernières: de 9.2 ms à 59.9 ms pour le routage de plus courte distance, de 9.9 ms à 74.0 ms pour le routage à nombre minimal de hops, alors que pour notre approche il se situe entre 6.3 ms et 32.4 ms, ce qui prouve que notre approche est plus stable que les autres heuristiques de routage.

Appliqués au réseau de la Figure 3.8 (pour un trafic de 2 paq/s), les algorithmes de routage optimal partent d'un flot réalisable conduisant à un délai initial de 246 ms, qu'ils tentent de réduire de façon itérative. Le nombre d'itérations et le temps d'exécution sont essentiellement déterminés par une tolérance sur le délai fixé à 0.001 et qui constitue la condition d'arrêt. L'augmentation de cette tolérance conduit éventuellement à une convergence plus rapide de ces algorithmes, mais à des résultats moins précis et moins proches de l'optimalité.

Les résultats montrent que l'algorithme BG est beaucoup plus lent que l'algorithme FD. En effet, l'algorithme BG doit non seulement déterminer la valeur du pas α à chaque itération, mais aussi calculer les dérivées partielles secondes du délai par rapport au flot de chaque liaison et trouver les meilleurs chemins entre toutes les paires de noeuds, avant de déplacer le flot vers de tels chemins. Toutes ces opérations exigent beaucoup d'espace-mémoire et de temps de calcul. C'est d'ailleurs la raison pour laquelle l'exécution de l'algorithme BG se fait en 13.94 s et 16 itérations pour déterminer le routage optimal du petit réseau de la Figure 3.8.

Pour ce même réseau, l'algorithme FD s'exécute en 14 itérations et 5.54 s. Évidemment, le temps d'exécution est moindre, mais la convergence vers la solution optimale s'avère plus lente. En effet, tel qu'implanté, cet algorithme atteint la condition d'arrêt bien avant sa convergence. C'est pourquoi il s'arrête à un délai de 190 ms. Ainsi, la comparaison des deux algorithmes de routage optimal pour ce réseau montrent que la méthode FD est plus rapide, mais que BG donne légèrement de meilleurs résultats. De tels résultats sont détaillés en Annexe I, section AI.3.

Ce chapitre nous a permis de comprendre combien il est important de concevoir un bon algorithme de routage lors de la planification d'un réseau distribué. Nous avons vu qu'il existe plusieurs catégories de ces algorithmes qui, dépendant de leurs critères de performance, comportent leurs avantages et leurs inconvénients. Ainsi, nous nous sommes surtout inspirés des avantages pour concevoir notre algorithme de routage, implémentant dynamiquement une méthode statique et conduisant à des résultats sous-optimaux. Plus spécifiquement, notre algorithme s'avère simple, de complexité

polynomiale et tend à minimiser le délai de chaque paquet, en évitant le plus possible les chemins congestionnés.

Par ailleurs, nous avons aussi étudié l'impact du débit élevé sur le routage et montré comment notre stratégie peut s'y adapter. Toutefois, nous n'avons jusqu'à présent considéré que des réseaux dont les composantes sont toujours fonctionnelles. Or, pratiquement, les noeuds et les liaisons d'un réseau ont une certaine probabilité de tomber en panne. Il serait donc intéressant d'étudier le comportement d'un tel réseau lorsque l'une de ses composantes devient défaillante pendant l'acheminement des paquets. Ce sera l'objet de notre prochain chapitre.

CHAPITRE IV

INTÉGRATION DES ASPECTS

DE ROUTAGE ET DE SURVIVABILITÉ

En général et par souci de simplicité, les concepteurs d'algorithmes de routage se réfèrent à des réseaux fonctionnant dans des conditions normales, sans considération de défaillance de liaisons ou de noeuds, ce qui donne lieu à des modèles à la fois simplistes et idéalistes (Bertsekas & Gallager, 1987; Fratta *et al.*, 1974; Gavish & Neuman, 1989). Toutefois, en pratique, chacune des composantes d'un réseau comporte une probabilité non nulle de tomber en panne, dépendant de son niveau de *fiabilité* ou de *survivabilité* (Wang & Schwartz, 1993). Il devient ainsi impérieux d'en tenir compte dans la conception d'un algorithme de routage. Dans ce chapitre, nous considérons un modèle de réseau plus réaliste et, du même coup, nous généralisons le problème de routage, en y intégrant l'aspect de survivabilité. Dans un premier temps, nous abordons quelques aspects relatifs à la fiabilité et à la survivabilité dans les réseaux. Puis, nous présentons les caractéristiques spécifiques d'un modèle général d'intégration, avant d'introduire une nouvelle méthode basée sur la notion de chemins disjoints. Nous en décrirons alors les différentes étapes d'implémentation, ainsi que les procédures de mise en oeuvre, ce qui nous amènera à l'analyse des résultats obtenus de cette méthode d'intégration.

4.1 Fiabilité et survivabilité des réseaux

La fiabilité et la survivabilité constituent deux paramètres importants en conception de réseaux à haut débit, dans la mesure où ces derniers doivent demeurer en tout temps opérationnels et maintenir un niveau de performance élevé (Kubat, 1991). Présentons une caractérisation de ces deux paramètres.

4.1.1 Considérations de fiabilité

À cause de ses diverses interprétations qui dépendent du réseau considéré, des causes et des types de défaillance, et des niveaux d'opération, il est difficile de définir la fiabilité d'un réseau de manière exacte et précise (Kubat, 1991). La définition la plus générale provient de Colbourn (1991) qui considère la fiabilité comme un indicateur quantitatif de la capacité du réseau à demeurer fonctionnel. Cette définition donne lieu à plusieurs méthodes de calcul. Par exemple, Aggarwal *et al.* (1975) proposent une méthode permettant de calculer la probabilité que le réseau demeure fonctionnel, alors que Wang et Schwartz (1993) choisissent de déterminer la probabilité de défaillance de chacune des liaisons pour exprimer la fiabilité. Toutefois, dans la plupart des cas, le processus aléatoire traduisant la défaillance des composantes conduit à des expressions très complexes de mesures de fiabilité.

Le comportement probabiliste d'un réseau étant difficile à analyser, certains auteurs (Pierre, 1991; Schumacher, 1984) se réfèrent à la K -connexité pour exprimer la fiabilité. Une telle mesure renvoie à deux problèmes classiques: génération d'un graphe

K -connexe et élaboration de règles de perturbations permettant de préserver la K -connexité (Hyppolite, 1994; Legault, 1994). Toutefois, l'utilisation de cette mesure déterministe de fiabilité est limitée, ne prenant pas en compte les caractéristiques opérationnelles du réseau ni le caractère aléatoire des défaillances.

Par ailleurs, la fiabilité peut être aussi exprimée par la qualité ou la disponibilité du service offert par le réseau (Kubat, 1991). Cette définition est surtout adoptée par les administrateurs qui caractérisent un réseau fiable par un système comportant un risque de défaillance quasiment nul. Combinant la durée moyenne de fonctionnement sans défaillance et la durée moyenne d'une défaillance, nous pouvons exprimer la disponibilité d'un réseau par la relation (Kubat, 1991):

$$A = \frac{MTBF}{MTBF + MTTR} \quad (4.1)$$

où:

- $MTBF$ (Mean Time Between Failure) désigne la durée moyenne de fonctionnement sans défaillance; et
- $MTTR$ (Mean Time To Repair) la durée moyenne d'une panne.

Nous en déduisons qu'un réseau est d'autant plus fiable que son $MTBF$ est élevé et que son $MTTR$ est faible.

Il convient toutefois de mentionner que tous ces modèles d'analyse mettent très peu d'emphase sur l'aspect opérationnel du réseau après une défaillance, ce qui conduit à une certaine ignorance de l'état de ce dernier lorsque survient une panne. Il importe alors d'étudier le comportement d'un tel réseau à la suite de cette défaillance.

4.1.2 Considérations de survivabilité

Selon Newport (1991), la survivabilité d'un réseau traduit la capacité de ce dernier à demeurer stable après la perte de l'une de ses composantes (noeuds ou liaisons). Ainsi, contrairement à l'étude de fiabilité qui détermine le comportement du réseau et calcule sa disponibilité en fonctionnement normal, la survivabilité caractérise le comportement de ce même réseau après défaillance. De ce fait, un réseau peut être fiable, mais non *survivable*. En effet, après la perte de l'une de ses composantes, un réseau peut demeurer encore connexe, mais la défaillance subie peut conduire à la dégradation des performances. Une telle dégradation se traduit souvent par un délai excessif des paquets sur certaines liaisons et peut mener à l'interruption de la communication entre certaines paires de noeuds (Yokohira *et al.*, 1991).

L'étude de la survivabilité considère un réseau comme un système fonctionnant selon un certain nombre d'états: l'état E_0 traduisant le fonctionnement normal et l'état E_f caractérisant une certaine défaillance. Dans cet ordre d'idée, Li (1984) propose une méthode permettant de déterminer les k états les plus probables d'un réseau et, moyennant certaines hypothèses, en étudie le comportement dans un état E_i , où $1 \leq i \leq k$. Ce qui permet de déterminer des bornes inférieures et supérieures du délai moyen des paquets à l'état E_i . Quant à Yokohira (1991), il propose une heuristique qui détermine la capacité $C(l, E_i)$ de chaque liaison l dans un état E_i , ce qui permet de calculer la capacité de chaque liaison l répondant aux contraintes de survivabilité du réseau.

Il est évident qu'à la suite de la défaillance d'une liaison, tous les paquets devant y transiter doivent être déviés sur d'autres liaisons disponibles, ce qui constitue le

reroutage de ces paquets. Les méthodes de survivabilité que nous avons présentées n'en tiennent pas compte et se basent uniquement sur la probabilité de défaillance de chaque liaison pour déterminer les états possibles et le comportement du réseau dans un état E_i . En procédant ainsi, elles ignorent le caractère aléatoire des défaillances. Il importe alors de considérer un modèle général de survivabilité qui, en plus de tenir compte du processus aléatoire des défaillances, effectue le reroutage des paquets en transit sur la liaison défaillante.

4.2 Survivabilité et routage

Gavish (1992) propose un modèle général d'intégration de routage et de survivabilité permettant de refaire l'affectation de flots après une défaillance et d'exprimer la variation de flot sur les liaisons non défaillantes. Ce modèle nécessite toutefois une série d'hypothèses qu'il convient d'abord de présenter.

4.2.1 Hypothèses

Les hypothèses habituelles d'un problème classique de routage restent les mêmes. En effet, nous considérons une configuration topologique représentée par un graphe $G = (N, A)$ où les paquets, exponentiellement distribués, se présentent à l'entrée de chaque liaison selon un processus de Poisson. En outre, nous supposons infinie la mémoire tampon de chaque liaison et le délai de traitement aux noeuds négligeable.

Il importe d'y ajouter toutefois un certain nombre d'hypothèses de survivabilité. En particulier, la probabilité de défaillance des noeuds est à toutes fins pratiques négligeable (Wang & Schwartz, 1993; Yokohira *et al.*, 1991). Ce qui sous-entend que le modèle tient uniquement compte des défaillances de liaisons. De plus, pour faciliter la résolution du problème, les hypothèses suivantes sont nécessaires (Gavish & Neuman, 1992):

Hypothèse 1: les défaillances sont indépendantes les unes des autres;

Hypothèse 2: la durée moyenne de fonctionnement sans défaillance d'une liaison est exponentiellement distribuée;

Hypothèse 3: une seule liaison peut être défaillante à un instant donné. Il en résulte qu'aucune liaison ne peut tomber en panne tant que la liaison défaillante ne soit réparée;

Hypothèse 4: le MTTR est négligeable par rapport au MTBF;

Hypothèse 5: le temps moyen d'interarrivée des paquets est beaucoup plus faible que celui de réparation.

Remarquons que l'hypothèse 1, quoique très approximative (il existe une certaine corrélation entre les occurrences de défaillances), contribue à réduire considérablement la complexité du modèle. Par ailleurs, les hypothèses 4 et 5 renforcent l'hypothèse 3, en réduisant substantiellement la probabilité d'avoir plus d'une liaison défaillante dans le réseau. Cela est d'ailleurs statistiquement vérifié dans le réseau d'informations d'IBM (IBM/IN), où le délai moyen de fonctionnement sans défaillance (231.69 h) s'avère

nettement supérieur à la durée moyenne d'une défaillance qui est de 0.55 h (Gavish & Neuman, 1992).

D'autre part, lorsque survient une défaillance, le réseau passe d'un état stable à un autre, en passant par un phénomène transitoire pendant lequel les noeuds sont avertis de la non disponibilité de la liaison défaillante et doivent décider de *réacheminer* certains paquets. Dans le même ordre d'idée, après une réparation, le retour à l'état normal de fonctionnement passe aussi par un régime transitoire. La considération d'un tel processus complique énormément le problème, ce qui nous porte à ignorer le régime transitoire dans le modèle d'intégration. D'ailleurs, les hypothèses 4 et 5 nous assurent que le régime permanent est rapidement atteint et, dans la mesure qu'elles sont respectées, une telle simplification n'affecte pas significativement les résultats traduisant le comportement du réseau (Gavish & Neuman, 1992).

4.2.2 Modèle d'intégration

Le modèle d'intégration proposé par Gavish (1992) détermine essentiellement pour chaque paire de noeuds un chemin principal et un ensemble de chemins de secours, de façon à utiliser l'un de ces derniers toutes les fois que le chemin principal n'est pas disponible. Mais avant de présenter les paramètres de ce modèle, examinons la manière dont interviennent les trois couches *Liaison*, *Réseau* et *Session* du modèle OSI à la suite d'une défaillance.

Après quelques essais de transmission non réussis sur la liaison défaillante, la couche Liaison détecte la faute et en avertit la couche Réseau. Celle-ci se charge alors de

g n rer des messages de contr le   tous les noeuds utilisant respectivement cette liaison dans leur chemin principal, leur informant de la non disponibilit  de cette derni re. Toutes les files d'attente de la liaison d faillante sont alors d truites et la couche R seau, tout en s'assurant que les paquets qui formaient ces files ne soient ni perdus, ni corrompus, active les chemins de secours correspondants, sans interruption de la session. Toutefois, lorsque la liaison d faillante fait aussi partie des chemins de secours, les couches de haut niveau en sont averties, ce qui donne lieu   l'interruption de toutes les sessions utilisant les ressources de cette liaison et au rejet de toute demande d' tablissement de session devant y transiter (Gavish & Neuman, 1992).

Pr sentons maintenant les diff rentes variables du mod le d'int gration. Soit Π l'ensemble de toutes les paires source/destination. Alors, pour un r seau de n noeuds, le cardinal de Π est  gal   $n(n-1)/2$. Soit k un  l ment de Π (k repr sente une commodit ) et S_k l'ensemble des chemins qui relie la commodit  k . Le chemin principal de k est not  r , alors que q en d signe l'un des chemins de secours. En outre, d signons par l une liaison non d faillante, par f la liaison d faillante et d finissons les variables binaires suivantes:

$$x_r = \begin{cases} 1 & \text{si l'on choisit de faire transiter les paquets} \\ & \text{par le chemin principal } r \\ 0 & \text{sinon} \end{cases} \quad (4.2)$$

$$u_q = \begin{cases} 1 & \text{si les paquets empruntent le chemin} \\ & \text{de secours } q \\ 0 & \text{sinon} \end{cases} \quad (4.3)$$

$$\delta_{rl} = \begin{cases} 1 & \text{si la liaison } l \text{ fait partie du chemin principal } r \\ 0 & \text{sinon} \end{cases} \quad (4.4)$$

Nous pouvons alors considérer l'état E_0 comme celui traduisant le fonctionnement normal du réseau. Dans cet état, les paquets empruntent leur chemin principal respectif. Par contre, si la liaison f devient défaillante, le réseau passe à l'état E_f et les paquets devant emprunter la liaison f doivent transiter par leur chemin de secours. Il en résulte une variation de flot sur la plupart des liaisons du réseau. Ainsi, l'augmentation de flot causée par la défaillance de f sur chaque liaison l du réseau s'exprime par (Gavish & Neuman, 1992):

$$\Delta f_{lf}^+ = \frac{1}{\mu} \sum_{k \in \Pi} \sum_{r \in S_k} \sum_{\substack{q \in S_k \\ q \neq r}} \gamma_k x_r u_q \delta_{rf} \delta_{ql} (1 - \delta_{af}) (1 - \delta_{rl}), \quad \forall l \neq f \in A \quad (4.5)$$

où:

- $1/\mu$ désigne la taille moyenne des paquets; et
- γ_k le trafic de la commodité k de Π .

Il en résulte un certain nombre de conditions caractérisant l'augmentation de flot sur chaque liaison l d'un chemin de secours q . Ce sont:

$$\begin{cases} x_r & = & 1 \end{cases} \quad (4.6)$$

$$\begin{cases} \delta_{rf} & = & 1 \end{cases} \quad (4.7)$$

$$\begin{cases} \delta_{af} & = & 0 \end{cases} \quad (4.8)$$

$$\begin{cases} u_q & = & 1 \end{cases} \quad (4.9)$$

$$\begin{cases} \delta_{ql} & = & 1 \end{cases} \quad (4.10)$$

$$\begin{cases} \delta_{rl} & = & 0 \end{cases} \quad (4.11)$$

D'autre part, à chaque augmentation de flot sur les liaisons de q correspond une diminution de flot sur celles de r , puisque le trafic de tout k diverge vers ses chemins de secours respectifs après la défaillance de la liaison f . Cette diminution de flot s'exprime par (Gavish & Neuman, 1992):

$$\Delta f_{lf}^- = \frac{1}{\mu} \sum_{k \in \Pi} \sum_{r \in S_k} \gamma_k x_r \delta_{rf} \delta_{rl}, \quad \forall l \neq f \in A \quad (4.12)$$

Ainsi, trois conditions caractérisent la diminution de flot sur chaque liaison l de r par suite de la défaillance de f . Ce sont:

$$\begin{cases} x_r &= 1 \\ \delta_{rf} &= 1 \\ \delta_{rl} &= 1 \end{cases} \quad \begin{matrix} (4.13) \\ (4.14) \\ (4.15) \end{matrix}$$

Cette méthode de détermination de variation de flot est puissante et générale. Par contre, le choix des différents chemins de secours n'y est pas explicite. Ce qui donne lieu à des expressions complexes de variation de flot et ne garantit nullement la communication entre une paire de noeuds, lorsque la liaison défaillante se retrouve à la fois sur le chemin principal et sur les chemins de secours utilisés par cette paire. Pour y remédier, nous proposons une nouvelle méthode d'intégration utilisant des chemins disjoints de liaisons.

4.3 Nouvelle méthode d'intégration de routage et de survivabilité

Le choix de deux chemins disjoints de liaisons vise non seulement à garantir la communication entre toutes les paires de noeuds, mais aussi à simplifier les relations traduisant la variation de flot sur chaque liaison après défaillance d'une liaison. Présentons les caractéristiques de cette nouvelle méthode d'intégration.

4.3.1 Variation de flot

En supposant un réseau suffisamment fiable (au moins 2-connexe), nous pouvons déterminer pour chaque commodité k un chemin principal r_k et un chemin de secours r'_k , les deux étant disjoints de liaisons. Un tel choix nous assure la disponibilité de r'_k après défaillance de n'importe quelle liaison de r_k , d'où la pertinence de ne choisir qu'un chemin de secours. Il en résulte une simplification considérable du problème.

Si nous désignons par S_k l'ensemble des chemins de k , alors S_k ne contient que les éléments r_k et r'_k . Par défaut, le trafic γ_k de la commodité k emprunte le chemin r_k . Par contre, lorsque ce dernier n'est pas disponible, γ_k transite par le chemin de secours r'_k . Nous en déduisons que, dans ce cas, les conditions (4.6) et (4.13) précédentes sont triviales et qu'implicitement la condition $x_r = 1$ est toujours vraie.

Par ailleurs, définissons les variables binaires suivantes pour chaque liaison l du réseau:

$$\delta_{r,l} = \begin{cases} 1 & \text{si } l \in r_k \\ 0 & \text{autrement} \end{cases} \quad (4.16)$$

$$\delta_{r_k l} = \begin{cases} 1 & \text{si } l \in r_k' \\ 0 & \text{autrement} \end{cases} \quad (4.17)$$

Les chemins r_k et r_k' étant disjoints de liaisons, la liaison défaillante f ne peut figurer que dans l'un ou l'autre chemin simultanément. En conséquence, les conditions (4.7) et (4.8) peuvent être fusionnées pour donner la condition suivante:

$$\delta_{r_k f} = 1 \quad (4.18)$$

alors que les conditions (4.10) et (4.11) peuvent être combinées en une condition unique, donnée par:

$$\delta_{r_k l} = 1 \quad (4.19)$$

En outre, puisque S_k ne contient que les éléments r_k et r_k' , les paquets transitent exclusivement par l'un ou par l'autre de ces chemins. Ainsi, lorsque r_k n'est pas disponible, les paquets de k ne peuvent transiter que par r_k' . Dans ce cas, les conditions (4.7) et (4.9) peuvent être aussi fusionnées pour produire la condition (4.18).

Nous nous rendons compte que le choix de deux chemins disjoints de liaisons réduit considérablement le nombre de conditions traduisant la variation de flot sur chaque liaison l et contribue ainsi à simplifier le modèle d'intégration de routage et de survivabilité. Aussi la défaillance d'une liaison f cause-t-elle une augmentation de flot sur l lorsque les conditions (4.18) et (4.19) sont vérifiées.

Procédons de la même façon pour la détermination des conditions de diminution de flot sur chaque liaison du réseau. En réécrivant la condition (4.15) selon notre modèle, nous obtenons la condition:

$$\delta_{r_l} = 1 \quad (4.20)$$

Alors, les conditions (4.18) et (4.20) traduisent la diminution de flot sur chaque liaison l , causée par la défaillance de f .

Si nous définissons une variable δ_l telle que:

$$\delta_l = \begin{cases} 1 & \text{si } l \in r_k \\ -1 & \text{si } l \in r_k \\ 0 & \text{autrement} \end{cases} \quad (4.21)$$

la variation de flot sur la liaison l due à la défaillance de f peut être calculée comme suit:

$$\Delta f_{lf} = \frac{1}{\mu} \sum_{k \in \Pi} \gamma_k \delta_{r_k} \delta_l, \quad \forall l \neq f \quad (4.22)$$

où:

- γ_k représente le trafic de la commodité k ;
- δ_{r_k} et δ_l deux variables binaires définies par les relations (4.16) et (4.21) respectivement.

Cette expression plus compacte résume les relations (4.5) et (4.12) adoptées par Gavish et Neuman (1992) pour exprimer la variation de flot sur chaque liaison l par suite de la défaillance de f . Elle représente une somme sur tous les γ_k et, puisqu'elle dépend de δ_l , elle peut être positive ou négative.

Sans perte de généralité, considérons le réseau de la Figure 4.1, comportant des risques de défaillance de liaisons et dont le trafic entre chaque paire de noeuds est aléatoire et représenté par une matrice $\Gamma = (\gamma_{ij})$. Le réseau étant 2-connexe, déterminons deux chemins disjoints de liaisons pour chaque paire de noeuds, ce qui nous assure la

disponibilité d'au moins un chemin lors d'une défaillance. En outre, supposons que ces deux chemins sont représentés au Tableau 4.1. Examinons le comportement du réseau dans les conditions normales de fonctionnement, puis en cas de défaillance d'une liaison.

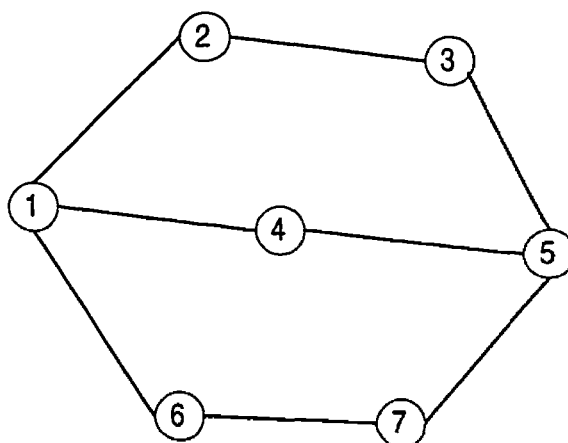


Figure 4.1 Réseau avec liaisons défaillantes (7 noeuds, 8 liaisons)

Dans les conditions normales de fonctionnement, les paquets transitent par les liaisons faisant partie de leur chemin principal respectif. Il n'en résulte pas de variation de flot. Toutefois, la défaillance de l'une des liaisons conduit à la variation du flot de la plupart des autres liaisons. Par exemple, lorsque la liaison (4, 5) tombe en panne, le trafic γ_{15} qui devrait normalement emprunter les liaisons (1, 4) et (4, 5) est dévié sur son chemin de secours formé des liaisons (1, 2), (2, 3) et (3, 5). Il en résulte une augmentation de flot de γ_{15} sur ces dernières et une diminution de γ_{15} sur (1, 4).

Tableau 4.1 Représentation des chemins

Commodité	Chemin principal	Chemin de secours
(1, 2)	(1, 2)	(1, 4, 5, 3, 2)
(1, 3)	(1, 2, 3)	(1, 4, 5, 3)
(1, 4)	(1, 4)	(1, 2, 3, 5, 4)
(1, 5)	(1, 4, 5)	(1, 2, 3, 5)
(1, 6)	(1, 6)	(1, 4, 5, 7, 6)
(1, 7)	(1, 6, 7)	(1, 4, 5, 7)
(2, 3)	(2, 3)	(2, 1, 4, 5, 3)
(2, 4)	(2, 1, 4)	(2, 3, 5, 4)
(2, 5)	(2, 3, 5)	(2, 1, 4, 5)
(2, 6)	(2, 1, 6)	(2, 3, 5, 7, 6)
(2, 7)	(2, 3, 5, 7)	(2, 1, 6, 7)
(3, 4)	(3, 5, 4)	(3, 2, 1, 4)
(3, 5)	(3, 5)	(3, 2, 1, 4, 5)
(3, 6)	(3, 2, 1, 6)	(3, 5, 7, 6)
(3, 7)	(3, 5, 7)	(3, 2, 1, 6, 7)
(4, 5)	(4, 5)	(4, 1, 2, 3, 5)
(4, 6)	(4, 1, 6)	(4, 5, 7, 6)
(4, 7)	(4, 5, 7)	(4, 1, 6, 7)
(5, 6)	(5, 7, 6)	(5, 4, 1, 6)
(5, 7)	(5, 7)	(5, 4, 1, 6, 7)
(6, 7)	(6, 7)	(6, 1, 4, 5, 7)

Par ailleurs, si nous désirons acheminer le trafic de la commodité (4, 3) par suite de la même défaillance, nous nous rendons compte que le flot de (1, 4) augmente de γ_{43} , alors que celui de (3, 5) diminue de γ_{43} . En reroutant tous les paquets devant transiter par la liaison défaillante (4, 5), nous obtenons la variation totale de flot sur les 7 liaisons non défaillantes, donnée au Tableau 4.2.

Tableau 4.2 Variation de flot après défaillance de la liaison (4, 5)

Liaison	Variation de flot
(1, 2)	$\gamma_{15} + \gamma_{34} + \gamma_{45}$
(1, 4)	$-\gamma_{15} + \gamma_{34} + \gamma_{45} + \gamma_{47}$
(1, 6)	γ_{47}
(2, 3)	$\gamma_{15} + \gamma_{34} + \gamma_{45}$
(3, 5)	$\gamma_{15} - \gamma_{34} + \gamma_{45}$
(4, 5)	-
(5, 7)	$-\gamma_{47}$
(6, 7)	γ_{47}

Nous nous rendons compte que la défaillance de la liaison (4, 5) influence le flot de toutes les liaisons non défaillantes du réseau. Généralement, la défaillance d'une liaison conduit à une augmentation de flot sur les autres liaisons. Cela n'empêche nullement d'aboutir sur certaines liaisons à une diminution de flot, comme pour la liaison (5, 7).

Comme nous l'avons mentionné, la défaillance d'une liaison constitue un phénomène aléatoire. Il s'en suit qu'à tout instant, n'importe quelle liaison peut tomber en panne, d'où l'intérêt d'étudier le comportement du réseau après défaillance d'une liaison choisie au hasard. Alors, si cette liaison défaillante devient (2, 3), nous obtenons une autre série de résultats donnant la variation de flot et présentée au Tableau 4.3. Ces résultats montrent combien la variation de flot sur les liaisons non défaillantes dépend de la liaison en panne. Ainsi, la variation de flot après défaillance sur une liaison particulière s'avère, à toutes fins pratiques, imprévisible et donne lieu à un nouveau problème. En effet, il faut dans ce cas déterminer la capacité de chaque liaison permettant d'absorber la

totalité de la variation de flot et qui, tout en maintenant une bonne performance du réseau, ne conduit guère au surdimensionnement de ce dernier.

Tableau 4.3 Variation de flot après défaillance de la liaison (2, 3)

Liaison	Variation de flot
(1, 2)	$-\gamma_{13} + \gamma_{23} + \gamma_{25} + \gamma_{27} - \gamma_{36}$
(1, 4)	$\gamma_{13} + \gamma_{23} + \gamma_{25}$
(1, 6)	$\gamma_{27} - \gamma_{36}$
(2, 3)	-
(3, 5)	$\gamma_{13} + \gamma_{23} - \gamma_{25} - \gamma_{27} + \gamma_{36}$
(4, 5)	$\gamma_{13} + \gamma_{23} + \gamma_{25}$
(5, 7)	$-\gamma_{27} + \gamma_{36}$
(6, 7)	$\gamma_{27} + \gamma_{36}$

4.3.2 Affectation de capacités

Yokohira *et al.* (1991) proposent une méthode de détermination de capacité en se basant sur la fiabilité de chaque liaison pour déterminer l'état du réseau. Toutefois, en plus d'utiliser des mesures déterministes pour calculer la capacité, cette méthode ne tient pas compte de la variation de flot causée par la liaison défaillante, ce qui s'avère pratiquement inapproprié. Nous proposons une méthode d'affectation de capacités qui dérive directement du calcul de variation de flot, tenant ainsi compte des conditions de survivabilité du réseau. Ces conditions sont telles que les liaisons non défaillantes sont en mesure d'absorber tout le trafic, sans dégradation appréciable du niveau de performance d'un tel réseau.

Tel que déjà mentionné, la défaillance d'une liaison f d'un réseau conduit à une variation de flot Δf_{lf} sur chaque liaison l , donnée par la relation (4.22). Une telle variation dépend de la liaison f . Toutefois, nous pouvons déterminer la variation maximale $\Delta f_{l_{\max}}$ de flot sur l , en calculant la variation de flot après considération de défaillance de chaque liaison $f \neq l$ du réseau et en en gardant la valeur maximale. Ce qui conduit à la relation:

$$\Delta f_{l_{\max}} = \max_{f \in A} \Delta f_{lf}$$

Remplaçant Δf_{lf} par sa valeur, nous pouvons exprimer la variation maximale de flot sur chaque liaison l par:

$$\Delta f_{l_{\max}} = \frac{1}{\mu} \max_{f \in A} \sum_{k \in \Pi} \gamma_k \delta_{r_k f} \delta_l, \quad \forall l \neq f \quad (4.23)$$

Ce qui nous permet de déterminer la capacité C_l de chaque liaison l satisfaisant aux conditions de survivabilité. Dans ce cas, C_l doit être choisie de façon à respecter la contrainte:

$$C_l \geq f_l + \Delta f_{l_{\max}} \quad (4.24)$$

où f_l désigne le flot de la liaison l lorsque le réseau fonctionne sans défaillance. Ainsi, la capacité C_l est non seulement en mesure d'absorber tout le trafic, mais permet aussi de maintenir une bonne performance du réseau, après défaillance de l'une de ses liaisons. Ce qui contribue à éviter la congestion ou l'interruption de service causée par une variation trop grande de flot sur certaines liaisons.

4.4 Implémentation et mise en oeuvre

L'intégration des aspects de survivabilité et de routage exigent plusieurs étapes d'implémentation. Présentons ces étapes, ainsi qu'une mise en oeuvre de la nouvelle méthode sur un réseau à haut débit.

4.4.1 Étapes d'implémentation

L'implémentation de notre méthode d'intégration comprend essentiellement cinq étapes. Nous nous proposons ici de les décrire brièvement.

Première étape: détermination du chemin principal r_k

Cette étape détermine le délai de chaque liaison par l'application de la relation (3.25) et vise à trouver le chemin le plus rapide r_k entre chaque paire k de noeuds. L'ensemble $\mathcal{R}$ de ces chemins constitue l'ensemble des chemins principaux et est gardé en mémoire dans un pointeur de chemins. Toutefois, les capacités déterminées à cette étape permettent seulement de calculer les attributs de liaisons nécessaires à la détermination des chemins les plus rapides et, par conséquent, ne répondent pas nécessairement aux conditions de survivabilité du réseau.

Deuxième étape: détermination du chemin de secours r'_k

Cette étape détermine, pour chaque paire k de noeuds, le second chemin le plus rapide r'_k , où r_k et r'_k sont disjoints de liaisons. Une telle étape s'avère importante dans la

mesure où, lorsque le chemin r_k n'est pas disponible, les paquets doivent emprunter r'_k et que le calcul de variation de flot sur les liaisons non défaillantes en dépend largement. Il convient aussi d'admettre que le choix de chemins disjoints de liaisons complique quelque peu l'implémentation des chemins de secours qui se fait en plusieurs phases. D'abord, pour chaque paire k de source/destination, nous déterminons l'ensemble de toutes les liaisons faisant partie du chemin principal r_k . Ensuite, nous étiquetons chacune d'entre elles par un délai infini, ce qui conduit à les éviter habilement dans la détermination de r'_k qui devient alors le chemin le plus rapide. Toutefois, il convient d'enlever les étiquettes des liaisons de r_k une fois le chemin de secours déterminé, avant de recommencer le processus pour la prochaine paire de noeuds. Comme pour l'ensemble $\mathfrak{R}$, l'ensemble $\mathfrak{R}'$ des chemins de secours r'_k est gardé en mémoire dans un pointeur de chemins différent de celui des r_k .

Troisième étape: Détermination de la variation maximale de flot et affectation de capacités

L'implémentation des relations (4.21) et (4.22) de variation de flot sur chaque liaison l suppose la connaissance respective de la liaison défaillante f , de r_k , et de r'_k pour chaque k de Π . Il s'agit alors de scruter chaque liaison l de r_k et de comparer l et f . S'il existe une liaison l de r_k égale à f , nous concluons que f fait partie de r_k . Nous soustrayons alors à chaque liaison $l \neq f$ de r_k le trafic de la commodité k et ajoutons à chaque l de r'_k ce

même trafic. Pour un f donné, cette opération se répète pour chaque k de Π . Il en résulte une variation totale de flot Δf_{lf} sur chaque liaison $l \neq f$.

Recommençons cette procédure de calcul de variation de flot en considérant une liaison défaillante f' . Il en résulte une nouvelle variation de flot $\Delta f_{lf'}$ sur chaque liaison $l \neq f'$. Comparons alors Δf_{lf} et $\Delta f_{lf'}$ et gardons en mémoire la valeur maximale des deux. Nous obtenons la variation maximale de flot possible sur chaque liaison l du réseau après considération de défaillance de f , puis de f' . Ainsi, après considération de défaillance de chacune des m liaisons, nous aboutissons à un maximum global traduisant la variation maximale de flot sur chaque liaison du réseau. Ce qui nous permet de déterminer le flot maximal de chaque liaison advenant une quelconque défaillance et de choisir la capacité requise permettant d'absorber la totalité de la variation de flot.

Quatrième étape: routage avec considération de défaillance

L'affectation de capacités effectuée à l'étape précédente modifie certains attributs de liaisons, dont le délai total de chaque liaison, exprimé par la relation (3.25). Il en résulte une nouvelle paire de chemins pour chaque paire k de Π . Le routage (avec considération de survivabilité) se fait alors selon les paires de chemins nouvellement définies.

Cinquième étape: Exécution

L'exécution de notre approche d'intégration de routage et de survivabilité suppose qu'un réseau de m liaisons peut se trouver à un instant donné dans l'un des $(m + 1)$ états: $E_0, E_1, \dots, E_m$, où l'état E_0 traduit le fonctionnement sans défaillance du réseau. Dans un tel état, les paquets transitent par leur chemin principal respectif. Toutefois, lorsque le réseau se retrouve dans un état E_f ($1 \leq f \leq m$), tous les noeuds utilisant la liaison f dans leur chemin principal doivent activer leur chemin de secours respectif. Il en résulte une certaine variation de flot sur la plupart des liaisons du réseau.

Nous présentons en Annexe II (section AII.1) les principales fonctions d'intégration des aspects de routage et de survivabilité.

4.4.2 Procédures de mise en oeuvre

Appliquons notre méthode d'intégration au réseau à haut débit de la Figure 4.2, dont le trafic uniforme est de 150 paq/s et dont les coordonnées sont spécifiées en Annexe I, section AI.4. Pour un tel réseau, il existe 190 paires de source/destination. Nous présentons en Annexe II (section AII.2) les deux chemins disjoints de liaisons les plus rapides pour chacune d'entre elles. Ces chemins sont déterminés à partir des attributs de liaisons du Tableau 4.4.

L'ensemble de ces chemins permettent de déterminer la variation maximale de flot sur chaque liaison, ainsi que la capacité correspondante. Ce qui modifie le délai de transmission, le délai total et le nombre de paquets transitant simultanément sur chaque liaison. Ces nouvelles valeurs des attributs de liaisons sont présentées au Tableau 4.5, en

supposant que le réseau se trouve dans l'état E_0 . Nous nous rendons compte qu'en général, les capacités de liaisons augmentent, permettant au réseau de satisfaire aux conditions de survivabilité. Il en résulte une diminution du délai moyen de chaque paquet qui passe de 5.7113 ms à 1.8942 ms.

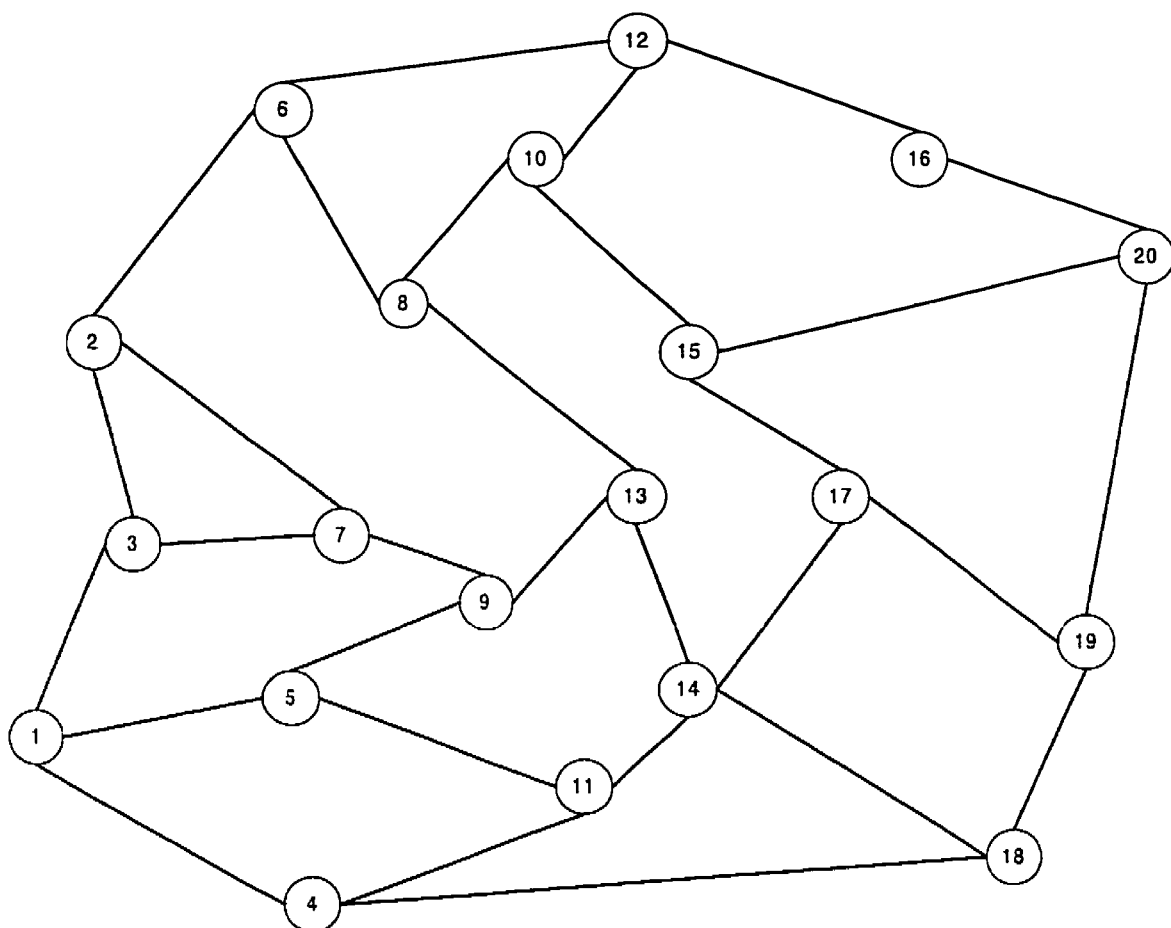


Figure 4.2 Réseau à haut débit avec liaisons défaillantes (20 noeuds, 30 liaisons)

Choisissons aléatoirement une liaison défaillante f et examinons le comportement du réseau dans l'état E_f . Pour $f = (1, 5)$, puis $f = (9, 13)$, nous présentons au Tableau 4.6 le

flot et le délai total de chaque paquet sur les liaisons non défaillantes. Nous nous rendons compte que le flot de chacune de ces liaisons dépend de la défaillance en question, mais que la capacité correspondante satisfait toujours aux conditions de survivabilité. En outre, dans les 2 cas, le délai moyen des paquets augmente, ce qui correspond à une diminution des performances globales du réseau.

Toutefois, la défaillance de la liaison (9, 13) affecte beaucoup plus le comportement du réseau que celle de (1, 5). En effet, lorsque la liaison (9, 13) est défaillante, le délai moyen des paquets passe de 1.8942 ms à 2.5405 ms, ce qui représente une augmentation de 34.0%, alors que la défaillance de la liaison (1, 5) ne conduit qu'à une augmentation d'environ 1.0% du délai moyen. Cependant, nous devons admettre que, dans les deux cas, l'augmentation du délai après défaillance reste à toutes fins pratiques acceptable.

4.5 Analyse des résultats

Dans cette section, nous analysons le comportement du réseau à haut débit de la Figure 4.2, puis nous comparons les résultats de notre méthode à ceux obtenus de la méthode proposée par Gavish et Neuman (1992) et exposée à la section 4.2.

Tableau 4.4 Attributs de liaisons sans considération de survivabilité

Liaison	Flot (Mbps)	Capacité (Mbps) (Signal)	$T_{\text{transmission}}$ (ms)	$T_{\text{propagation}}$ (ms)	T_{total} (ms)
(1, 3)	4.200	6.312 DS-2	0.1584	0.3804	0.8539
(1, 4)	5.550	6.312 DS-2	0.11584	0.3779	1.6902
(1, 5)	8.850	12.624 2 DS-2	0.0792	0.4249	0.6899
(2, 3)	2.850	3.152 DS-1C	0.3173	0.3480	3.6593
(2, 6)	13.950	18.528 12 DS-1	0.0540	0.5336	0.7520
(2, 7)	12.300	12.624 2 DS-2	0.0792	0.5336	3.6200
(3, 7)	2.850	3.152 DS-1C	0.3173	0.3167	3.6279
(4, 11)	3.900	6.312 DS-2	0.1584	0.6549	1.0695
(4, 18)	6.150	12.624 2 DS-2	0.0792	1.3858	1.5403
(5, 9)	10.050	12.624 2 DS-2	0.0792	0.3727	0.7612
(5, 11)	4.050	6.312 DS-2	0.1584	0.5270	0.9691
(6, 8)	2.700	3.152 DS-1C	0.3173	0.3887	2.6011
(6, 12)	11.850	12.624 2 DS-2	0.0792	0.5069	1.7989
(7, 9)	11.850	12.624 2 DS-2	0.0792	0.2635	1.5555
(8, 10)	3.450	6.312 DS-2	0.1584	0.3308	0.6802
(8, 13)	3.750	6.312 DS-2	0.1584	0.5207	0.9110
(9, 13)	11.400	12.624 2 DS-2	0.0792	0.3145	1.1315
(10, 12)	11.550	12.624 2 DS-2	0.0792	0.1863	1.1174
(10, 15)	9.450	12.624 2 DS-2	0.0792	0.4167	0.7317
(11, 14)	6.000	6.312 DS-2	0.1584	0.2357	3.4408
(12, 16)	8.400	12.624 2 DS-2	0.0792	0.6194	0.8561
(13, 14)	9.450	12.624 2 DS-2	0.0792	0.3399	0.6550
(14, 17)	5.850	6.312 DS-2	0.1584	0.3727	2.5372
(14, 18)	5.850	6.312 DS-2	0.1584	0.6346	2.7991
(15, 17)	5.850	6.312 DS-2	0.1584	0.3005	2.4650
(15, 20)	2.850	3.152 DS-1C	0.3173	0.7683	4.0796
(16, 20)	5.100	6.312 DS-2	0.1584	0.3598	1.1849
(17, 19)	1.200	1.544 DS-1	0.6477	0.5590	3.4660
(18, 19)	9.000	12.624 2 DS-2	0.0792	0.3436	0.6195
(19, 20)	5.700	6.312 DS-2	0.1584	0.6719	2.3058
Délai moyen des paquets $T_{\text{moy}} = 5.7113$ ms					

Tableau 4.6 Flot et délai après défaillance respective des liaisons (1, 5) et (9, 13)

Liaison	Capacité (Mbps)	Défaillance de la liaison (1, 5)		Défaillance de la liaison (9, 13)	
		Flot (Mbps)	T _{total} (ms)	Flot (Mbps)	T _{total} (ms)
(1, 3)	12.624	3.900	0.4950	6.600	0.5464
(1, 4)	12.624	3.150	0.4834	6.150	0.5323
(1, 5)	6.312	-	-	3.750	0.8152
(2, 3)	6.312	2.850	0.6369	5.550	1.6603
(2, 6)	12.624	5.850	0.6812	11.850	1.8256
(2, 7)	12.624	7.500	0.7288	9.600	0.8643
(3, 7)	12.624	5.400	0.4551	4.200	0.4354
(4, 11)	12.624	3.750	0.7676	4.050	0.7715
(4, 18)	12.624	3.000	1.4897	4.500	1.5089
(5, 9)	12.624	5.700	0.5171	5.700	0.5171
(5, 11)	6.312	2.100	0.7645	4.200	1.0005
(6, 8)	12.624	4.200	0.5074	7.500	0.5839
(6, 12)	6.312	2.850	0.7957	4.350	1.0166
(7, 9)	18.528	12.750	0.4366	8.250	0.3608
(8, 10)	18.528	11.550	0.4741	11.550	0.4741
(8, 13)	18.528	13.350	0.7138	10.650	0.6476
(9, 13)	25.248	16.650	0.4308	-	-
(10, 12)	18.528	8.250	0.2836	7.650	0.2783
(10, 15)	12.624	4.800	0.5445	7.200	0.6010
(11, 14)	12.624	8.400	0.4724	11.400	1.0527
(12, 16)	12.624	6.000	0.7703	5.700	0.7638
(13, 14)	25.248	17.700	0.4724	12.300	0.4172
(14, 17)	18.528	11.400	0.5130	8.700	0.4744
(14, 18)	6.312	2.850	0.9235	2.550	0.9005
(15, 17)	12.624	5.250	0.4361	5.850	0.4481
(15, 20)	6.312	1.200	0.9639	1.200	0.9639
(16, 20)	6.312	2.400	0.6154	3.300	0.6918
(17, 19)	12.624	8.250	0.7876	7.350	0.7486
(18, 19)	12.624	4.500	0.4667	5.700	0.4880
(19, 20)	12.624	4.650	0.7973	4.950	0.8022
		T _{moy} = 1.9094 ms		T _{moy} = 2.5405 ms	

4.5.1 Comportement d'un réseau à haut débit en cas de défaillance d'une liaison

Étudions le comportement de chaque liaison du réseau à haut débit de la Figure 4.2, en étiquetant les liaisons par des numéros selon la correspondance présentée au Tableau 4.7. Ce réseau peut être soit à l'état E_0 lorsqu'il n'y a pas de défaillance, ou à l'état E_f (avec $f = 1, 2, \dots, 30$) quand la liaison f tombe en panne. En outre, toutes les hypothèses de la section 4.2 sont respectées, de sorte que le réseau ne puisse passer que de l'état E_0 à un état E_f ou inversement (après réparation de la liaison f). Et, en passant d'un état à l'autre, le flot de chaque liaison du réseau subit une certaine variation, variation qui dépend de la liaison défaillante. Exprimons cette variation en pourcentage par rapport au flot sans défaillance. Nous obtenons la variation relative de flot causée par la panne de f sur chaque liaison l , donnée par la relation:

$$\Delta f_{lf} (\%) = \frac{\Delta f_{lf}}{f_l} * 100(\%), \quad \forall l \neq f \quad (4.25)$$

où:

- Δf_{lf} représente la variation de flot sur la liaison l causée par la défaillance de f et exprimée par la relation (4.22); et
- f_l le flot de la liaison l à l'état E_0 .

Cette expression caractérise le comportement de chaque liaison l du réseau en fonction de la défaillance. La Figure 4.3 illustre la variation relative de flot sur chacune des trois premières liaisons du réseau en fonction de son état, le comportement des autres liaisons étant illustré en Annexe II (section AII.3).

Tableau 4.7 Numérotation des liaisons

Liaison	No		Liaison	No		Liaison	No
(1, 3)	1		(5, 11)	11		(12, 16)	21
(1, 4)	2		(6, 8)	12		(13, 14)	22
(1, 5)	3		(6, 12)	13		(14, 17)	23
(2, 3)	4		(7, 9)	14		(14, 18)	24
(2, 6)	5		(8, 10)	15		(15, 17)	25
(2, 7)	6		(8, 13)	16		(15, 20)	26
(3, 7)	7		(9, 13)	17		(16, 20)	27
(4, 11)	8		(10, 12)	18		(17, 19)	28
(4, 18)	9		(10, 15)	19		(18, 19)	29
(5, 9)	10		(11, 14)	20		(19, 20)	30

Étant donné un régime transitoire inexistant, le flot de chaque liaison change instantanément d'un état à l'autre, ce qui se traduit par des courbes discrètes, où chaque point représente la variation de flot pour un état considéré du réseau. Toutefois, pour augmenter la clarté des graphiques, nous relierons les différents points de chaque courbe par des segments de droite. Il convient alors d'être prudent dans l'interprétation de ces courbes: les états ne se lisent pas séquentiellement, puisque le réseau ne peut passer d'un état E_f à E_{f+1} . En fait, chaque courbe traduit, pour un état possible du réseau, la variation relative de flot sur une liaison donnée. Par exemple, nous pouvons nous intéresser à la variation de flot sur la liaison 3 lorsque la liaison 22 est défaillante. Nous notons alors une augmentation relative de flot d'environ 21%.

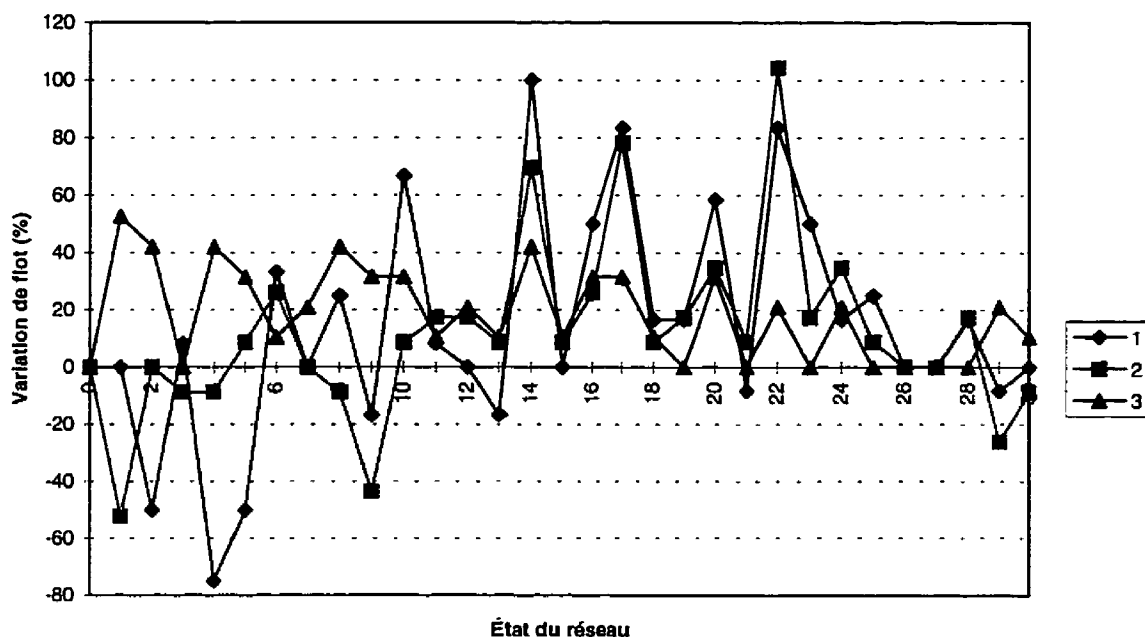


Figure 4.3 Variation relative de flot sur les liaisons 1, 2, 3

Conformément à l'analyse précédente, la variation de flot sur une liaison donnée peut être aussi bien positive que négative, dépendant de l'état du réseau: nous ne pouvons prévoir ou déterminer de façon précise la borne supérieure de cette variation. Cette dernière peut, dans certains cas, doubler ou même tripler la valeur du flot à l'état E_0 , comme d'ailleurs illustré à la Figure AII.8 (en Annexe II, section AII.3), où la défaillance de la liaison 21 augmente le flot de la liaison 26 d'environ 350%. Toutefois, cette variation est inférieurement bornée par la valeur négative du flot, ce qui donne lieu à une diminution maximale relative de flot de 100%. Dans le cas de notre réseau à haut débit, cette diminution atteint 94.7% sur la liaison 4 après défaillance de la liaison 1 (Figure AII.1 de l'Annexe II).

Par ailleurs, certaines liaisons proches de la défaillance ont tendance à subir une variation importante de flot. Par exemple, l'augmentation de flot de la liaison 26 est de 350% lorsque la liaison 21 tombe en panne et celle de la liaison 30 est de 77.4% lorsque la liaison 25 devient défaillante; tandis que l'augmentation de flot sur cette même liaison 30 demeure relativement faible pour toute défaillance des liaisons 1 à 20 (voir les graphiques de l'Annexe II, section AII.3). Cependant, les liaisons proches de la défaillance ne sont pas les seules à subir de variation importante de flot. Ainsi, la défaillance de la liaison 5 donne lieu à une augmentation relative de flot d'environ 125% sur la liaison 26, alors que celle de la liaison 22 augmente le flot de la liaison 2 d'environ 105%. Ces résultats confirment qu'il est difficile de prévoir la variation de flot sur les différentes liaisons à la suite d'une panne.

Il est évident que la variation de flot sur chaque liaison non défaillante donne lieu à un changement de la charge globale du réseau. Étudions alors ce changement, en déterminant la variation moyenne de flot dans le réseau causée par la défaillance d'une liaison f selon la relation:

$$\Delta f_{f_{\text{moy}}} = \frac{1}{m} \sum_{l=1}^m \Delta f_{lf}, \quad l \neq f \quad (4.26)$$

où:

- m représente le nombre total de liaisons du réseau; et
- Δf_{lf} la variation de flot causée par la défaillance de la liaison f sur une liaison l .

En déterminant pour chaque état la variation moyenne de la charge du réseau, nous obtenons la variation de sa charge moyenne en fonction de son état. La Figure 4.4 en

donne une illustration. Nous nous rendons compte qu'une telle variation est toujours positive. Autrement dit, une défaillance donne lieu à une augmentation de la charge globale du réseau. Pour le réseau de la Figure 4.2, l'augmentation moyenne de la charge est maximale lorsque la liaison 23 tombe en panne.

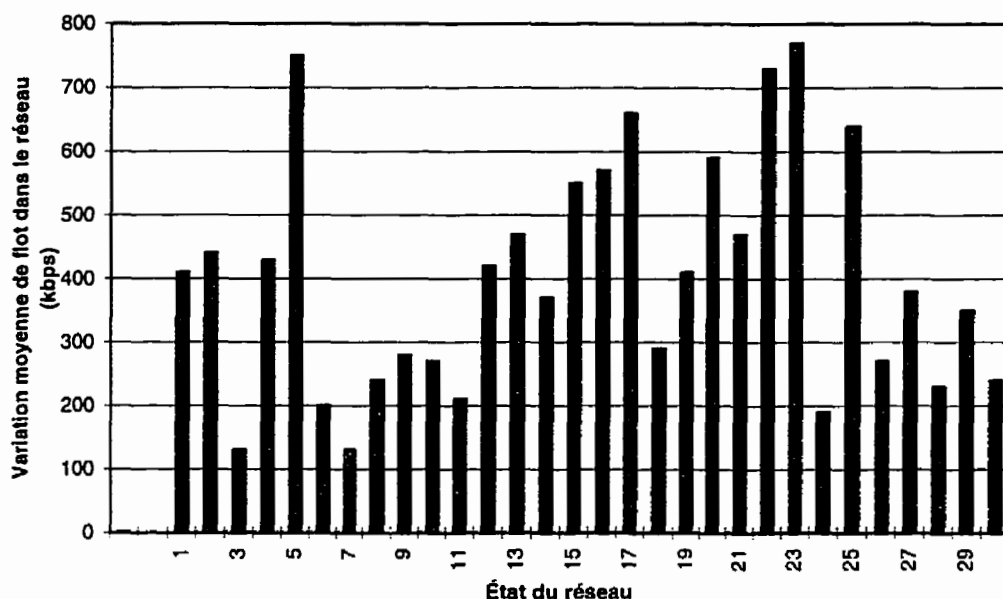


Figure 4.4 Variation moyenne de la charge du réseau

Conformément à la relation (3.25), une variation de flot donne lieu à une variation du temps d'attente des paquets. Autrement dit, l'augmentation de flot sur une liaison conduit à une augmentation du délai total des paquets sur cette liaison et, de ce fait, à l'augmentation du délai moyen de paquet dans le réseau. Aussi devrait-on s'attendre à ce que le délai moyen de chaque paquet suive la même allure que la variation moyenne de la charge du réseau. Nous illustrons à la Figure 4.5 l'évolution du délai

moyen en fonction de l'état du réseau. Effectivement, la défaillance de la liaison 23 causant la variation de flot la plus élevée conduit au délai le plus élevé, alors que la liaison 3 dont la défaillance affecte très peu le comportement du reste du réseau aboutit au délai le plus faible.

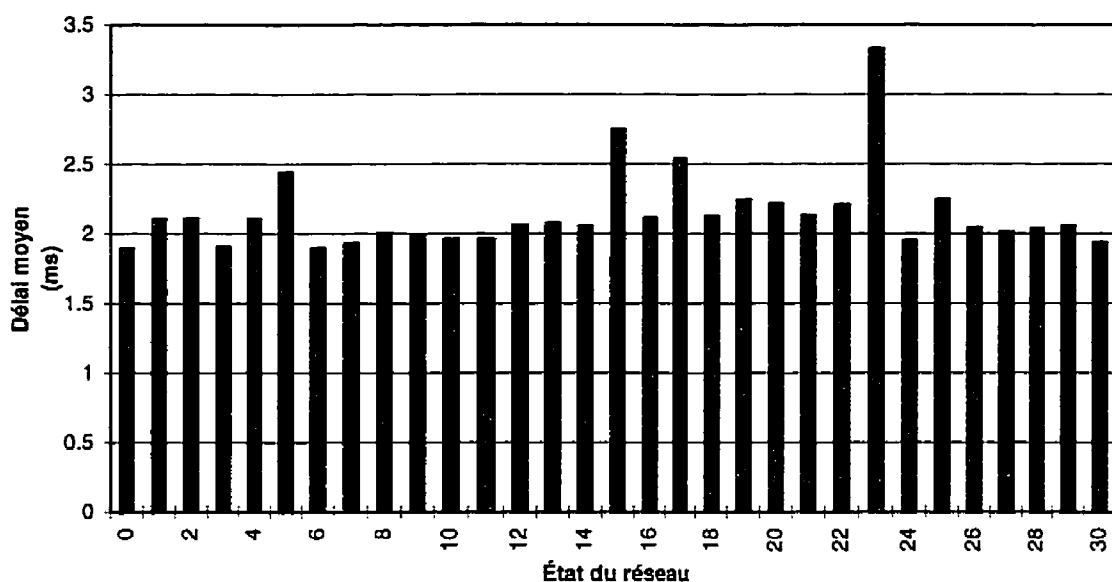


Figure 4.5 Variation du délai moyen de paquet

Cette étude de comportement d'un réseau à haut débit nous a permis de confirmer par les différents graphiques quelques résultats importants: la variation de flot sur chaque liaison a une borne inférieure et la charge moyenne globale du réseau augmente, par suite de la défaillance d'une liaison. Il serait maintenant intéressant de comparer les résultats obtenus de notre méthode à ceux produits par la méthode de Gavish et Neuman (1992).

4.5.2 Comparaison de résultats

Les résultats obtenus, ainsi que la performance de notre méthode, nous ont motivés à l'appliquer aux réseaux GTE et USA, respectivement représentés par les figures 4.6 et 4.7 (Gavish & Neuman, 1992), et à comparer nos résultats à ceux obtenus par l'application de la méthode de Gavish et Neuman (1992). Cette comparaison comprend deux volets: dans un premier temps, nous comparons les délais obtenus par les deux algorithmes de routage avec et sans défaillance de liaison; dans un deuxième temps, en plus de l'affectation de flots, nous utilisons notre méthode pour refaire l'affectation de capacités et comparons les délais résultants, ainsi que les capacités utilisées par les deux méthodes.

Identifions la méthode de survivabilité de Gavish et Neuman par *Méthode 1* et la nôtre par *Méthode 2*. Pour comparer les résultats de ces deux méthodes, utilisons les données suivantes:

- un trafic uniforme de 1 paq/s entre chaque paire de noeuds;
- des paquets de 400 bits, 450 bits et de 500 bits pour le réseau GTE et des paquets de 430 bits pour le réseau USA;
- les capacités indiquées respectivement sur les liaisons des deux réseaux.

Montrons que notre stratégie (qui intègre également l'affectation de capacités) peut résoudre le problème de routage seul, *i. e.* sans faire d'affectation de capacités aux liaisons. Soit Π_k le chemin choisi pour acheminer le trafic γ_k d'une commodité k . Désignons par $f_{l1}^p, f_{l2}^p, \dots, f_{lk}^p$ les flots courants et $C_{l1}, C_{l2}, \dots, C_{lk}$ les capacités

respectives des x liaisons de Π_k . Selon les principes mêmes de notre algorithme (section 3.4.2), Π_k constitue le chemin le plus rapide de la commodité k . Il convient alors d'ajouter γ_k au flot courant f_{li}^P de chaque liaison l_i de Π_k et de comparer ensuite, pour chaque l_i , C_{li} à la capacité provisoire C_{li}^P (voir la section 3.4.2).

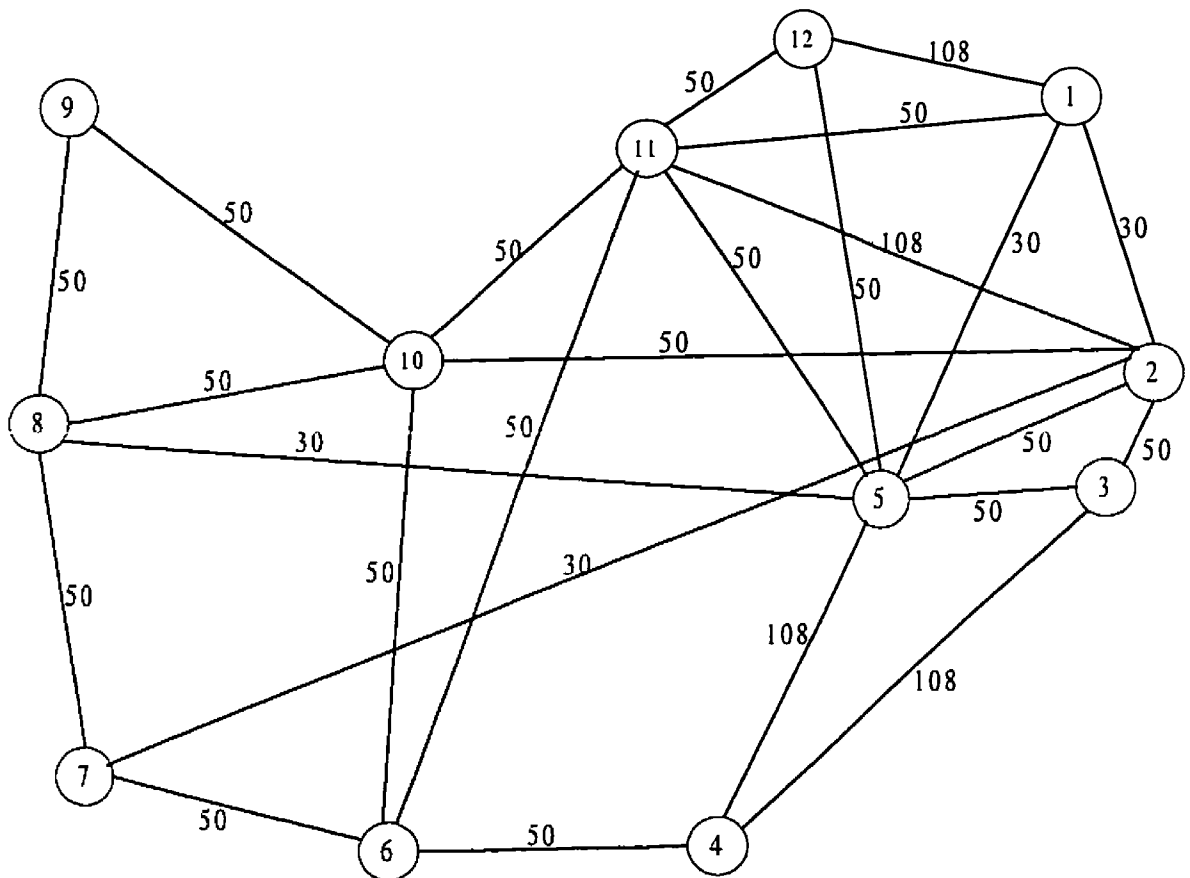


Figure 4.6 Réseau GTE (12 noeuds, 25 liaisons)

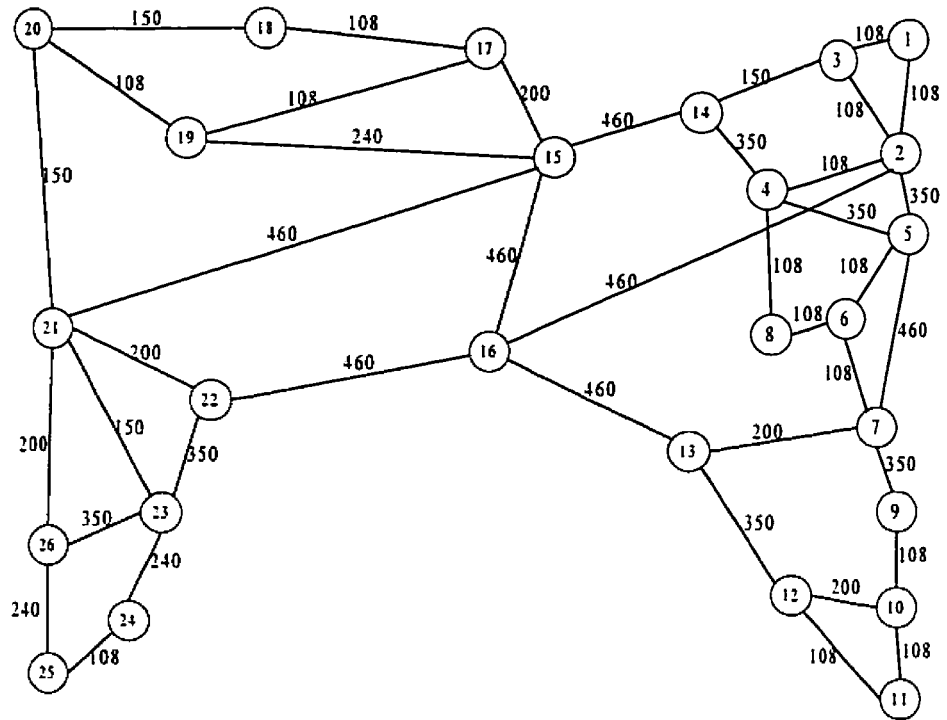


Figure 4.7 Réseau USA (26 noeuds, 41 liaisons)

Cette comparaison peut donner lieu à deux types de résultats. Lorsque $C_{li}^P < C_{li}$ pour toute liaison l_i , nous concluons que les contraintes de capacités sont respectées pour toutes les liaisons de Π_k . S'il en est de même pour toutes les commodités du réseau, il n'y aura pas d'affectation explicite de capacités et, par conséquent, $C_l^P = C_l$ pour toute liaison l du réseau. Par contre, si après l'affectation de γ_k à une liaison l_i de Π_k , on se rend compte que $C_{li}^P \geq C_{li}$, il importe de comparer $(f_{li}^P + \gamma_k)$ et C_{li} . Lorsque $(f_{li}^P + \gamma_k) < C_{li}$, nous concluons que l_i est en mesure d'absorber la totalité du trafic sans risque de congestion. Dans le cas contraire, le chemin Π_k n'est pas disponible et l'on doit acheminer le trafic γ_k

par un autre chemin disponible du réseau. Toutefois, lorsqu'il n'y reste plus de ces chemins, nous concluons que le réseau est sous-dimensionné.

Une telle modification a été intégrée à l'algorithme que nous avons proposé à la section 3.4.2 et a donné lieu à une nouvelle version de l'approche, version que nous appliquons aux réseaux GTE et USA des figures 4.6 et 4.7. À partir de la taille moyenne des paquets, de la capacité et du flot de chaque liaison, nous pouvons déterminer le délai moyen de paquet, sans défaillance. Toutefois, comme illustré à la Figure 4.5, le délai moyen par suite d'une panne dépend de la liaison défaillante, ce qui nous conduit à utiliser la moyenne du délai à l'état E_f . Ainsi, pour différentes tailles de paquets, nous déterminons respectivement pour les deux méthodes le délai moyen sans défaillance et avec défaillance.

Nous présentons aux tableaux 4.8 et 4.9 les résultats comparatifs des deux méthodes. Dans les deux cas, le délai moyen obtenu de la Méthode 2 est meilleur que celui de la Méthode 1. Pour le réseau GTE, l'amélioration de délai est comprise entre 27.3 et 37.2% quand il n'y a pas de défaillance, alors qu'elle se situe entre 26.6 et 37.7% dans le cas de défaillance d'une liaison. Cette amélioration est moindre pour le réseau USA, mais se situe tout de même autour de 20%, ce qui témoigne de l'efficacité de la Méthode 2.

Dans un autre ordre d'idée, appliquons notre algorithme de routage et d'affectation de capacités (CFA) aux configurations de réseaux des figures 4.6 et 4.7. Notons que, dans ce cas, les capacités des liaisons sont inconnues. Ainsi, l'affectation de capacités se réalise à partir des options du Tableau 4.10. Comparons alors, d'une part, les

délais obtenus et, d'autre part, les capacités obtenues de notre stratégie à celles utilisées par la Méthode 1.

Tableau 4.8 Comparaison de délai du réseau GTE (Cas 1)

	Méthode 1*			Méthode 2*			Amélioration (%)		
$1/\mu$ (bits/paq)	400	450	500	400	450	500	400	450	500
T_{moy} sans défaillance (ms)	22.2	27.5	33.1	16.1	18.4	20.8	27.3	33.0	37.2
T_{moy} avec défaillance (ms)	22.9	28.3	35.0	16.8	19.2	21.8	26.6	32.2	37.7

Note: Méthode 1 correspond à la méthode de Gavish et Neuman, et Méthode 2 correspond à la nôtre.

Tableau 4.9 Comparaison de délai du réseau USA pour $1/\mu=430$ bits/paq (Cas 1)

	Méthode 1	Méthode 2	Amélioration (%)
T_{moy} sans défaillance (ms)	11.3	8.8	22.1
T_{moy} avec défaillance (ms)	12.1	9.8	19.0

Nous présentons au Tableau 4.11 les capacités utilisées par la Méthode 1 et celles obtenues de l'affectation effectuée par la Méthode 2. Nous nous rendons compte que, sur la plupart des liaisons, les capacités obtenues de notre méthode sont nettement plus faibles que celles utilisées par la Méthode 1. Dans la mesure où le coût d'une liaison est fonction de sa capacité (voir la section 2.1.6), nous déduisons que les capacités de la

Méthode 2 sont nettement plus économiques que celles utilisées par la Méthode 1, comme en témoigne le calcul de la somme des capacités utilisées par chaque méthode.

Tableau 4.10 Options de capacités des réseaux GTE et USA

C_l (kbps)	30	50	108	150	200	240	350	460
-----------------	----	----	-----	-----	-----	-----	-----	-----

Tableau 4.11 Capacités de liaisons en kbps des réseaux GTE et USA

Liaison	Réseau GTE		Réseau USA	
	Méthode 1	Méthode 2	Méthode 1	Méthode 2
(1, 2)	30	30	108	150
(1, 3)	-	-	108	150
(1, 5)	30	30	-	-
(1, 11)	50	30	-	-
(1, 12)	108	30	-	-
(2, 3)	50	30	108	150
(2, 4)	-	-	108	150
(2, 5)	50	30	350	150
(2, 7)	30	30	-	-
(2, 10)	50	30	-	-
(2, 11)	108	30	-	-
(2, 16)	-	-	460	150
(3, 4)	108	30	-	-
(3, 5)	50	30	-	-
(3, 14)	-	-	150	150
(4, 5)	108	30	350	150
(4, 6)	50	30	-	-
(4, 8)	-	-	108	150
(4, 14)	-	-	350	150
(5, 6)	-	-	108	150
(5, 7)	-	-	460	150
(5, 8)	30	30	-	-
(5, 11)	50	30	-	-

Tableau 4.11 Capacités de liaisons en kbps des réseaux GTE et USA (suite)

Liaison	Réseau GTE		Réseau USA	
	Méthode 1	Méthode 2	Méthode 1	Méthode 2
(5, 12)	50	30	-	-
(6, 7)	50	30	108	150
(6, 8)	-	-	108	150
(6, 10)	50	30	-	-
(6, 11)	50	30	-	-
(7, 8)	50	30	-	-
(7, 9)	-	-	350	150
(7, 13)	-	-	200	150
(8, 9)	50	30	-	-
(8, 10)	50	30	-	-
(9, 10)	50	30	108	150
(10, 11)	50	30	108	150
(10, 12)	-	-	200	150
(11, 12)	50	30	108	150
(12, 13)	-	-	350	200
(13, 16)	-	-	460	200
(14, 15)	-	-	460	200
(15, 16)	-	-	460	150
(15, 17)	-	-	200	150
(15, 19)	-	-	240	150
(15, 21)	-	-	460	150
(16, 22)	-	-	460	200
(17, 18)	-	-	108	150
(17, 19)	-	-	108	150
(18, 20)	-	-	150	150
(19, 20)	-	-	108	150
(20, 21)	-	-	150	150
(21, 22)	-	-	150	150
(21, 23)	-	-	150	150
(21, 26)	-	-	200	150
(22, 23)	-	-	350	150
(23, 24)	-	-	240	150
(23, 26)	-	-	350	150
(24, 25)	-	-	108	150
(25, 26)	-	-	240	150
Capacité totale (kbps)	1402	750	9560	6350

Les résultats comparatifs du délai résultant des deux méthodes sont présentés aux tableaux 4.12 et 4.13. En plus du délai moyen, nous déterminons l'augmentation relative du délai avec défaillance par rapport à celui sans défaillance. De façon générale, le délai moyen avec ou sans défaillance varie dans le même sens que la taille moyenne des paquets. Par exemple, pour le réseau GTE, l'utilisation de la Méthode 1 conduit à un délai sans défaillance de 22.2 ms lorsque la taille des paquets est de 400 bits, et ce délai passe respectivement à 27.5 et 33.1 ms lorsque la taille devient 450 et 500 bits. Ceci peut aisément s'expliquer par la relation (2.5), où le délai sur chaque liaison est proportionnel à la taille moyenne ($1/\mu$) des paquets. De ce fait, le délai moyen des paquets augmente dans le même sens que leur taille.

Tableau 4.12 Comparaison de délai du réseau GTE (Cas 2)

	Méthode 1			Méthode 2		
$1/\mu$ (bits/paq)	400	450	500	400	450	500
T_{moy} sans défaillance (ms)	22.2	27.5	33.1	26.1	29.9	33.9
T_{moy} avec défaillance (ms)	22.9	28.3	35.0	27.2	31.3	34.6
ΔT_{moy} (%)	3.1	2.8	4.4	4.0	4.4	4.8

En outre, pour les deux réseaux, en dépit de l'affectation de capacités plus faibles de notre méthode, les résultats de cette dernière restent comparables à ceux de la

Méthode 1 (Gavish & Neuman, 1992) pour les tailles de paquets considérées. Par exemple, pour des paquets de 500 bits, l'utilisation de la Méthode 1 donne lieu à un délai sans défaillance de 33.1 ms et un délai avec défaillance de 35.0 ms, soit une augmentation de 4.4%, alors que celle de notre méthode conduit à un délai sans défaillance de 33.9 ms et un délai moyen de 34.6 ms avec défaillance pour une augmentation de 4.8% (en considérant le réseau GTE).

Tableau 4.13 Comparaison de délai du réseau USA pour $1/\mu=430$ bits/paq (Cas 2)

	Méthode 1	Méthode 2
$T_{\text{moy sans}} \text{ défaillance (ms)}$	11.3	12.2
$T_{\text{moy avec}} \text{ défaillance (ms)}$	12.1	12.9
$\Delta T_{\text{moy}} (\%)$	6.6	4.4

Ainsi, lorsque les deux méthodes utilisent les mêmes capacités, notre méthode produit de meilleurs résultats que la méthode de Gavish et Neuman (1992). L'amélioration du délai moyen de paquet varie alors entre 19.0 et 37.7%. Par ailleurs, lorsque notre méthode affecte des capacités plus économiques aux liaisons, nous nous rendons compte que, dans la plupart des cas, le délai moyen obtenu d'une telle méthode est comparable à celui de la Méthode 1. Pour le réseau GTE, l'augmentation moyenne de délai par rapport à celui de la Méthode 1 avec ou sans défaillance est de l'ordre de 9%, alors qu'elle n'est que de 7% pour le réseau USA. Ces résultats sont satisfaisants dans la

mesure où la Méthode 1 constitue une méthode exacte qui décompose le problème en $\{m + n(n-1)/2\}$ composantes et qui utilise le meilleur lagrangien pour trouver la solution réalisable la plus proche de l'optimum en un certain nombre d'itérations compris entre 50 et 70 (Gavish & Neuman, 1992); alors que notre méthode constitue une heuristique qui s'exécute en une seule itération et qui, intrinsèquement, s'avère beaucoup plus rapide. Ainsi, l'affectation de capacités effectuée par notre méthode prouve que les réseaux GTE et USA des figures 4.6 et 4.7 ont été surdimensionnés.

Ce chapitre nous a donc permis de nous rendre compte de la complexité du problème d'intégration des aspects de routage et de survivabilité en conception topologique. Nous sommes toutefois parvenus à le simplifier considérablement en déterminant pour chaque paire de noeuds un chemin principal et un chemin de secours, les deux étant disjoints de liaisons. Notre méthode a aussi permis de calculer les capacités permettant au réseau de survivre à une défaillance quelconque, sans dégradation appréciable de qualité de service. La mise en oeuvre et l'analyse des résultats obtenus d'un réseau à haut débit témoigne de l'efficacité d'une telle méthode. Son application sur les réseaux GTE et USA donne lieu en général à des résultats comparables, sinon meilleurs à ceux obtenus par la méthode de Gavish et Neuman (1992).

CHAPITRE V

CONCLUSION

Après avoir présenté dans son ensemble le problème de conception topologique des réseaux distribués, nous avons élaboré sur le routage qui constitue à la fois un aspect clé et un sous-problème important du problème général. Il nous revient maintenant de synthétiser les divers résultats et de proposer quelques orientations de recherches.

5.1 Contributions et synthèse des résultats

Tout d'abord, nous avons élaboré une nouvelle stratégie de routage axée sur certains attributs de liaisons et basée fondamentalement sur deux principes, qui consistent à imposer un seuil au délai de chaque liaison et à acheminer les paquets selon les chemins les plus rapides. Cette stratégie comprend essentiellement trois étapes: la première achemine les paquets des paires de noeuds adjacents et permet de déterminer les attributs initiaux de liaisons, la seconde achemine selon les chemins les plus rapides les paquets des autres commodités et s'occupe de mettre à jour les attributs de liaisons impliquées dans de tels chemins. Quant à la dernière étape, elle utilise les données de l'étape précédente pour déterminer le délai moyen des paquets. La complexité de l'ensemble de la stratégie a été évaluée et se compare bien à celle des algorithmes de routage existants.

L'analyse des résultats révèle que, pour un réseau de 10 noeuds et 14 liaisons, et pour différents niveaux de trafic, le délai moyen calculé selon notre approche est, dans la plupart des cas, meilleur que celui déterminé par des heuristiques de routage statique, telles le routage de plus courte distance ou à nombre minimal de hops. De tels résultats nous ont alors motivés à comparer notre approche aux algorithmes classiques de routage optimal tels la méthode de déviation de flot (FD) et l'algorithme de Bertsekas-Gallager (BG). Effectivement, nous nous sommes rendus compte que, pour un temps d'exécution 50 fois plus faible que la méthode FD et plus de 100 fois plus faible que l'algorithme BG, les résultats obtenus de notre stratégie sont relativement proches de ceux produits par ces deux algorithmes de routage optimal. Cela nous a alors convaincus d'appliquer une telle stratégie de routage aux réseaux à haut débit.

Par ailleurs, pour être plus réaliste, nous avons considéré la défaillance possible d'une liaison du réseau pendant le routage, défaillance qui nécessite le reroutage de tous les paquets devant transiter par les chemins utilisant cette liaison. Cela nous a amené à intégrer dans notre stratégie une nouvelle méthode de survivabilité basée sur le concept de chemins disjoints de liaisons et qui garantit la disponibilité d'au moins un chemin après la défaillance aléatoire d'une liaison. Une telle méthode intègre entre autres une stratégie d'affectation de capacités qui s'assure que, par suite d'une panne, les liaisons non défaillantes soient en mesure d'absorber tout le trafic du réseau.

En plus de faciliter la détermination de la variation de flot causée par la panne en question sur les liaisons non défaillantes, notre méthode d'intégration de routage et de survivabilité a aussi permis d'étudier le comportement de chacune de ces liaisons après la

défaillance. Son application sur les réseaux GTE et USA a donné lieu à des résultats comparables, sinon meilleurs que ceux obtenus par la méthode de Gavish et Neuman (1992).

5.2 Limitations des travaux et orientations de recherches

Même si dans l'ensemble les résultats se sont révélés satisfaisants, nous n'avons considéré dans les diverses mises en oeuvre que des réseaux dont le trafic est uniforme. Ceci peut toutefois se justifier par le fait que la plupart des chercheurs (Kershenbaum, 1993; Gavish & Neuman, 1992) utilisent des patrons de trafic uniforme. Ainsi, pour fins de comparaison avec diverses méthodes existantes, nous avons aussi opté pour un trafic uniforme. Il serait par contre intéressant de vérifier la performance de nos algorithmes sur des réseaux dont le trafic entre les paires de noeuds est aléatoire.

D'autre part, l'étude du comportement des liaisons de haut débit après une défaillance nous a permis d'aboutir à des résultats encourageants. Il serait donc opportun de savoir dans quelle mesure de tels résultats peuvent être généralisés. Certains de ces résultats concernant la variation globale de la charge du réseau et la limite inférieure de la variation de flot sur une liaison non défaillante peuvent toutefois être généralisés. Pour les autres, il faudrait tester d'autres réseaux avant d'en conclure sur la généralisation.

Par ailleurs, même s'il existe deux chemins disjoints de liaisons entre une paire de noeuds, il est toutefois possible de ne pas pouvoir déterminer le chemin de secours, si le chemin principal est mal choisi. Par exemple, pour le réseau USA de la Figure 4.7, il existe deux chemins disjoints de liaisons entre les noeuds 1 et 14. Ces chemins sont

respectivement formés des noeuds 1, 3, 14 et 1, 2, 4, 14. Cependant, rien n'empêche de choisir le chemin 1, 2, 3 et 14 comme chemin principal. Si tel est le cas, quel serait alors le chemin de secours de sorte que les deux chemins soient disjoints de liaisons? Il importe donc d'ajouter à la méthode certaines règles qui génèrent les deux chemins disjoints de liaisons de façon à éviter de telles situations. L'élaboration d'une série de règles qui pourraient dicter les chemins les plus fiables à utiliser pour l'acheminement des paquets constitue finalement une avenue de recherche intéressante à explorer.

BIBLIOGRAPHIE

AGGARWAL, K. K., GUPTA, J. S. et MISRA, K. B. (1975). A Simple Method for Reliability Evaluation of a Communication System. IEEE Trans. on Communications, 23, 563-566.

BARANSEL, C., DOBOSIEWICZ, W. et GBURZYNSKI, P. (1995). Routing in Multihop Packet Switching Networks: Gb/s Challenge. IEEE Network, 9, 38-61.

BERTSEKAS, D. et GALLAGER, R. (1987). Data Networks, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 297-407.

BOORSTYN, R. et FRANK, H. (1977). Large-Scale Network Topological Optimization. IEEE Trans. on Communications, 25, 29-47.

CAREY, M. et HENDRICKSON, C. (1984). Bounds on Expected Performance of Networks with Links Subject to Failure. Networks, 14, 439-456.

CHANG, S.-G. (1992). Fair Integration of Routing and Flow Control in Communication Networks. IEEE Trans. on Communications, 40, 821-834.

CHEN, H. et ZHOU, J. (1991). Reliability Optimization in Generalized Stochastic-Flow Networks. IEEE Transactions on Reliability, 40, 92-97.

CHATTOPADHYAY, N. G., MORGAN, T. W. et RAGHURAM, A. (1989). An Innovative Technique for Backbone Network Design. IEEE Trans. on Systems. Man and Cybernetics, 19, 1122-1132.

CIDON, I., GOPAL, I. et KUTTEN, S. (1995). New Models and Algorithms for Future Networks. IEEE Trans. on Information Theory, 41, 769-780.

COLBOURN, C. J. (1991). Combinatorial Aspects of Network Reliability. Annals of Operations Research, 33, 3-15.

COURTOIS, P. J. et SEMAL, P. (1980). A Flow Deviation Algorithm Based on the Flow Deviation Method. Philips Res. Lab. Brussels, 77-81.

DAVIES, D. W., BARBER, D. L. A., PRICE, W. L. et SOLOMONIDES, C. M. (1979). Computer Networks and their Protocols, John Wiley, New York, 417-455.

DUTTA, A. et MITRA, S. (1993). Integrating Heuristic Knowledge and Optimization Models for Communication Network Design. IEEE Trans. on Knowledge and Data Engineering, 5, 999-1017.

EVANS, J. R. et MINIEKA, E. (1992). Optimization Algorithms for Networks and Graphs, Marcel Dekker Inc., New York, 27-122.

FRANK, H. et CHOU, W. (1971). Routing in Computer Network. Networks, 1, 9-112.

FRATTA, L., GERLA, M. et KLEINROCK, L. (1974). The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design. Networks, 3, 97-33.

GAVISH, B. (1992). Topological Design of Computer Networks-The Overall Design Problem. European Journal of Operational Research, 58, 149-172.

GAVISH, B. et NEUMAN, I. (1989). A System for Routing and Capacity Assignment in Computer Communication Networks. IEEE Trans. on Communications, 37, 360-366.

GAVISH, B. et NEUMAN, I. (1992). Routing in a Network with Unreliable Components. IEEE Trans. on Communications, 40, 1248-1258.

GELENBE, E. et PUJOLLE, G. (1982). Introduction aux réseaux de files d'attente, Eyrolles, Paris, 1-112.

GERLA, M. (1973). The Design of Store-and-forward Networks for Computer Communications, Ph. D. dissertation, School of Eng. and Appl. Sci., UCLA, Los Angeles.

GERLA, M. (1975). Approximations and Bounds for the Topological Design of Distributed Computer Networks. Proc. of Fourth Data Communications Symposium, Québec City, 4.9-4.15.

GERLA, M., FRANK, H., CHOU, W. et ECKL, J. (1974). A Cut Saturation Algorithm for Topological Design of Packet-Switched Networks. Proc. of Nat. Telecomm. Conf., San Diego, 1074-1085.

GERLA, M. et KLEINROCK, L. (1977). On the Topological Design of Distributed Computer Networks. IEEE Trans. on Communications, 25, 48-60.

GIRARD, A. (1990). Routing and Dimensioning in Circuit-Switched Networks, Addison-Wesley, Reading, Massachusetts.

HYPPOLITE, M.-A. (1994). Optimisation de topologies de réseaux téléinformatiques à commutation de paquets par la méthode de recuit simulé, Mémoire de maîtrise, Université du Québec à Montréal, Canada.

HWANG, R.-H., KUROSE, J. F. et TOWSLEY, D. (1995). On-Call Processing Delay in High Speed Networks. IEEE/ACM Transactions on Networking, 3, 628-639.

JAN, R.-H., HWANG, F.-J. et CHENG, S.-T. (1993). Topological Optimization of a Communication Network Subject to a Reliability Constraint. IEEE Transactions on Reliability, 42, 63-70.

KERSHENBAUM, A. (1993). Telecommunications Network Design Algorithms, McGraw-Hill Inc., New York.

KERSHENBAUM, A., KERMANI, P. et GROVER, G.A. (1991). MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing. IEEE Trans. on Comm., 39, 503-513.

KERSHENBAUM, A. et RUBINSON, T. (1991). An Application of Fuzzy Multi-Criteria Decision Making for Topological Design of Large Networks. Third International Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Polytech. Univ., Brooklyn, NY, 528-537.

KHASNABISH, B. (1993). A New Method for Evaluating Packet Routing Policies in Supra-High-Speed Metropolitan (or wide) Area Networks. Computer Networks and ISDN Systems, 195-216.

KLEINROCK, L. (1975). Queueing Systems Vol. I: Theory, John Wiley & Sons, New York.

KLEINROCK, L. (1976). Queueing Systems Vol. II: Computer Applications, John Wiley & Sons, New York, 270-421.

KLEINROCK, L. (1992). The Latency/Bandwidth Tradeoff in Gigabit Networks. IEEE Communications Magazine, 30, 36-40.

KUBAT, P. (1991). Alternatives for Reliable and Quality Telecommunication Network Design. Annals of Operations Research, 33, 95-105.

LEGAULT, G. (1994). Application de l'algorithme génétique à la conception topologique de réseaux téléinformatiques à commutation de paquets, Mémoire de maîtrise, Université du Québec à Montréal, Canada.

MOLLOY, M. K. (1989). Fundamentals of Performance Modeling, Mc. Millan, New York.

NEWPORT, K. T. et VARSHNEY, P. K. (1989). On the Design of Performance-Constrained Survivable Networks. The MITRE Corporation and Syracuse University, 663-670.

NEWPORT, K. T. et VARSHNEY, P. K. (1991). Design of Survivable Communications Networks under Performance Constraints. IEEE Transactions on Reliability, 40, 433-440.

PIERRE, S. (1991). Approche d'Intelligence Artificielle pour l'Amélioration de Topologies de Réseaux Téléinformatiques, Thèse de doctorat, École Polytechnique de Montréal, Canada.

PIERRE, S. (1994). A Learning-by-Example Method for Improving Performance of Network Topologies. Engng Applic. Artif. Intel., 7, 279-288.

PIERRE, S. (1995). Réseaux locaux: fondements, implantation et études de cas, Télé-Université, Sainte-Foy, Québec.

PIERRE, S., HYPPOLITE, M.-A., BOURJOLLY, J.-M. et DIOUME O. (1995). Topological Design of Computer Communication Networks Using Simulated Annealing. Engng Applic. Artif. Intel., 8, 61-69.

RAHMAN, M. et Latif, M. (1989). Topological Design of a Distributed Computer Communication Network in a Noisy Environment. Computer Communications, 12, 331-336.

- RUDNIAŃSKI, M. (1986). Architecture de réseaux: le modèle OSI, Éditest, Paris, 31-46.
- SAKSENA, V. R. (1989). Topological Analysis of Packet Networks. IEEE Journal on Selected Areas in Communications, 7, 1243-1252.
- SANSO, B. (1991). Communication & Transportation Network Reliability Using Routing Models. IEEE Trans. on Reliability, 40, 29-38.
- SAPIR, D. L. (1978). A Generalized Cut-saturation Algorithm for Distributed Computer Communications Network Optimization, M. S. Thesis, Computer Studies Program. North Carolina State University, Raleigh, NC.
- SCHUMACHER, U. (1984). An Algorithm for Construction for a k -connected Graph with Minimum Number of Edges and Quasiminimal Diameter. Networks, 14, 63-74.
- SCHWARTZ, M. (1987). Telecommunication Networks: Protocols. Modeling and Analysis, Addison-Wesley, Reading, Massachusetts.
- SCHWARTZ, M. et Stern, T. E. (1980). Routing Techniques Used in Computer Communication Networks. IEEE Trans. on Communications, 28, 539-552.

STAN, K-B. (1988). Turbo C par la pratique, Sybex, Paris.

TANENBAUM, A. (1988). Computer Networks, Prentice-Hall, Englewood Cliffs, New Jersey, 32-81.

TOMY, M. et HOANG, D. B. (1987). Joint Optimization of Capacity and Flow Assignment in a Packet-Switched Communications Network. IEEE Trans. on Communications, 35, 202-209.

VETTER, R. J. et DU, D. H. C. (1993). Distributed Computing with High-Speed Optical Networks. Computer, 26, 8-18.

WANG, C. et SCHWARTZ, M. (1993). Identification of Faulty Links in Dynamic-Routed Networks. IEEE Journal on Selected Areas in Communications, 11, 1449-1460.

YOKOHIRA, T., SUGANO, M., NISHIDA, T. et MIYAHARA, H. (1991). Fault Tolerant Packet-Switched Network Design and its Sensitivity. IEEE Trans. on Reliability, 40, 452-460.

ZANAKIS, S. H. et EVANS, J. R. (1981). Heuristic Optimization: Where, when and how to use it. Interface, 11, 84-90.

ZAYCHENKO, Y. P. et GNATOVSKIY, V. N. (1987). Topological Design of Computer Networks on the Basis of a Combination of Analytical-Simulation Models. Soviet Journal of Automation and Information Science, 20, 55-62.

ANNEXE I

RÉSULTATS DÉTAILLÉS DU CHAPITRE III

Nous présentons ici les résultats détaillés du chapitre III. Nous exposerons d'abord les principales fonctions de notre stratégie de routage, ensuite certains résultats détaillés obtenus avec les heuristiques, puis avec les algorithmes de routage optimal.

AI.1 Principales fonctions de la nouvelle stratégie de routage

Nous présentons les principaux modules de notre algorithme de routage. La structure de liaisons définie par ListeLiaison contient tous les attributs de liaisons utilisés pour acheminer les paquets selon les chemins les plus rapides. Les macros TraficMoyen, BitParPaquet et MaxNoeud désignent respectivement le trafic moyen entre chaque paire de noeuds, la taille moyenne des paquets et le nombre maximal de noeuds que doit préciser l'utilisateur si ce dernier désire utiliser son propre réseau. Une fois les structures de données définies, il importe de faire l'allocation dynamique de mémoire pour créer dynamiquement les noeuds et liaisons du réseau. Pour le reste, nous suivons les différentes étapes de l'algorithme, en fusionnant toutefois les étapes 2 et 3.

```

/*****
/* Structures de données de la nouvelle stratégie de routage */
*****/

#define TraficMoyen 5000 /* trafic considere entre chaque paire de noeuds */
#define BitParPaquet 1000 /* taille moyenne d'un paquet */

```

```

#define Infini      99000001L
#define MaxNoeud    28

typedef struct ListeLiaison {
    unsigned char No, No1, No2; /* numero et noeuds incidents a la liaison*/
    double Dist, CoutLiaison; /* longueur, flot et capacite de la liaison */
    float Ti;
    long capa, reste, trafic;
    struct ListeLiaison *Prec, *Suivant;
} ListeLiaison;

typedef struct ListeNoeud {
    unsigned char No, X, Y;
    double dist;
    ListeLiaison *lien;
    struct ListeNoeud *chemin;
    struct ListeNoeud *Prec, *Suivant;
} ListeNoeud;

Static ListeNoeud *tas1[MaxNoeud];
Static unsigned char nodepart, noarrive;
Static ListeNoeud *NDebut, *NFin, *Nnouveau, *NTemp, *No1, *No2, *NCase;
Static ListeLiaison *LDebut, *LFin, *LNouveau, *LTemp, *LCase, *Chemin;
Static uchar NbreNoeud, NbreLiaison;
Static double DistanceTotal, MaxLiaison, CoutTotal;
Static float delai, DelaiChemin;
/*****

/*****
/* NewApproach: calcule le routage des paquets selon une nouvelle technique*/
Static Void NewApproach()
{
    Etape1();
    Etape2();
    Tas(nodepart);
    PccDelai();
    afficherchemin();
}

/*****
/*InitNoeud: Cette procedure initialise les coordonnees des noeuds*/
Static Void InitNoeud()
{
    long FORLIM;

    NDebut = NULL;
    NFin = NULL;

```

```

FORLIM = NbreNoeud;
for (I = 1; I <= FORLIM; I++) {
    NCase = (ListeNoeud *)malloc(sizeof(ListeNoeud));
    NCase->No = I;
    NCase->X = 0;
    NCase->Y = 0;
    if (NDebut == NULL)
        NDebut = NCase;
    else
        NFin->Suivant = NCase;
    NCase->Prec = NFin;
    NFin = NCase;
    NFin->Suivant = NULL;
} /* fin de la boucle for */
} /* fin de la fonction InitNoeud */

/*****
/*InitLiaison: Cette fonction initialise les liaisons */
Static Void InitLiaison()
{
    long FORLIM;

    LDebut = NULL;
    LFin = NULL;
    FORLIM = NbreLiaison;
    for (I = 1; I <= FORLIM; I++) {
        LCase = (ListeLiaison *)malloc(sizeof(ListeLiaison));
        LCase->No = I;
        LCase->No1 = 0;
        LCase->No2 = 0;
        LCase->Dist = 0.0;
        if (LDebut == NULL)
            LDebut = LCase;
        else
            LFin->Suivant = LCase;
        LCase->Prec = LFin;
        LFin = LCase;
        LFin->Suivant = NULL;
    }
}

Static Void Etape1()
{
    ListeLiaison *Lien;
    ListeNoeud *Noeud;
    float caparequis;
    uchar i, j, no, FORLIM;

```

```

delai = 0.0;
Lien = LDebut;

while (Lien != NULL) {
    Lien->trafic = 2*TraficMoyen;
    Lien->Ti = 600.0/(float) TraficMoyen; /* seuil de délai */
    caparequis = (float) (BitParPaquet)/Lien->Ti + (float) (Lien->trafic);
    /* affectation de capacités */
    if (caparequis < 9600)
        Lien->capa = 9600;
    else if (caparequis < 19200)
        Lien->capa = 19200;
    else if (caparequis < 50000L)
        Lien->capa = 50000L;
    else if (caparequis < 100000L)
        Lien->capa = 100000L;
    else if (caparequis < 230400L)
        Lien->capa = 230400L;
    else if (caparequis < 460800L)
        Lien->capa = 460800L;
    else if (caparequis < 921600L)
        Lien->capa = 921600L;
    else if (caparequis < 1843200L)
        Lien->capa = 1843200L;

    Lien->reste = Lien->capa - Lien->trafic; /* capacités résiduelles */
    Lien->Ti = (float) (BitParPaquet)/(float) (Lien->reste); /* délai de la liaison */
    Lien = Lien->Suivant;
}
} /* fin de la fonction Etape1() */

```

```

Static Void Etape2()
{
    ListeNoeud *ptr, *ptr2;
    ListeLiaison *lien;
    uchar no, FORLIM;
    float caparequis;

    FORLIM = NbreNoeud;

    for (no = 1; no <= FORLIM; no++) {
        Tas(no);
        PccDelai();
        comparerDelai(no);
    }

    delai = 0.0;
    lien = LDebut;
    while (lien != NULL) {

```

```

    lien->Ti = 600.0/(float) TraficMoyen;
    caparequis = (float) (BitParPaquet)/lien->Ti + (float) (lien->trafic);

    if (caparequis < 9600)
        lien->capa = 9600;
    else if (caparequis < 19200)
        lien->capa = 19200;
    else if (caparequis < 50000L)
        lien->capa = 50000L;
    else if (caparequis < 100000L)
        lien->capa = 100000L;
    else if (caparequis < 230400L)
        lien->capa = 230400L;
    else if (caparequis < 460800L)
        lien->capa = 460800L;
    else if (caparequis < 921600L)
        lien->capa = 921600L;
    else if (caparequis < 1843200L)
        lien->capa = 1843200L;

    lien->reste = lien->capa - lien->trafic;
    lien->Ti = (float) (BitParPaquet)/(float) (lien->reste);
    delai = delai + (float) (lien->trafic)/(float) (lien->reste);
    lien = lien->Suivant;
}
} /* fin de la fonction Etape2() */

/*****
/* PccDelai: Trouve le chemin au plus court delai entre chaque paire de noeuds*/
Static Void PccDelai()
{
    ListeNoeud *tpn;
    ListeLiaison *tpl;
    long i, j;

    for (i = NbreNoeud; i >= 1; i--) {
        tpn = tasl[0];
        tpl = LDebut;
        while (tpl != NULL) {
            if (tpl->No1 == tpn->No1 || tpl->No2 == tpn->No1) {
                for (j = 1; j < i; j++) {
                    if (tasl[j]->No == tpl->No1 || tasl[j]->No == tpl->No2) {
                        if (tasl[j]->dist > tpn->dist + tpl->Ti) {
                            tasl[j]->dist = tpn->dist + tpl->Ti;
                            tasl[j]->chemin = tpn;
                            tasl[j]->lien = tpl;
                            Tasascendant(j + 1);
                        }
                    }
                }
            }
            tpl = tpl->Suivant;
        }
        tpn = tpn->Suivant;
    }
}

```

```

    }
    }
    }
    tpl = tpl->Suivant;
}
tasl[0] = tasl[i - 1];
Tasdescendant(1L, i);
}
}

```

```

/*****

```

```

/* comparerDelai: calcule le trafic de chaque Liaison */

```

```

Static Void comparerDelai(uchar no)

```

```

{
    ListeNoeud *ptr, *ptr2;
    ListeLiaison *Liaison;

```

```

    float caparequis;

```

```

    ptr = NDebut;

```

```

    while (ptr != NULL) {

```

```

        if (ptr->No != no && ptr->chemin->No != no) {

```

```

            ptr2 = ptr;

```

```

            while (ptr2->No != no) {

```

```

                ptr2->lien->trafic += TraficMoyen; /* Acheminement des paquets sur chaque

```

```

liaison du chemin */

```

```

                ptr2->lien->Ti = 600.0/(float)TraficMoyen; /* seuil de délai */

```

```

                caparequis = (float) (TraficMoyen)/ptr2->lien->Ti + ptr2->lien->trafic; /*

```

```

capacité minimale pour respecter seuil de délai */

```

```

                if (caparequis < 9600)

```

```

                    ptr2->lien->capa = 9600;

```

```

                else if (caparequis < 19200)

```

```

                    ptr2->lien->capa = 19200;

```

```

                else if (caparequis < 50000L)

```

```

                    ptr2->lien->capa = 50000L;

```

```

                else if (caparequis < 100000L)

```

```

                    ptr2->lien->capa = 100000L;

```

```

                else if (caparequis < 230400L)

```

```

                    ptr2->lien->capa = 230400L;

```

```

                else if (caparequis < 460800L)

```

```

                    ptr2->lien->capa = 460800L;

```

```

                else if (caparequis < 921600L)

```

```

                    ptr2->lien->capa = 921600L;

```

```

                else if (caparequis < 1843200L)

```

```

                    ptr2->lien->capa = 1843200L;

```

```

                ptr2->lien->reste = ptr2->lien->capa - ptr2->lien->trafic; /* capacité résiduelle

```

```

de la liaison du chemin */

```

```

                ptr2->lien->Ti = (float) (TraficMoyen) / (float) (ptr2->lien->reste); /* délai

```

```

d'attente et de transmission de la liaison */

```

```

                ptr2 = ptr2->chemin;

```

```

    }
    } /* fin du premier if */
    ptr = ptr->Suivant;
    } /* fin du premier while */
}

```

```

/*****/
/* Tas: Entasser les noeuds à partir du noeud source et initialiser la distance de tous les autres */
/*      noeuds à l'infini */
Static Void Tas(uchar no)
{
    ListeNoeud *ptr;
    long i, n, FORLIM;

    ptr = NDebut;
    n = 2;
    FORLIM = NbreNoeud;
    for (i = 1; i <= FORLIM; i++) {
        if (ptr->No != no) {
            tasl[n - 1] = ptr;
            n++;
            ptr->dist = Infini;
        } else {
            tasl[0] = ptr;
            ptr->dist = 0.0;
        }
        ptr = ptr->Suivant;
    }
}

```

```

/*****/
/* Tasascendant: remonte le Tas minimum avec le noeud */
/*      de la position i a remplacer */
Static Void Tasascendant(i)
long i;
{
    ListeNoeud *tmpn;
    long j;

    while (j > 1) {
        j = (long)(i / 2.0);
        if (tasl[i - 1]->dist >= tasl[j - 1]->dist) {
            j = 0;
            break;
        }
        tmpn = tasl[i - 1];
        tasl[i - 1] = tasl[j - 1];
        tasl[j - 1] = tmpn;
    }
}

```

```

    i = j;
}
}

```

```

/*****
/* Tasdescendant: reforme le tas -heap - minimum par decente de la valeur i          */
/*           pour déterminer le noeud le plus proche du noeud source                */
Static Void Tasdescendant(i, n)
long i, n;
{
    ListeNoeud *tmpn;
    long j;

    tmpn = tasl[i - 1];
    j = i;
    while (j <= n / 2.0) {
        j = i + i;
        if (j < n && tasl[j - 1]->dist > tasl[j]->dist)
            j++;
        if (tmpn->dist > tasl[j - 1]->dist) {
            tmpn = tasl[i - 1];
            tasl[i - 1] = tasl[j - 1];
            tasl[j - 1] = tmpn;
        }
        i = j;
    }
}

```

AI.2 Résultats comparatifs des heuristiques

Nous présentons pour plusieurs niveaux de trafic le chemin emprunté par les paquets de la commodité (1, 9), ainsi que le temps d'attente et de transmission de chaque paquet le long d'un tel chemin. Le réseau utilisé est celui de la Figure 3.6 et la taille moyenne des paquets est fixée à 1000 bits. Ces résultats proviennent des trois heuristiques étudiées au CHAPITRE III.

Tableau AI.1 Résultats comparatifs de routage pour un trafic de 10 paq/s

Heuristiques de routage	T_{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	59.9	1-9	1-5-7-9	119.4
Nombre minimal de hops	38.6	1-9	1-5-7-9	43.8
Nouvelle approche	32.4	1-9	1-5-7-9	27.3

Tableau AI.2 Résultats comparatifs de routage pour un trafic de 15 paq/s

Heuristiques de routage	T_{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	26.6	1-9	1-5-7-9	32.0
Nombre minimal de hops	74.0	1-9	1-5-7-9	53.8
Nouvelle approche	22.2	1-9	1-5-4-8-9	20.0

Tableau AI.3 Résultats comparatifs de routage pour un trafic de 20 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	37.5	1-9	1-5-7-9	59.7
Nombre minimal de hops	32.8	1-9	1-5-7-9	21.9
Nouvelle approche	14.8	1-9	1-2-3-8-9	30.1

Tableau AI.4 Résultats comparatifs de routage pour un trafic de 25 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	18.3	1-9	1-5-7-9	25.4
Nombre minimal de hops	20.3	1-9	1-5-7-9	15.8
Nouvelle approche	10.2	1-9	1-2-4-8-9	15.3

Tableau AI.5 Résultats comparatifs de routage pour un trafic de 30 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	11.3	1-9	1-5-7-9	16.0
Nombre minimal de hops	37.0	1-9	1-5-7-9	26.9
Nouvelle approche	11.3	1-9	1-5-4-8-9	16.3

Tableau AI.6 Résultats comparatifs de routage pour un trafic de 35 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	21.7	1-9	1-5-7-9	33.9
Nombre minimal de hops	63.5	1-9	1-5-7-9	200.8
Nouvelle approche	8.1	1-9	1-5-7-9	6.9

Tableau AI.7 Résultats comparatifs de routage pour un trafic de 40 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	18.7	1-9	1-5-7-9	29.9
Nombre minimal de hops	16.3	1-9	1-5-7-9	10.9
Nouvelle approche	6.3	1-9	1-5-7-9	7.5

Tableau AI.8 Résultats comparatifs de routage pour un trafic de 45 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	56.7	1-9	1-5-7-9	8.8
Nombre minimal de hops	19.7	1-9	1-5-7-9	20.6
Nouvelle approche	6.4	1-9	1-2-4-8-9	8.7

Tableau AI.9 Résultats comparatifs de routage pour un trafic de 50 paq/s

Heuristiques de routage	T _{moy} (ms)	Commodité	Chemin	Délai du chemin (ms)
Plus courte distance	9.2	1-9	1-5-7-9	12.7
Nombre minimal de hops	9.9	1-9	1-5-7-9	7.9
Nouvelle approche	8.7	1-9	1-2-4-8-9	10.2

AI.3 Résultats détaillés obtenus des algorithmes de routage

Nous présentons ici les résultats détaillés obtenus par l'application des algorithmes de routage au réseau de la Figure 3.8. Nous exposons d'abord les résultats obtenus des heuristiques, ensuite ceux des algorithmes de routage optimal. Présentons d'abord les coordonnées des noeuds utilisées.

Tableau AI.10 Coordonnées des noeuds

Ville	NYK	LSA	CHI	DAL	BAL	SFO	MIA	DEN
Noeud	1	2	3	4	5	6	7	8
Abscisse	444	60	324	287	433	8	507	176
Ordonnée	447	16	346	96	396	89	103	191

Présentons les attributs de chaque liaison du réseau, tels le flot, la capacité, ainsi que le délai obtenu de chacune des heuristiques.

Tableau AL.11 Attributs des liaisons selon les trois heuristiques

	Plus courte distance			Nombre minimal de hops			Nouvelle stratégie		
Liaison	Flot (kbps)	Capacité (kbps)	Délai (s)	Flot (kbps)	Capacité (kbps)	Délai (s)	Flot (kbps)	Capacité (kbps)	Délai (s)
(1, 5)	28.00	28.80	1.2500	28.00	28.80	1.2500	28.00	38.40	0.0962
(2, 6)	16.00	19.20	0.3125	4.00	9.60	0.1786	6.00	9.60	0.2778
(2, 8)	36.00	38.40	0.4167	24.00	28.80	0.2083	26.00	38.40	0.0806
(3, 4)	12.00	19.20	0.1389	16.00	19.20	0.3125	10.00	19.20	0.1087
(3, 5)	24.00	28.80	0.2083	20.00	28.80	0.1136	16.00	28.80	0.0781
(3, 6)	12.00	19.20	0.1389	24.00	28.80	0.2083	22.00	28.80	0.1471
(3, 7)	4.00	9.60	0.1786	12.00	19.20	0.1389	10.00	19.20	0.1087
(3, 8)	8.00	9.60	0.6250	12.00	19.20	0.1389	10.00	19.20	0.1087
(4, 8)	16.00	19.20	0.3125	12.00	19.20	0.1389	18.00	28.80	0.0926
(5, 7)	8.00	9.60	0.6250	8.00	9.60	0.6250	8.00	19.20	0.0893
(5, 8)	16.00	19.20	0.3125	20.00	28.80	0.1136	24.00	28.80	0.2083
(7, 8)	16.00	19.20	0.3125	8.00	9.60	0.6250	10.00	19.20	0.1087
	$T_{\text{moy}} = 0.7951 \text{ s}$			$T_{\text{moy}} = 0.6273 \text{ ms}$			$T_{\text{moy}} = 0.2024 \text{ ms}$		

Présentons maintenant l'évolution des paramètres λ de la méthode de déviation de flot et α de l'algorithme de Bertsekas-Gallager après chaque itération, avant de converger vers la solution optimale.

Tableau AI.12 Évolution des paramètres des algorithmes de routage optimal

Itération	Méthode de déviation de flot		Algorithme de Bertsekas-Gallager	
	λ	T_{moy} (ms)	α	T_{moy} (ms)
1	-	0.246	-	0.246
2	0.2	0.207	0.3750	0.196
3	0.121	0.201	0.3750	0.190
4	0.122	0.198	0.3750	0.189
5	0.253	0.195	0.3750	0.189
6	0.072	0.194	0.3750	0.189
7	0.148	0.193	0.3750	0.188
8	0.034	0.193	0.3750	0.188
9	0.145	0.192	0.3750	0.188
10	0.031	0.192	0.3750	0.188
11	0.097	0.191	0.3750	0.187
12	0.038	0.191	0.3750	0.187
13	0.094	0.191	0.3750	0.187
14	0.025	0.190	0.1875	0.187
15			0.09375	0.187
16			0.046875	0.187

Présentons maintenant les attributs de liaisons obtenus des méthodes classiques de routage optimal avec $\lambda = 0.025$ pour la méthode de déviation de flot et $\alpha = 0.046875$ pour l'algorithme de Bertsekas-Gallager.

**Tableau AI.13 Attributs de liaisons proches de l'optimalité des algorithmes
de routage optimal**

Liaison	Méthode de déviaton de flot			Algorithme de Bertsekas-Gallager		
	Flot (kbps)	Capacité (kbps)	Délai (s)	Flot (kbps)	Capacité (kbps)	Délai (s)
(1, 5)	14.00	28.80	0.0676	14.00	28.80	0.0676
(2, 6)	8.86	19.20	0.0967	8.75	19.20	0.0957
(2, 8)	18.80	38.40	0.0510	18.75	38.40	0.0509
(3, 4)	13.94	28.80	0.0673	13.98	28.80	0.0675
(3, 5)	16.56	38.40	0.0458	15.97	38.40	0.0446
(3, 6)	5.14	9.60	0.2242	5.25	9.60	0.2299
(3, 7)	4.34	9.60	0.1901	4.43	9.60	0.1934
(3, 8)	3.60	9.60	0.1667	3.18	9.60	0.1558
(4, 8)	12.50	28.80	0.0613	11.98	28.80	0.0594
(5, 7)	4.18	9.60	0.1845	4.00	9.60	0.1786
(5, 8)	4.20	9.60	0.1852	4.02	9.60	0.1792
(7, 8)	5.45	9.60	0.2410	5.57	9.60	0.2481
	$T_{\text{mov}} = 0.190 \text{ s}$			$T_{\text{mov}} = 0.187 \text{ s}$		

AI.4 Coordonnées des noeuds du réseau à haut débit

Présentons les coordonnées des noeuds du réseau à haut débit des figures 3.9 et 4.2. Il s'agit de coordonnées écran permettant de visualiser le réseau lors de l'exécution du programme.

Tableau AI.14 Coordonnées des noeuds du réseau

Tableau AI.14 a) Coordonnées des noeuds 1 à 10

Noeud	1	2	3	4	5	6	7	8	9	10
Abscisse	25	50	80	110	150	150	175	210	250	275
Ordonnée	125	325	225	50	150	450	225	350	200	425

Tableau AL.14 b) Coordonnées des noeuds 11 à 20

Noeud	11	12	13	14	15	16	17	18	19	20
Abscisse	300	300	330	350	350	470	400	525	550	575
Ordonnée	100	475	250	150	325	400	250	75	175	375

ANNEXE II

RÉSULTATS DÉTAILLÉS D'INTÉGRATION DES ASPECTS DE ROUTAGE ET DE SURVIVABILITÉ

Cette partie complète les résultats exposés au CHAPITRE IV. Nous y présentons d'abord les principales fonctions d'intégration de notre méthode de routage et de survivabilité, ensuite un tableau donnant les deux chemins disjoints de liaisons pour toutes les paires de noeuds du réseau à haut débit de la figure 4.2 et, pour finir, les graphiques illustrant le comportement des liaisons 4 à 30 de ce même réseau après une défaillance.

AII.1 Principales fonctions d'intégration de routage et de survivabilité

Nous présentons les structures de données et les principales fonctions qui implémentent notre méthode d'intégration de routage et de survivabilité. Cette implémentation est effectuée en TURBO C version 3.0 et utilise le système d'exploitation DOS.

```

/*****
/*                               */
/*****

#define TraficMoyen  150000 /* trafic considere entre chaque paire de noeuds */
#define BitParPaquet  1000
                               /* taille moyenne d'un paquet */

#define Infini      99000001L
#define MaxNoeud    28
#define MAX(x, y) ((x) > (y) ? (x) : (y));

typedef struct ListeLiaison {
    int No, No1, No2, NbPaq; /* numero et noeuds incidents a la liaison*/
    double Dist; /* longueur, flot et capacite de la liaison */
    float Ti, Tprop, Ttr, Ttotal, Ttotal1;
    long capa, reste, flot, deltaflot, deltatampon, flotMax, capaSurviv, flotReel;
    struct ListeLiaison *Prec, *Suivant;
} ListeLiaison;

typedef struct ListeNoeud {
    int No, X, Y;
    double dist, dist1;
    ListeLiaison *lien, *lien1;
    struct ListeNoeud *chemin, *chemin1;
    struct ListeNoeud *Prec, *Suivant;
} ListeNoeud;

Static ListeNoeud *tas1[MaxNoeud], *tas2[MaxNoeud];
Static int nodepart, noarrive, NbreNoeud, NbreLiaison, panne;
Static ListeNoeud *NDebut, *NFin, *Nnouveau, *NTemp, *No1, *No2, *NCase;
Static ListeLiaison *LDebut, *LFin, *LNouveau, *LTemp, *LCase, *Chemin;
Static double NbreTab1, Distance, DistanceTotal,
              MaxLiaison, NbreTab01;
Static float delai, DelaiChemin, temps_exec;
/*****

/*****
/* Principales fonctions de notre methode d'integration de routage et de survivabilite */
/*****

Static Void NewApproach()
{
    uchar i;
    Etape1();
    Etape2();
    CapaSurviv();
    panne = 0;
    Comportement(panne);
    Avertissement(panne);
    Tas(nodepart);
    PccDelai();
}

```

```

RechercherChemin(panne);
randomize();
panne = 1 + random(NbreLiaison);
Comportement(panne);
Avertissement(panne);
Tas(nodepart);
PccDelai();
RechercherChemin(panne);
}      /* fin de la fonction NewApproach */

```

```

Static Void Etape1()
{
    ListeLiaison *Lien;
    ListeNoeud *Noeud;
    float caparequis, u, ro, po, t1, t2, t3, t4;
    uchar i, j, no, FORLIM;
    int k;

    delai = 0.0;
    Lien = LDebut;

    while (Lien != NULL) {
        Lien->flot = 2*TraficMoyen;
        Lien->Ti = 600.0/(float) TraficMoyen;
        caparequis = (float) (BitParPaquet)/Lien->Ti + (float) (Lien->flot);

        if (caparequis < 1544000L)
            Lien->capa = 1544000L;    /* signal DS-1 */
        else if (caparequis < 3152000L)
            Lien->capa = 3152000L;    /* signal DS-1C */
        else if (caparequis < 6312000L)
            Lien->capa = 6312000L;    /* signal DS-2 */
        else if (caparequis < 12624000L)
            Lien->capa = 12624000L;   /* 2 DS-2 */
        else if (caparequis < 18528000L)
            Lien->capa = 18528000L;   /* 12 DS-1 */
        else if (caparequis < 25248000L)
            Lien->capa = 25248000L;   /* 4 DS-2 */
        else if (caparequis < 44736000L)
            Lien->capa = 44736000L;   /* signal DS-3 */
        else if (caparequis < 89472000L)
            Lien->capa = 89472000L;   /* 2 DS-3 */
        else if (caparequis < 178944000L)
            Lien->capa = 178944000L;  /* 4 DS-3 */
        else if (caparequis < 274176000L)
            Lien->capa = 274176000L;  /* signal DS-4 */
        else {
            congestion();
            delay(15000); exit(1);
        }
    }
}

```

```

    }
    Lien->reste = Lien->capa - Lien->flot; /* capacite, residuelle */
    Lien->Tprop = (1/3.0e2)*Lien->Dist; /* delai de propagation */
    Lien->Ttr = 1000.0 * (float) (BitParPaquet)/(float) (Lien->capa); /* delai de
transmission */
    Lien->NbPaq = (int) (Lien->Tprop/Lien->Ttr); /* Nombre de paquets simultann,ment
sur la liaison */
    if (Lien->NbPaq < 1)
        Lien->NbPaq = 1;

    u = (float) Lien->flot/(float) Lien->capa; /* utilisation de la liaison */
    ro = u/(float) Lien->NbPaq; /* intensite de circulation */
    t1 = 0.0;
    for(k=0; k < Lien->NbPaq; k++) {
        if (k <= 10) /* pour limiter factoriel de k */
            t1 = t1 + pow(u, k)/(float) Factoriel(k);
    }
    if(Lien->NbPaq <= 12)
        t2 = pow(u, Lien->NbPaq)/(float) (Factoriel(Lien->NbPaq) * (1 - ro));
    else
        t2 = pow(u, Lien->NbPaq)/(float) (Factoriel(12) * (1 - ro));
    po = 1.0/(t1 + t2);

    t3 = (Lien->NbPaq) * po * pow(u, Lien->NbPaq);
    if (Lien->NbPaq <= 12)
        t4 = (float) Factoriel(Lien->NbPaq) * pow((Lien->NbPaq - u), 2);
    else
        t4 = (float) Factoriel(12) * pow((Lien->NbPaq - u), 2);
    Lien->Ttotal = 1000.0 * (((float) BitParPaquet/(float) Lien->capa) * (1.0 + t3/t4)) +
Lien->Tprop;
/* delai total comprenant attente, transmission et
propagation sur la liaison selon nouvelle methode */
    Lien->Ttotal1 = Lien->Ttotal;
    Lien = Lien->Suivant;
}
} /* fin de la fonction Etape1() */

```

```

Static Void Etape2()
{
    ListeNoeud *ptr, *ptr2;
    ListeLiaison *lien;
    uchar no, FORLIM;
    float caparequis, u, ro, po, t1, t2, t3, t4;
    int k;

    FORLIM = NbreNoeud;

    for (no = 1; no <= FORLIM; no++) {
        Tas(no);
    }
}

```

```

PccDelai();
comparerDelai(no);
}

delai = 0.0;
lien = LDebut;
while (lien != NULL) {
    lien->Ti = 600.0/(float) TraficMoyen;
    caparequis = (float) (BitParPaquet)/lien->Ti + (float) (lien->flot);
    if (caparequis < 1544000L)
        lien->capa = 1544000L; /* signal DS-1 */
    else if (caparequis < 3152000L) /* signal DS-1C */
        lien->capa = 3152000L;
    else if (caparequis < 6312000L) /* signal DS-2 */
        lien->capa = 6312000L;
    else if (caparequis < 12624000L) /* 2 DS-2 */
        lien->capa = 12624000L;
    else if (caparequis < 18528000L) /* 12 DS-1 */
        lien->capa = 18528000L;
    else if (caparequis < 25248000L) /* 4 DS-2 */
        lien->capa = 25248000L;
    else if (caparequis < 44736000L) /* signal DS-3 */
        lien->capa = 44736000L;
    else if (caparequis < 89472000L) /* 2 DS-3 */
        lien->capa = 89472000L;
    else if (caparequis < 178944000L) /* 4 DS-3 */
        lien->capa = 178944000L;
    else if (caparequis < 274176000L) /* signal DS-4 */
        lien->capa = 274176000L;

    else {
        congestion();
        delay(15000); exit(1);
    }

    lien->reste = lien->capa - lien->flot;
    lien->Tprop= (1/3.0e2)*lien->Dist; /* delai de propagation */
    lien->Ttr = 1000.0 * (float) (BitParPaquet)/(float) (lien->capa); /* delai de transmission */
    lien->NbPaq = (lien->Tprop/lien->Ttr); /* Nombre de paquets simultann,ment sur la liaison */
    if (lien->NbPaq < 1)
        lien->NbPaq = 1;

    u = (float) lien->flot/(float) lien->capa; /* utilisation de la liaison */
    ro = u/(float) lien->NbPaq; /* intensite de circulation */
    t1 = 0.0;
    for(k=0; k < lien->NbPaq; k++) {
        if (k <= 10) /* pour limiter factoriel de k */
            t1 = t1 + pow(u, k)/(float) Factoriel(k);
    }
    if(lien->NbPaq <= 12)
        t2 = pow(u, lien->NbPaq)/(float) (Factoriel(lien->NbPaq) * (1 - ro));
    else
        t2 = pow(u, lien->NbPaq)/(float) (Factoriel(12) * (1 - ro));
}

```

```

    po = 1.0/(t1 + t2);

    t3 = (lien->NbPaq) * po * pow(u, lien->NbPaq);
    if (lien->NbPaq <= 12)
        t4 = (float) Factoriel(lien->NbPaq) * pow((lien->NbPaq - u), 2);
    else
        t4 = (float) Factoriel(12) * pow((lien->NbPaq - u), 2);
    lien->Ttotal = 1000.0 * (((float) BitParPaquet/(float) lien->capa) * (1.0 + t3/t4)) + lien->Tprop;
    /* delai total comprenant attente, transmission et
       propagation sur la liaison selon nouvelle methode */

    lien = lien->Suivant;
}
} /* fin de la fonction Etape2() */

```

```

/*****

```

```

/* CapaSurviv: Permet de determiner la capacite de chaque liaison capable */

```

```

/* de resister a la perte de n'importe quelle liaison */

```

```

Static Void CapaSurviv()

```

```

{
    ListeNoeud *ptr, *ptr2;
    ListeLiaison *lien;
    uchar no, FORLIM;
    float caparequis, u, ro, po, t1, t2, t3, t4;
    int k, rupture, NL;

```

```

    FORLIM = NbreNoeud;

```

```

    NL = NbreLiaison;

```

```

/* Consideration de possibilite de rupture de chacune des liaisons */

```

```

    InitDeltaFlot();

```

```

    for (rupture = 1; rupture <= NL; rupture++){

```

```

        InitDeltaTampon(); /* Initialisation de variation de flot */

```

```

        for (no = 1; no <= FORLIM; no++) {

```

```

            Tas(no);

```

```

            PccDelaiSurviv(rupture);

```

```

        }

```

```

        lien = LDebut;

```

```

        while (lien != NULL) {

```

```

            lien->deltaflot = MAX(lien->deltaflot, lien->deltatampon);

```

```

            lien = lien->Suivant;

```

```

        }

```

```

    }

```

```

    lien = LDebut;

```

```

    while (lien != NULL) {

```

```

        lien->flotMax = lien->flot + lien->deltaflot;

```

```

        if (lien->flotMax < 1544000L)

```

```

        lien->capaSurviv = 1544000L;    /* signal DS-1 */
    else if ( lien->flotMax < 3152000L) /* signal DS-1C */
        lien->capaSurviv = 3152000L;
    else if ( lien->flotMax < 6312000L) /* signal DS-2 */
        lien->capaSurviv = 6312000L;
    else if ( lien->flotMax < 12624000L) /* 2 DS-2 */
        lien->capaSurviv = 12624000L;
    else if ( lien->flotMax < 18528000L) /* 12 DS-1 */
        lien->capaSurviv = 18528000L;
    else if ( lien->flotMax < 25248000L) /* 4 DS-2 */
        lien->capaSurviv = 25248000L;
    else if ( lien->flotMax < 44736000L) /* signal DS-3 */
        lien->capaSurviv = 44736000L;
    else if ( lien->flotMax < 89472000L) /* 2 DS-3 */
        lien->capaSurviv = 89472000L;
    else if ( lien->flotMax < 178944000L) /* 4 DS-3 */
        lien->capaSurviv = 178944000L;
    else if ( lien->flotMax < 274176000L) /* signal DS-4 */
        lien->capaSurviv = 274176000L;

    else {
        congestion();
        delay(15000); exit(1);
    }

    lien = lien->Suivant;
}
} /* fin de la fonction CapaSurviv() */

```

```

/*****
/* Comportement: Traduit le nouveau comportement du reseau lors de la perte */
/* d'une liaison */
Static Void Comportement(int rupture)
{
    ListeNoeud *ptr, *ptr2;
    ListeLiaison *lien;
    uchar no, FORLIM;
    float caparequis, u, ro, po, t1, t2, t3, t4;
    int k;

    FORLIM = NbreNoeud;

    InitDeltaTampon();
    for (no = 1; no <= FORLIM; no++) {
        Tas(no);
        PccDelaiSurviv(rupture);
    }

    delai = 0.0;
    lien = LDebut;

```

```

while (lien != NULL) {
    if (lien->No == rupture){
        lien->flotReel = 0;
        lien->Ttotal = Infini;
    }
    else {
        lien->flotReel = lien->flot + lien->deltatampon;
        lien->reste = lien->capaSurviv - lien->flotReel;
        lien->Tprop= (1/3.0e2)*lien->Dist; /* delai de propagation */
        lien->Ttr = 1000.0 * (float) (BitParPaquet)/(float) (lien->capaSurviv); /* delai de transmission */
        lien->NbPaq = (lien->Tprop/lien->Ttr); /* Nombre de paquets simultann,ment sur la liaison */
        if (lien->NbPaq < 1)
            lien->NbPaq = 1;

        u = (float) lien->flotReel/(float) lien->capaSurviv; /* utilisation de la liaison */
        ro = u/(float) lien->NbPaq; /* intensite de circulation */
        t1 = 0.0;
        for(k=0; k < lien->NbPaq; k++) {
            if (k <= 10) /* pour limiter factoriel de k */
                t1 = t1 + pow(u, k)/(float) Factoriel(k);
        }
        if(lien->NbPaq <= 12)
            t2 = pow(u, lien->NbPaq)/(float) (Factoriel(lien->NbPaq) * (1 - ro));
        else
            t2 = pow(u, lien->NbPaq)/(float) (Factoriel(12) * (1 - ro));
        po = 1.0/(t1 + t2);

        t3 = (lien->NbPaq) * po * pow(u, lien->NbPaq);
        if (lien->NbPaq <= 12)
            t4 = (float) Factoriel(lien->NbPaq) * pow((lien->NbPaq - u), 2);
        else
            t4 = (float) Factoriel(12) * pow((lien->NbPaq - u), 2);
        lien->Ttotal = 1000.0 * (((float) BitParPaquet/(float) lien->capaSurviv) * (1.0 + t3/t4)) +
        lien->Tprop;

        /* delai total comprenant attente, transmission et
        propagation sur la liaison selon nouvelle methode */
        delai = delai + (1/(float) BitParPaquet) * (float) lien->flotReel * lien->Ttotal;
        /* delai moyen a haut debit selon methode propos,e */
    }
    lien = lien->Suivant;
}
} /* fin de la fonction Comportement(rupture) */

```

```

/*****

```

```

/* PccDelai: Trouve le chemin au plus court delai entre chaque paire de noeuds*/

```

```

Static Void PccDelai()

```

```

{
    ListeNoeud *tpn, *tpn0, dest;
    ListeLiaison *tpl, tpl1;
    long i, j;

    /*tpn0 = tas1[0];*/
    for (i = NbreNoeud; i >= 1; i--) {
        tpn = tas1[0];
        tpl = LDebut;
        while (tpl != NULL) {
            if (tpl->No1 == tpn->No || tpl->No2 == tpn->No) {
                for (j = 1; j < i; j++) {
                    if (tas1[j]->No == tpl->No1 || tas1[j]->No == tpl->No2) {
                        if (tas1[j]->dist > tpn->dist + tpl->Ttotal) {
                            tas1[j]->dist = tpn->dist + tpl->Ttotal;
                            tas1[j]->chemin = tpn;
                            tas1[j]->lien = tpl;
                            Tasascendant(j + 1);
                        }
                    }
                }
            }
            tpl = tpl->Suivant;
        }
        tas1[0] = tas1[i - 1];
        Tasdescendant(1L, i);    /* pour remonter au noeud le plus proche du noeud source */
    }
}

```

```

/*****
/* CheminSecours : determine le chemin de secours entre chaque paire de
/*      noeuds                               */

```

```

Static Void CheminSecours(source, destination)

```

```

ListeNoeud *source, *destination;

```

```

{
    ListeLiaison *tpl1;
    ListeNoeud *tampon, *dest;
    int j, i;

```

```

    InitTabSecond();    /* initialisation du second tableau de pointeurs de noeuds */

```

```

    Tas2(source->No);    /* reorganisation de ce tableau */

```

```

    dest = destination;

```

```

    while (dest != source){
        dest->lien->Ttotal1 = 2 * Infini;    /* initialisation du delai de chaque */
        dest->lien->Ttotal1 = 2 * Infini;    /* liaison du chemin principal a une */
        dest = dest->chemin;                /* valeur tres grande */
    }

```

```

    for(i=NbreNoeud; i>=1; i--) {

```

```

tampon = tas2[0];    /* noeud le plus proche du noeud source */
tpl1 = LDebut;      /* liaison 1 */
while (tpl1 != NULL) {
    if (tpl1->No1 == tampon->No || tpl1->No2 == tampon->No) {
        for (j = 1; j < i; j++) {
            if (tas2[j]->No == tpl1->No1 || tas2[j]->No == tpl1->No2) {
                if (tas2[j]->dist1 > tampon->dist1 + tpl1->Ttotal1) {
                    tas2[j]->dist1 = tampon->dist1 + tpl1->Ttotal1;
                    tas2[j]->chemin1 = tampon;
                    tas2[j]->lien1 = tpl1;
                    Tasascendant2(j + 1);
                } /* fin de troisieme if */
            } /* fin de second if */
        } /* fin de la boucle for */

        } /* fin de premier if */
    tpl1 = tpl1->Suivant;
} /* fin de while */
tas2[0] = tas2[i-1];
Tasdescendant2(1L, i);
if (tas2[0] == destination) {
    dest = destination;
    while (dest != source) { /* une fois le chemin de secours defini, */
        dest->lien->Ttotal1 = dest->lien->Ttotal; /* remise du delai de */
        dest = dest->chemin; /* chaque liaison du chemin */
    } /* principal a sa bonne valeur */
    break;
} /* fin de if */
} /* fin de premier for */
} /* fin de la fonction CheminSecours */

```

/******

/* PccDelaiSurviv: Trouve le chemin au plus court delai entre chaque paire de */

/* noeuds avec possibilite de rupture d'une liaison de liaisons*/

Static Void PccDelaiSurviv(int lien_en_panne)

{

ListeNoeud *tpn, *tpn0, *dest, *tampon;

ListeLiaison *tpl;

int i, j, k;

tpn0 = tampon = tas1[0];

for (i = NbreNoeud; i >= 1; i--) {

tpn = tas1[0];

tpl = LDebut;

while (tpl != NULL) {

if (tpl->No1 == tpn->No || tpl->No2 == tpn->No) {

for (j = 1; j < i; j++) {

if (tas1[j]->No == tpl->No1 || tas1[j]->No == tpl->No2) {

```

        if (tasl[j]->dist > tpn->dist + tpl->Ttotal) {
            tasl[j]->dist = tpn->dist + tpl->Ttotal;
            tasl[j]->chemin = tpn;
            tasl[j]->lien = tpl;
            Tasascendant(j + 1);
        }
    }
}
tpl = tpl->Suivant;
}
tasl[0] = tasl[i - 1];
Tasdescendant(1L, i); /* pour remonter au noeud le plus proche du noeud source */

if (tasl[0] != tampon) { /* pour s'assurer que destination ne soit atteinte a plusieurs reprises */
    tampon = tasl[0];
    dest = tasl[0]; /*on connait chemin entre source et destination */
    while (dest!=tpn0 && dest->lien->No!=lien_en_panne) {
        dest = dest->chemin; /* pour determiner si la liaison en panne*/
    }
    /* fait partie du chemin principal */

    if (dest!=tpn0 && dest->lien->No==lien_en_panne) /* si liaison en panne */
    {
        /* fait partie de chemin principal */
        dest = tasl[0];
        while (dest!=tpn0){ /* retrancher trafic de la paire de chaque liaison */
            if (dest->lien->No != lien_en_panne) /* du chemin principal */
                dest->lien->deltatampon = dest->lien->deltatampon - TraficMoyen;
            dest = dest->chemin;
        }
        dest = tasl[0];
        CheminSecours(tpn0, dest); /* determination du chemin de secours */
        while (dest!=tpn0){ /* ajouter trafic de la paire a chaque liaison */
            /* du chemin de secours */
            dest->lien->deltatampon = dest->lien->deltatampon + TraficMoyen;
            dest = dest->chemin;
        }
    } /* fin du dernier if */
} /* fin de l'avant dernier if */
} /* fin du premier for */
} /* fin de la fonction PccDelaiSurviv */

```

```

/*****
/* RechercherChemin: Determine le chemin entre la source et la destination en tenant */
/* compte de la rupture possible d'une liaison faisant partie du */
/* chemin principal */
Static Void RechercherChemin(int lien_en_panne)
{
    ListeNoeud *source, *dest;

```

```

source = dest = NDebut;
while (dest->No != noarrive)
    dest = dest->Suivant;
while (source->No != nodepart)
    source = source->Suivant;
while (dest!=source && dest->lien->No!=lien_en_panne) {
    dest = dest->chemin; /* pour determiner si la liaison en panne*/
                        /* fait partie du chemin principal */
}

if (dest!=source && dest->lien->No==lien_en_panne) /* si liaison en panne */
    AfficherCheminSecond(); /* fait partie de chemin principal */
else
    afficherchemin();
} /* fin de la fonction RechercherChemin */

```

```

Static Void InitDelta()
{
    ListeLiaison *lien;
    lien = LDebut;
    while(lien != NULL) {
        lien->deltaflot = 0;
        lien->deltatampon = 0;
        lien = lien->Suivant;
    }
}

```

```

Static Void InitDeltaTampon()
{
    ListeLiaison *lien;
    lien = LDebut;
    while(lien != NULL) {
        lien->deltatampon = 0;
        lien = lien->Suivant;
    }
}

```

```

Static Void InitDeltaFlot()
{
    ListeLiaison *lien;
    lien = LDebut;
    while(lien != NULL) {
        lien->deltaflot = 0;
    }
}

```

```

    lien = lien->Suivant;
  }
}

```

```

/*****

```

```

/* comparerDelai: calcule le flot de chaque Liaison faisant partie */
/* du chemin au plus court delai */

```

```

Static Void comparerDelai(uchar no)

```

```

{
  ListeNoeud *ptr, *ptr2;
  ListeLiaison *Liaison;

```

```

  float caparequis, u, ro, po, t1, t2, t3, t4;
  int k;

```

```

  ptr = NDebut;

```

```

  while (ptr != NULL) {

```

```

    if (ptr->No != no && ptr->chemin->No != no) {

```

```

      ptr2 = ptr;

```

```

      while (ptr2->No != no) {

```

```

        ptr2->lien->flot += TraficMoyen;

```

```

        ptr2->lien->Ti = 600.0/(float)TraficMoyen;

```

```

        caparequis = (float) (BitParPaquet)/ptr2->lien->Ti + (float) (ptr2->lien->flot);

```

```

      if (caparequis < 1544000L)

```

```

        ptr2->lien->capa = 1544000L; /* signal DS-1 */

```

```

      else if (caparequis < 3152000L) /* signal DS-1C */

```

```

        ptr2->lien->capa = 3152000L;

```

```

      else if (caparequis < 6312000L) /* signal DS-2 */

```

```

        ptr2->lien->capa = 6312000L;

```

```

      else if (caparequis < 12624000L) /* 2 DS-2 */

```

```

        ptr2->lien->capa = 12624000L;

```

```

      else if (caparequis < 18528000L) /* 12 DS-1 */

```

```

        ptr2->lien->capa = 18528000L;

```

```

      else if (caparequis < 25248000L) /* 4 DS-2 */

```

```

        ptr2->lien->capa = 25248000L;

```

```

      else if (caparequis < 44736000L) /* signal DS-3 */

```

```

        ptr2->lien->capa = 44736000L;

```

```

      else if (caparequis < 89472000L) /* 2 DS-3 */

```

```

        ptr2->lien->capa = 89472000L;

```

```

      else if (caparequis < 178944000L) /* 4 DS-3 */

```

```

        ptr2->lien->capa = 178944000L;

```

```

      else if (caparequis < 274176000L) /* signal DS-4 */

```

```

        ptr2->lien->capa = 274176000L;

```

```

      else {

```

```

        congestion();

```

```

        delay(15000); exit(1);

```

```

      }

```

```

      ptr2->lien->reste = ptr2->lien->capa - ptr2->lien->flot;

```

```

ptr2->lien->Tprop= (1/3.0e2)*ptr2->lien->Dist; /* delai de propagation */
ptr2->lien->Ttr = 1000.0 * (float) (BitParPaquet)/(float) (ptr2->lien->capa); /* delai de
transmission */
ptr2->lien->NbPaq = (ptr2->lien->Tprop/ptr2->lien->Ttr); /* Nombre de paquets
simultann,ment sur la liaison */
if (ptr2->lien->NbPaq < 1)
    ptr2->lien->NbPaq = 1;
u = (float) ptr2->lien->flot/(float) ptr2->lien->capa; /* utilisation de la liaison */
ro = u/(float) ptr2->lien->NbPaq; /* intensite de circulation */
t1 = 0;
for(k=0; k < ptr2->lien->NbPaq; k++) {
    if (k <= 10) /* pour limiter factoriel de k */
        t1 = t1 + pow(u, k)/(float) Factoriel(k);
}

if(ptr2->lien->NbPaq <= 12)
    t2 = pow(u, ptr2->lien->NbPaq)/(float) (Factoriel(ptr2->lien->NbPaq) * (1 -
ro));
else
    t2 = pow(u, ptr2->lien->NbPaq)/(float) (Factoriel(12) * (1 - ro));
po = 1.0/(t1 + t2);

t3 = (ptr2->lien->NbPaq) * po * pow(u, ptr2->lien->NbPaq);
if (ptr2->lien->NbPaq <= 12)
    t4 = (float) Factoriel(ptr2->lien->NbPaq) * pow((ptr2->lien->NbPaq - u), 2);
else
    t4 = (float) Factoriel(12) * pow((ptr2->lien->NbPaq - u), 2);

ptr2->lien->Ttotal = 1000.0 * (((float) BitParPaquet/(float) ptr2->lien->capa) * (1.0 +
t3/t4)) + ptr2->lien->Tprop;
/* delai total comprenant attente, transmission et
propagation sur la liaison selon nouvelle methode */
ptr2->lien->Ttotal1 = ptr2->lien->Ttotal;
ptr2 = ptr2->chemin;
}
} /* fin du premier if */
ptr = ptr->Suivant;
} /* fin du premier while */
}

/*****/
/* Initialisation du second tableau de pointeurs pour determiner */
/* le chemin secondaire */
Static Void InitTabSecond()
{
    int i;
    for(i=0; i<MaxNoeud; i++)
        tas2[i] = tas1[i];
}

```

```

/*****
/* Tas: cree la liste des pointeurs sur les noeuds pour un heap */
Static Void Tas(uchar no)
{
    ListeNoeud *ptr;
    long i, n, FORLIM;

    ptr = NDebut;
    n = 2;
    FORLIM = NbreNoeud;
    for (i = 1; i <= FORLIM; i++) {
        if (ptr->No != no) {
            tas1[n - 1] = ptr;
            n++;
            ptr->dist = Infini;
            ptr->dist1 = Infini;
        } else {
            tas1[0] = ptr;
            ptr->dist = 0.0;
            ptr->dist1 = 0.0;
        }
        ptr = ptr->Suivant;
    }
}

```

```

/*****
/* Tas2: cree la liste des pointeurs sur les noeuds pour un heap */
Static Void Tas2(uchar no)
{
    ListeNoeud *ptr;
    long i, n, FORLIM;

    ptr = NDebut;
    n = 2;
    FORLIM = NbreNoeud;
    for (i = 1; i <= FORLIM; i++) {
        if (ptr->No != no) {
            tas2[n - 1] = ptr;
            n++;
            ptr->dist1 = Infini;
        } else {
            tas2[0] = ptr;
            ptr->dist1 = 0.0;
        }
        ptr = ptr->Suivant;
    }
}

```

```

/*****/
/* Avertissement: Affiche s'il existe une liaison en panne ou non */
Static Void Avertissement(int lien_en_panne)
{
    ListeLiaison *lien;

    if (lien_en_panne == 0){
        Ecran(1, 80, 1, 25, noir);
        Ecran(20, 60, 10, 15, bleu);
        gotoxy(8, 3);
        printf("AUCUNE RUPTURE DE LIAISONS");
        AffBarPresse(jaune, rouge);
        Attendre();
    }
    else {
        lien = LDebut;
        while (lien->No != lien_en_panne)
            lien = lien->Suivant;
        if (lien->No == lien_en_panne) {
            Ecran(1, 80, 1, 25, noir);
            Ecran(20, 60, 5, 20, bleu);
            gotoxy(15, 3);
            printf("ATTENTION");
            gotoxy(12, 7);
            printf("LA LIAISON %4d",lien->No);
            gotoxy(4, 9);
            printf("ADJACENTE AUX NOEUDS %3d ET %3d", lien->No1, lien->No2);
            gotoxy(10, 11);
            printf("EST TOMBEE EN PANNE");
            AffBarPresse(jaune, rouge);
            Attendre();
        }
        else {
            sound(100);
            delay(500);
            nosound();
            Ecran(20, 60, 15, 20, rouge);
            textcolor(blanc);
            gotoxy(19, 2);
            printf("ERREUR!");
            textcolor(jaune);
            gotoxy(10, 4);
            printf("LIAISON EN PANNE INCONNUE");
            AffBarPresse(blanc, rouge);
            Attendre();
        }
    }
}

```

AII.2 Chemin principal et chemin de secours de chaque paire de noeuds

Comme nous le savons, il existe $n(n-1)/2$ paires de source/destination pour un réseau de n noeuds. De ce fait, le réseau de la Figure 4.2 contient 190 paires de source/destination. Nous présentons pour chacune d'entre elles les deux chemins disjoints de liaisons les plus rapides, en mentionnant les noeuds qui en font partie et le délai total de chaque paquet sur les liaisons formant ces chemins.

Tableau AII.1 Détermination du chemin principal et du chemin de secours

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
1-2	1-5-9-13-8-10-12-6-2	3.9382	1-3-2	4.2336
1-3	1-3	3.6917	1-5-9-7-3	3.8786
1-4	1-5-11-4	2.1013	1-4	3.6891
1-5	1-5	0.6001	1-4-11-5	5.1904
1-6	1-5-9-13-8-10-12-6	3.3253	1-3-2-6	4.8464
1-7	1-5-9-7	1.3974	1-3-7	6.17-29
1-8	1-5-9-13-8	1.9952	1-3-2-6-12-10-8	6.1765
1-9	1-5-9	1.0521	1-3-7-9	6.5182
1-10	1-5-9-13-8-10	2.3800	1-3-2-6-12-10	5.7917
1-11	1-5-11	1.2877	1-4-11	4.5027
1-12	1-5-9-13-8-10-12	2.6558	1-3-2-6-12	5.5159
1-13	1-5-9-13	1.4206	1-4-11-14-13	6.2288
1-14	1-5-9-13-14	1.8145	1-4-11-14	5.8349
1-15	1-5-9-13-8-10-15	2.8759	1-4-11-14-17-15	9.6022
1-16	1-5-9-13-8-10-12-16	3.3544	1-4-18-19-20-16	7.0167
1-17	1-5-9-13-14-17	2.3744	1-4-18-19-17	9.1223
1-18	1-5-9-13-14-18	2.5283	1-4-18	5.2334
1-19	1-5-9-13-14-18-19	2.9512	1-3-2-6-12-16-20-19	7.5750
1-20	1-5-9-13-14-18-19-20	3.7819	1-3-2-6-12-16-20	6.7443
2-3	2-3	0.5419	2-6-12-10-8-13-9-7-3	5.7125

Tableau AII.1 Détermination du chemin principal et du chemin de secours (suite)

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
2-4	2-6-12-10-8-13-9-5-11-4	4.8394	2-3-1-4	7.9227
2-5	2-6-12-10-8-13-9-5	3.3381	2-3-1-5	4.8337
2-6	2-6	0.6128	2-3-7-9-13-8-10-12-6	5.6416
2-7	2-3-7	3.0231	2-6-12-10-8-13-9-7	3.2314
2-8	2-6-12-10-8	1.9429	2-3-7-9-13-8	4.3115
2-9	2-6-12-10-8-13-9	2.8861	2-3-7-9	3.3684
2-10	2-6-12-10	1.5581	2-3-7-9-13-8-10	4.6963
2-11	2-6-12-10-8-13-9-5-11	4.0258	2-3-1-4-11	8.7363
2-12	2-6-12	1.2823	2-3-7-9-13-8-10-12	4.9721
2-13	2-6-12-10-8-13	2.5176	2-3-7-9-13	3.7368
2-14	2-6-12-10-8-13-14	2.9115	2-3-7-9-5-11-14	5.8403
2-15	2-6-12-10-15	2.0540	2-3-7-9-13-14-17-15	7.8981
2-16	2-6-12-16	1.9809	2-3-7-9-13-14-18-19-20-16	6.6280
2-17	2-6-12-10-8-13-14-17	3.4714	2-3-7-9-5-11-14-18-19-17	10.4430
2-18	2-6-12-10-8-13-14-18	3.6254	2-3-7-9-5-11-4-18	6.8660
2-19	2-6-12-16-20-19	3.3414	2-3-7-9-13-14-18-19	5.2675
2-20	2-6-12-16-20	2.5107	2-3-7-9-13-14-18-19-20	6.0981
3-4	3-7-9-5-11-4	4.7798	3-2-6-12-10-8-13-14-18-4	5.7116
3-5	3-7-9-5	3.2785	3-1-5	4.2917
3-6	3-2-6	1.1547	3-7-9-13-8-10-12-6	5.0997
3-7	3-7	2.4812	3-2-6-12-10-8-13-9-7	3.7733
3-8	3-2-6-12-10-8	2.4849	3-7-9-13-8	3.7696
3-9	3-7-9	2.8265	3-2-6-12-10-8-13-9	3.4280
3-10	3-2-6-12-10	2.1001	3-7-9-13-8-10	4.1544
3-11	3-7-9-5-11	3.9662	3-2-6-12-10-8-13-14-11	4.7856
3-12	3-2-6-12	1.8242	3-7-9-13-8-10-12	4.4302
3-13	3-2-6-12-10-8-13	3.0595	3-7-9-13	3.1949
3-14	3-2-6-12-10-8-13-14	3.4534	3-7-9-5-11-14	5.2984
3-15	3-2-6-12-10-15	2.5959	3-7-9-13-14-17-15	7.3562
3-16	3-2-6-12-16	2.5228	3-7-9-13-14-18-19-20-16	6.0861
3-17	3-2-6-12-10-8-13-14-17	4.0133	3-7-9-5-11-14-18-19-17	9.9011
3-18	3-2-6-12-10-8-13-14-18	4.1673	3-7-9-5-11-4-18	6.3240
3-19	3-2-6-12-16-20-19	3.8833	3-7-9-13-14-18-19	4.7256
3-20	3-2-6-12-16-20	3.0527	3-7-9-13-14-18-19-20	5.5562
4-5	4-11-5	1.5013	4-18-14-13-9-5	3.4725
4-6	4-11-5-9-13-8-10-12-6	4.2265	4-1-3-2-6	8.5355
4-7	4-11-5-9-7	2.2986	4-18-14-13-8-10-12-6-2-3-7	8.1927
4-8	4-11-5-9-13-8	2.8964	4-18-19-20-16-12-10-8	4.6868
4-9	4-11-5-9	1.9533	4-18-14-13-9	3.0205

Tableau AII.1 Détermination du chemin principal et du chemin de secours (suite)

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
4-10	4-11-5-9-13-8	3.2812	4-18-19-20-16-12-10	4.3020
4-11	4-11	0.8136	4-18-14-11	3.5903
4-12	4-11-5-9-13-8-10-12	3.5570	4-18-19-20-16-12	4.0262
4-13	4-11-5-9-13	2.3218	4-18-14-13	2.6520
4-14	4-11-14	2.1458	4-18-14	2.2581
4-15	4-11-5-9-13-8-10-15	3.7771	4-18-14-17-15	6.0255
4-16	4-18-19-20-16	3.3276	4-11-5-9-13-8-10-12-16	4.2556
4-17	4-11-14-17	2.7057	4-18-19-17	5.4332
4-18	4-18	1.5443	4-11-14-18	2.8597
4-19	4-18-19	1.9672	4-11-5-9-13-8-10-12-16-20-19	5.6161
4-20	4-18-19-20	2.7978	4-11-5-9-13-8-10-12-16-20	4.7854
5-6	5-9-13-8-10-12-6	2.7253	5-1-3-2-6	5.4465
5-7	5-9-7	0.7973	5-1-3-7	6.7729
5-8	5-9-13-8	1.3951	5-11-14-18-19-20-16-12-10-8	5.8763
5-9	5-9	0.4520	5-11-14-13-9	2.7822
5-10	5-9-13-8-10	1.7799	5-11-14-18-19-20-16-12-10	5.4915
5-11	5-11	0.6877	5-9-13-14-11	2.5466
5-12	5-9-13-8-10-12	2.0558	5-11-14-18-19-20-16-12	5.2157
5-13	5-9-13	0.8205	5-11-14-13	2.4138
5-14	5-9-13-14	1.2144	5-11-14	2.0199
5-15	5-9-13-8-10-15	2.2758	5-11-14-17-15	5.7872
5-16	5-9-13-8-10-12-16	2.7543	5-11-14-18-19-20-16	4.5171
5-17	5-9-13-14-17	1.7743	5-11-14-18-19-17	6.6226
5-18	5-9-13-14-18	1.9283	5-11-4-18	3.0455
5-19	5-9-13-14-18-19	2.3512	5-11-14-17-19	6.0457
5-20	5-9-13-14-18-19-20	3.1818	5-1-3-2-6-12-16-20	7.3444
6-7	6-12-10-8-13-9-7	2.6185	6-2-3-7	3.6359
6-8	6-12-10-8	1.3301	6-8	3.7000
6-9	6-12-10-8-13-9	2.2732	6-2-3-7-9	3.9812
6-10	6-12-10	0.9453	6-8-10	4.0848
6-11	6-12-10-8-13-9-5-11	3.4130	6-2-3-1-4-11	9.3491
6-12	6-12	0.6695	6-8-10-12	4.3606
6-13	6-12-10-8-13	1.9048	6-2-3-7-9-13	4.3497
6-14	6-12-10-8-13-14	2.2987	6-2-3-7-9-5-11-14	6.4531
6-15	6-12-10-15	1.4412	6-8-13-14-17-15	8.4359
6-16	6-12-16	1.3681	6-8-13-14-18-19-20-16	7.1658
6-17	6-12-10-8-13-14-17	2.8586	6-2-3-7-9-5-11-14-18-19-17	11.0559
6-18	6-12-10-8-13-14-18	3.0126	6-2-3-7-9-5-11-4-18	7.4788

Tableau AII.1 Détermination du chemin principal et du chemin de secours (suite)

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
6-19	6-12-16-20-19	2.7286	6-8-13-14-18-19	5.8053
6-20	6-12-16-20	1.8979	6-8-13-14-18-19-20	6.6359
7-8	7-9-13-8	1.2884	7-3-2-6-12-10-8	4.9660
7-9	7-9	0.3453	7-3-2-6-12-10-8-13-9	5.9091
7-10	7-9-13-8-10	1.6732	7-3-2-6-12-10	4.5812
7-11	7-9-5-11	1.4850	7-3-2-6-12-10-8-13-14-11	7.2668
7-12	7-9-13-8-10-12	1.9490	7-3-2-6-12	4.3054
7-13	7-9-13	0.7138	7-3-2-6-12-10-8-13	5.5407
7-14	7-9-13-14	1.1077	7-3-2-6-12-16-20-19-18-14	7.5012
7-15	7-9-13-8-10-15	2.1691	7-3-2-6-12-16-20-15	9.2091
7-16	7-9-13-8-10-12-16	2.6476	7-3-1-5-11-14-18-19-20-16	11.2900
7-17	7-9-13-14-17	1.6676	7-3-2-6-12-10-15-17	8.2846
7-18	7-9-13-14-18	1.8215	7-3-2-6-12-16-20-19-18	6.7874
7-19	7-9-13-14-18-19	2.2444	7-3-2-6-12-16-20-19	6.3645
7-20	7-9-13-14-18-19-20	3.0751	7-3-2-6-12-16-20	5.5338
8-9	8-13-9	0.9431	8-10-12-6-2-3-7-9	5.3113
8-10	8-10	0.3848	8-13-14-18-19-20-16-12-10	4.4402
8-11	8-13-9-5-11	2.0828	8-10-12-16-20-19-18-14-11	5.1886
8-12	8-18-12	0.6606	8-13-14-18-19-20-16-12	4.1644
8-13	8-13	0.5747	8-10-12-16-20-19-18-14-13	4.2503
8-14	8-13-14	0.9686	8-10-12-16-20-19-18-14	3.8564
8-15	8-10-15	0.8807	8-13-14-17-15	4.7359
8-16	8-10-12-16	1.3592	8-13-14-18-19-20-16	3.4658
8-17	8-13-14-17	1.5285	8-10-15-17	4.0881
8-18	8-13-14-18	1.6824	8-10-12-16-20-19-18	3.1426
8-19	8-13-14-18-19	2.1053	8-10-12-16-20-19	2.7197
8-20	8-10-12-16-20	1.8890	8-13-14-18-19-20	2.9360
9-10	9-13-8-10	1.3279	9-7-3-2-6-12-10	4.9265
9-11	9-5-11	1.1397	9-13-14-11	2.0946
9-12	9-13-8-10-12	1.6037	9-7-3-2-6-12	4.6507
9-13	9-13	0.3685	9-5-11-14-13	2.8658
9-14	9-13-14	0.7624	9-5-11-14	2.4719
9-15	9-13-8-10-15	1.8238	9-5-11-14-17-15	6.2392
9-16	9-13-8-10-12-16	2.3023	9-5-11-14-18-19-20-16	4.9691
9-17	9-13-14-17	1.3223	9-5-11-14-18-19-17	7.0747
9-18	9-13-14-18	1.4762	9-5-11-4-18	3.4976
9-19	9-13-14-18-19	1.8991	9-5-11-14-17-19	6.4978
9-20	9-13-14-18-19-20	2.7298	9-7-3-2-6-12-16-20	5.8791

Tableau AII.1 Détermination du chemin principal et du chemin de secours (suite)

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
10-11	10-8-13-9-5-11	2.4676	10-12-16-20-19-18-14-11	4.8038
10-12	10-12	0.2758	10-8-13-14-18-19-20-16-12	4.5492
10-13	10-8-13	0.9595	10-12-16-20-19-18-14-13	3.8655
10-14	10-8-13-14	1.3534	10-12-16-20-19-18-14	3.4716
10-15	10-15	0.4959	10-8-13-14-17-15	5.1207
10-16	10-12-16	0.9744	10-8-13-14-18-19-20-16	3.8506
10-17	10-8-13-14-17	1.9133	10-15-17	3.7033
10-18	10-8-13-14-18	2.0672	10-12-16-20-19-18	2.7578
10-19	10-12-16-20-19	2.3349	10-8-13-14-18-19	2.4901
10-20	10-12-16-20	1.5042	10-8-13-14-18-19-20	3.3208
11-12	11-5-9-13-8-10-12	2.7434	11-14-18-19-20-16-12	4.5280
11-13	11-5-9-13	1.5082	11-14-13	1.7261
11-14	11-14	1.3322	11-5-9-13-14	1.9021
11-15	11-5-9-13-8-10-15	2.9635	11-14-17-15	5.0995
11-16	11-5-9-13-8-10-12-16	3.4420	11-14-18-19-20-16	3.8294
11-17	11-14-17	1.8921	11-5-9-13-8-10-15-17	6.1709
11-18	11-14-18	2.0461	11-4-18	2.3579
11-19	11-14-18-19	2.4690	11-5-9-13-8-10-12-16-20-19	4.8025
11-20	11-14-18-19-20	3.2996	11-5-9-13-8-10-12-16-20	3.9718
12-13	12-10-8-13	1.2353	12-16-20-19-18-14-13	3.5897
12-14	12-10-8-13-14	1.6292	12-16-20-19-18-14	3.1958
12-15	12-10-15	0.7717	12-16-20-15	4.9037
12-16	12-16	0.6986	12-10-8-13-14-18-19-20-16	4.1264
12-17	12-10-8-13-14-17	2.1891	12-16-20-19-17	5.5250
12-18	12-10-8-13-14-18	2.3430	12-16-20-19-18	2.4819
12-19	12-16-20-19	2.0590	12-10-8-13-14-18-19	2.7659
12-20	12-16-20	1.2284	12-10-8-13-14-18-19-20	3.5966
13-14	13-14	0.3939	13-9-5-11-14	2.8404
13-15	13-8-10-15	1.4553	13-14-17-15	4.1612
13-16	13-8-10-12-16	1.9338	13-14-18-19-20-16	2.8911
13-17	13-14-17	0.9538	13-8-10-15-17	4.6628
13-18	13-14-18	1.1078	13-8-10-12-16-20-19-18	3.7172
13-19	13-14-18-19	1.5307	13-8-10-12-16-20-19	3.2943
13-20	13-14-18-19-20	2.3613	13-8-10-12-16-20	2.4637
14-15	14-13-8-10-15	1.8492	14-17-15	3.7673
14-16	14-13-8-10-12-16	2.3278	14-18-19-20-16	2.4972
14-17	14-17	0.5599	14-18-19-17	4.6028
14-18	14-18	0.7139	14-11-4-18	3.6901

Tableau AII.1 Détermination du chemin principal et du chemin de secours (suite)

Source/ destination	Chemin principal		Chemin de secours	
	Noeuds visités	Délai du chemin (ms)	Noeuds visités	Délai du chemin (ms)
14-19	14-18-19	1.1368	14-13-8-10-12-16-20-19	3.6882
14-20	14-18-19-20	1.9674	14-13-8-10-12-16-20	2.8576
15-16	15-10-12-16	1.4703	15-20-16	4.2051
15-17	15-10-8-13-14-17	2.4091	15-17	3.2074
15-18	15-10-8-13-14-18	2.5631	15-20-19-18	4.9288
15-19	15-10-12-16-20-19	2.8307	15-17-14-18-19	4.9041
15-20	15-10-12-16-20	2.0001	15-20	3.6753
16-17	16-12-10-8-13-14-17	2.8876	16-20-19-17	4.8265
16-18	16-20-19-18	1.7834	16-12-10-8-13-14-18	3.0416
16-19	16-20-19	1.3605	16-12-10-8-13-14-18-19	3.4645
16-20	16-20	0.5298	16-12-10-8-13-14-18-19-20	4.2951
17-18	17-14-18	1.2738	17-19-18	3.8889
17-19	17-14-18-19	1.6966	17-19	3.4660
17-20	17-14-18-19-20	2.5273	17-15-10-12-16-20	5.2075
18-19	18-19	0.4229	18-14-13-8-10-12-16-20-19	4.4021
18-20	18-19-20	1.2535	18-14-13-8-10-12-16-20	3.5715
19-20	19-20	0.8306	19-18-14-13-8-10-12-16-20	3.9943

AII.3 Comportement des liaisons d'un réseau à haut débit en cas de défaillance

Le flot de chaque liaison varie selon la liaison défaillante. Nous présentons les graphiques illustrant la variation de flot sur chacune des liaisons 4 à 30 du réseau de la Figure 4.2. L'axe x traduit l'état du réseau et indique le numéro de la liaison en panne. L'axe y représente la variation relative du flot par rapport au flot à l'état E_0 . Cette variation est déterminée par la relation (4.25).

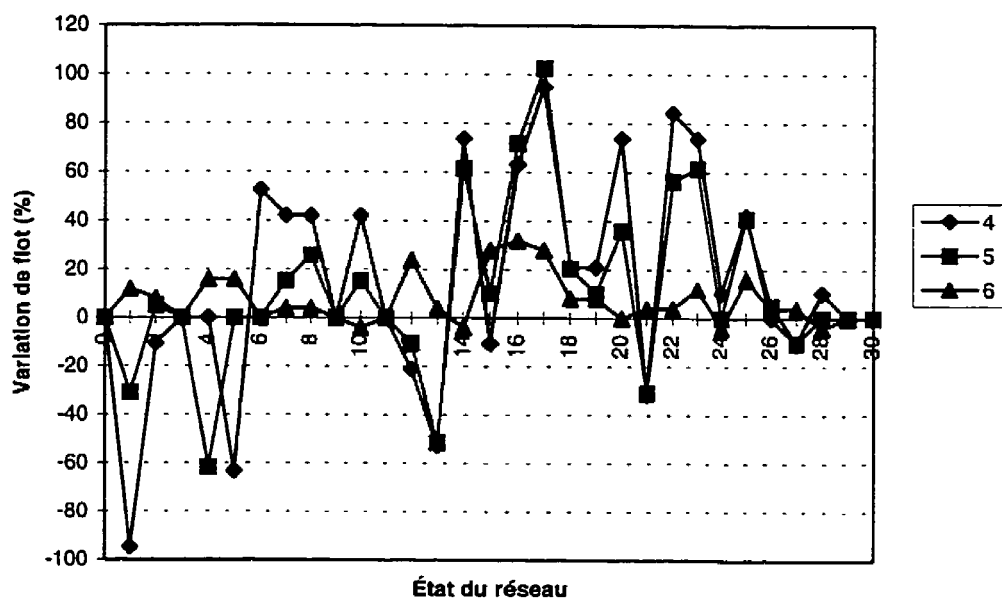


Figure AII.1 Variation relative de flot sur les liaisons 4, 5, 6

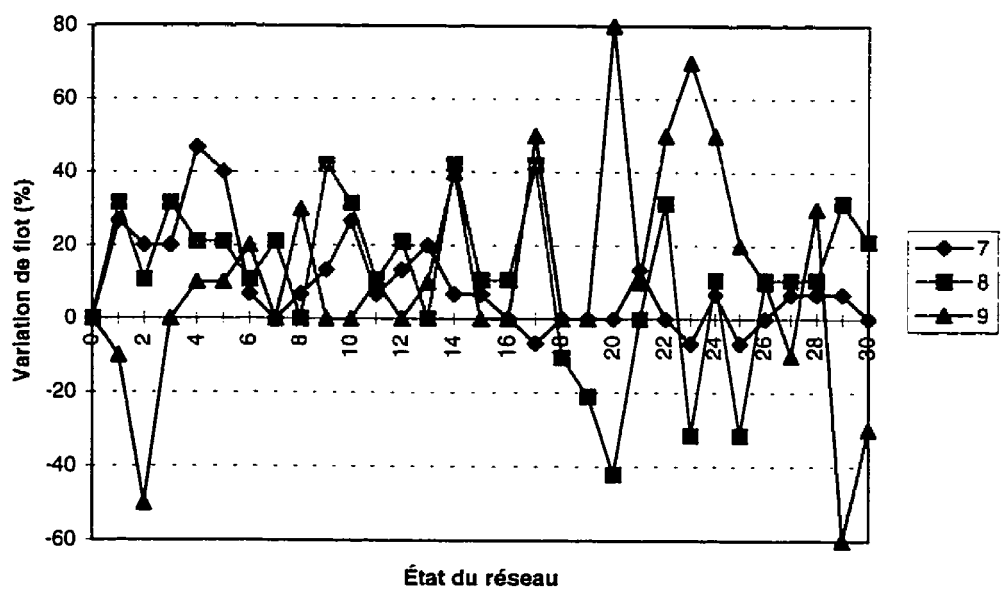


Figure AII.2 Variation relative de flot sur les liaisons 7, 8, 9.

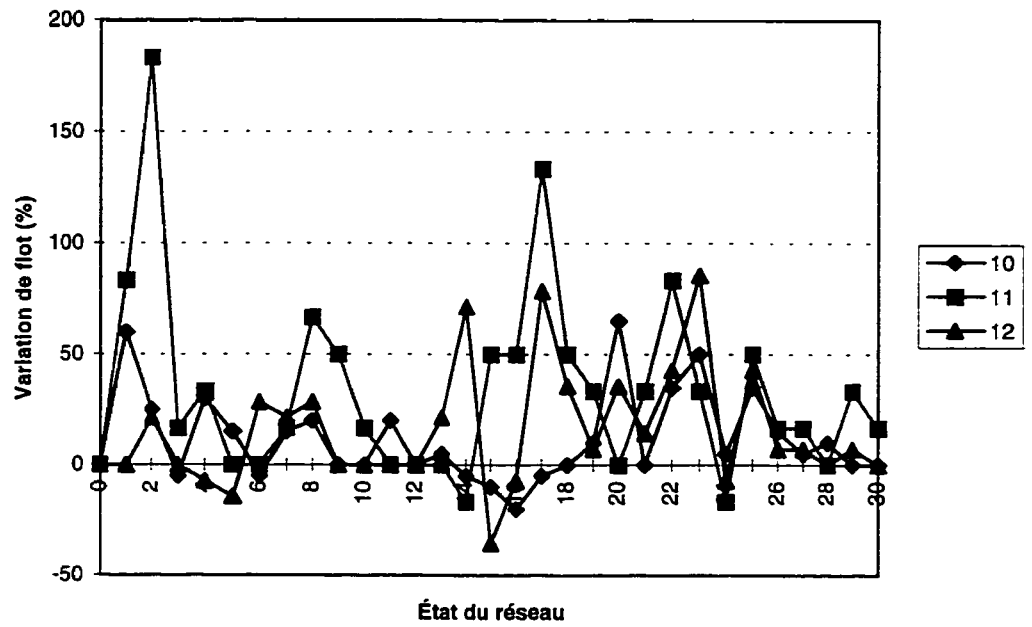


Figure AII.3 Variation relative de flot sur les liaisons 10, 11, 12.

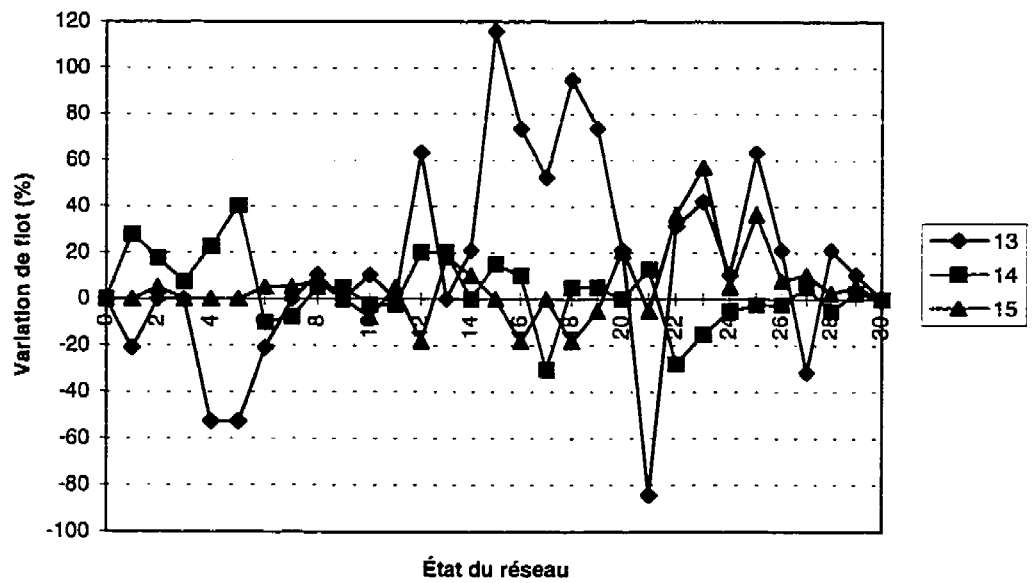


Figure AII.4 Variation relative de flot sur les liaisons 13, 14, 15.

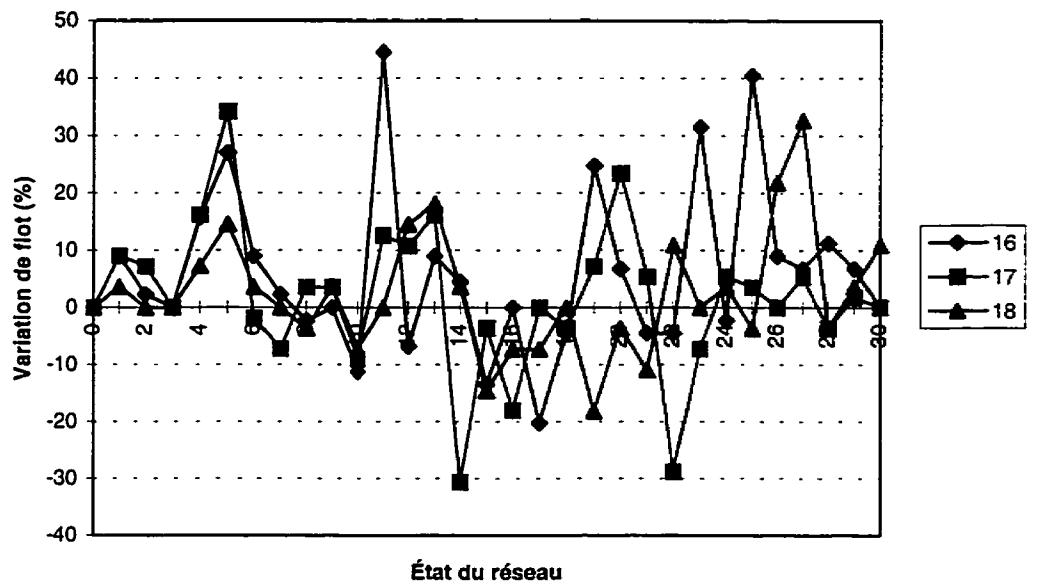


Figure AII.5 Variation relative de flot sur les liaisons 16, 17, 18

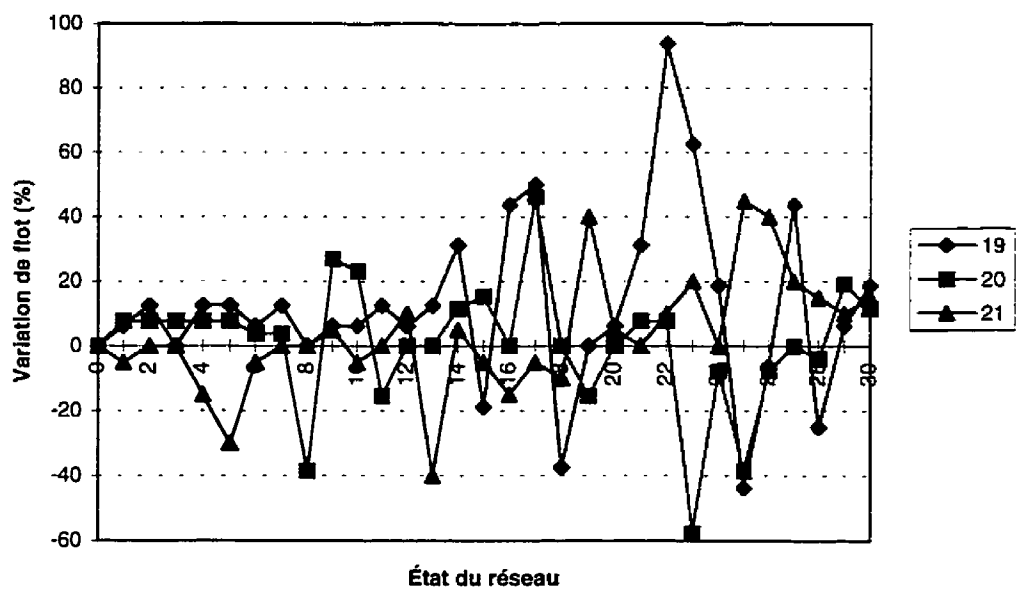


Figure AII.6 Variation relative de flot sur les liaisons 19, 20, 21

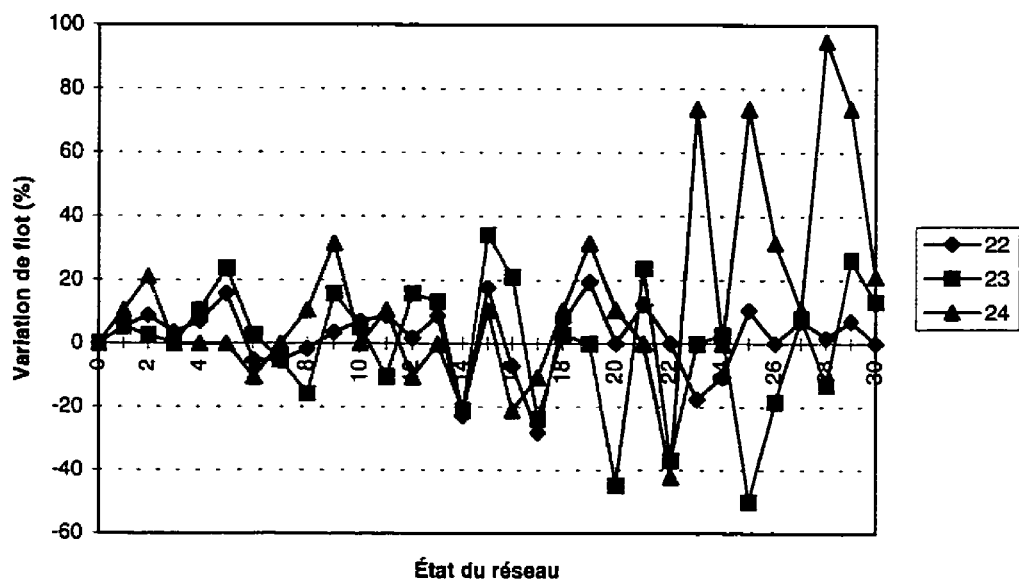


Figure AIL.7 Variation relative de flot sur les liaisons 22, 23, 24

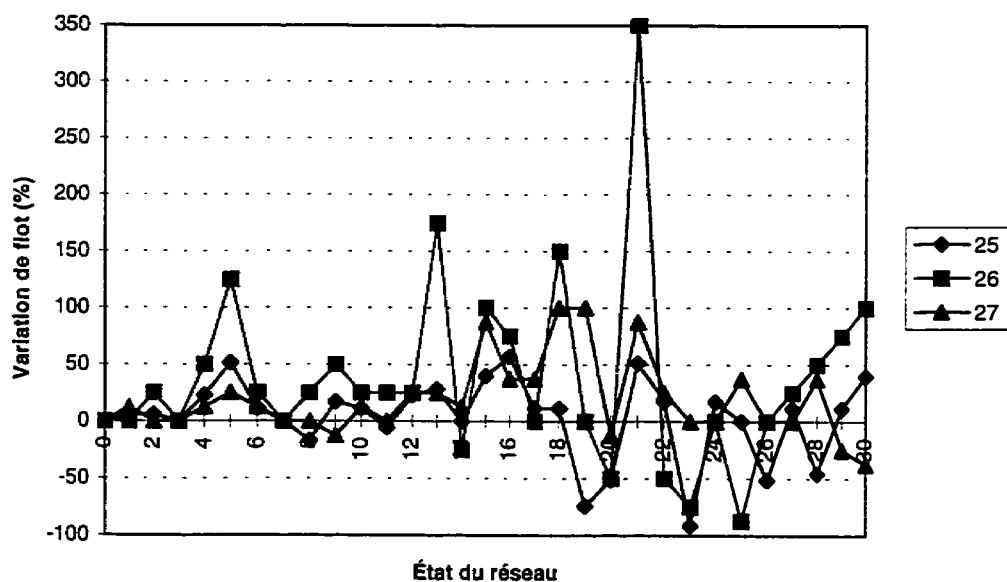


Figure AIL.8 Variation relative de flot sur les liaisons 25, 26, 27

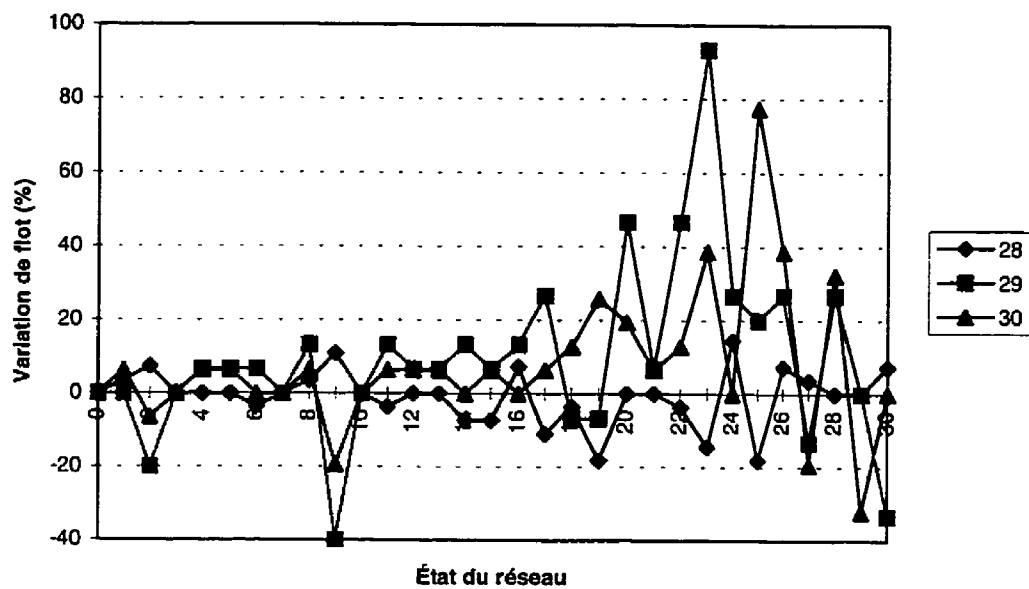
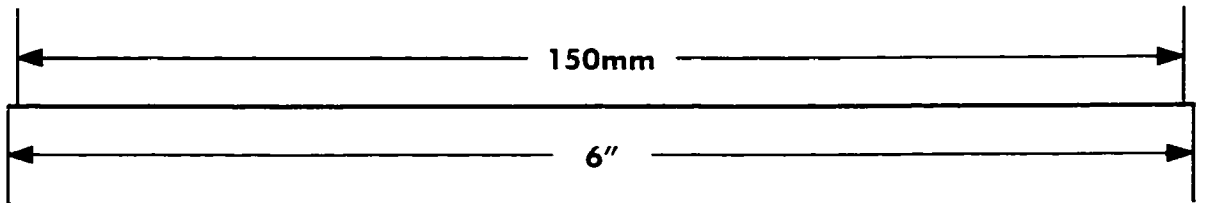
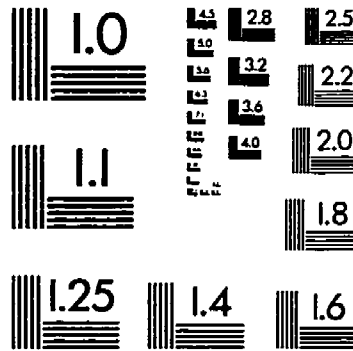
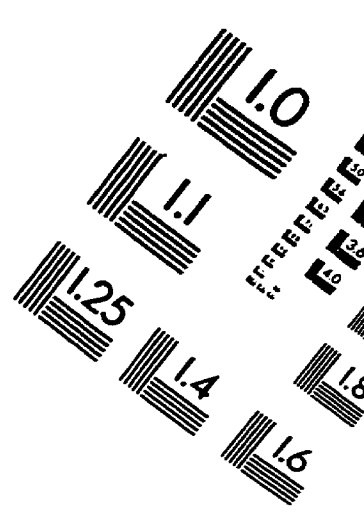
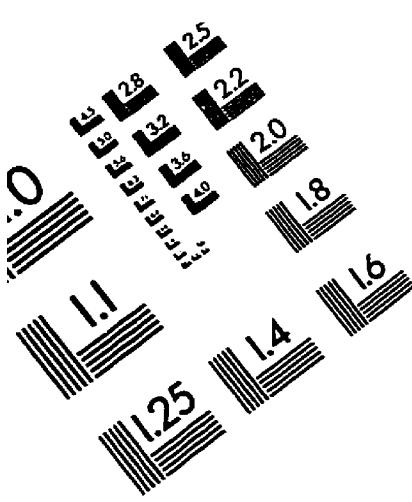


Figure AII.9 Variation relative de flot sur les liaisons 28, 29, 30

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved

