Techniques d'Apprentissage par Renforcement pour le Routage Adaptatif dans les Réseaux de Télécommunication à Trafic Irrégulie

Said Hoceini

THESE

Présentée à
L'UNIVERSITE PARIS XII – VAL DE MARNE
U.F.R des Sciences et Technologies
Par :

Said HOCEINI

Pour obtenir le grade de
DOCTEUR DE L'UNIVERSITE PARIS VAL DE MARNE

Spécialité **: Informatique – Réseaux**

Techniques d'Apprentissage par Renforcement pour le Routage Adaptatif dans les Réseaux de Télécommunication à Trafic Irrégulier

Soutenue le 23 Novembre 2004 devant le jury :

Rapporteurs :

P. GALLINARI	Professeur à l'Université Pierre et Marie Curie (Paris6)
E. RONDEAU	Professeur à l'Université Henri Poincaré (Nancy I)

Examinateurs:

Z .MAMMERI	Professeur à l'Université Paul Sabatier, (Toulous 3)
E. GRESSIER	HDR et Maître de Conférences au CNAM paris
Y. AMIRAT	Professeur à l'Université Paris 12 Val de Marne
A. MELLOUK	Maître de Conférences à l'Université Paris 12 Val de Marne

Remerciements

L'ensemble des travaux présentés dans ce mémoire a été effectué au Laboratoire d'Informatique Industrielle et d'Automatique (L.I.I.A) de l'Université Paris 12 Val de Marne.

J'adresse mes sincères remerciements à Monsieur Zoubir MAMMERI, Professeur à l'Université Paul Sabatier, Toulouse, qui m'a fait l'honneur de présider le jury de thèse.

J'exprime ma profonde reconnaissance à Monsieur Eric RONDEAU, Professeur à l'Université Henri Poincaré - Nancy I, et à Monsieur Patrick GALLINARI, Professeur à l'Université Pierre et Marie Curie, pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant de l'examiner et d'en être rapporteurs.

Je remercie également Monsieur Eric GRESSIER, HDR et maître de conférences au CNAM paris, d'avoir accepté de juger mon travail.

Ma profonde gratitude va à Monsieur Jean PONTNAU, professeur à l'Université Paris Val de Marne et ancien directeur du LI.IA, pour m'avoir accueilli dans son laboratoire, et m'avoir permis de travailler dans de bonnes conditions.

Je tiens aussi à exprimer ma profonde gratitude et mes remerciements les plus sincères à Monsieur Yacine AMIRAT, et à Monsieur Abdelhamid MELLOUK, respectivement Professeur et Maître de Conférences à l'Université Paris Val de Marne, pour m'avoir encadré et pour la confiance qu'ils m'ont toujours, témoigné. Leurs expériences de la recherche, leurs conseils, et leurs encouragements m'ont été très précieux et m'ont permis de mener à bien mon travail.

Je tiens également à remercier l'ensemble des membres du LI.IA, qui resteront anonymes dans cette page, mais qui m'ont permis de mener à terme ce travail dans une ambiance très amicale.

Dédicaces

Je dédie cette thèse

à la mémoire de mon petit frère Ali Sofiane

à la mémoire de mes grands-parents

à mes parents et mon frère, à qui je dois beaucoup

à Mr et madame Ouahioune, à qui je souhaite un bon rétablissement

à tous les miens

Tables des Matières

Tables des Figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Chapitre V

Introduction Générale

Depuis quelques années, nous assistons à un développement rapide des applications communicantes. Outre celles qui ont contribué à la popularité de l'Internet à ses débuts (messagerie électronique, transfert de fichiers, etc.), on trouve désormais des applications qui confient au réseau des données plus sensibles. Parmi celles-ci, on peut citer les applications permettant de faire transiter du son (voix, programmes musicaux, etc.), de l'image (programmes de télévision, vidéoconférence, vidéo à la demande, etc.) ou des informations urgentes (ordres de bourse). Ainsi, l'utilisation des nouvelles générations de réseau dans le cadre d'applications multimédia ou de services à qualité garantie, à diffusion, mobiles, etc. impose que l'acheminement soit assuré avec une Qualité de Service (QoS) maîtrisée. Cette dernière comporte une série de paramètres qui permettent de caractériser les garanties qui peuvent être fournies à chaque flux (ou connexion) transitant par le réseau. Ces paramètres sont généralement la bande passante, le délai de bout en bout, la gigue ou le taux de perte des données.

Le problème de l'intégration de la qualité de service dans les réseaux constitue un vaste sujet de recherche et a fait l'objet de nombreuses techniques proposées dans la littérature. Celles-ci interviennent à différents niveaux. Le premier concerne les extrémités d'un lien de communication avec comme exemples le démarrage lent et le contrôle de débit dans le protocole TCP. Le deuxième niveau concerne quant à lui les nœuds intermédiaires du réseau. Ainsi, des techniques de prévention de congestion (RED, WRED) et des solutions de gestion explicite de la Qualité de service réseau (IntServ/RSVP, DiffServ) ont été proposées. Même si ces techniques apportent des éléments de solution, elles ne garantissent pas à elles seules la QoS de bout en bout et doivent être complétées par des techniques de routage adaptatif.

L'objectif de ce travail de thèse est de proposer des approches algorithmiques permettant de traiter la problématique du routage adaptatif dans un réseau de communication à trafic irrégulier de type IP. Ce type de réseau est caractérisé par l'hétérogénéité de ses liens, ainsi que par des conditions de trafic dynamiques, ce qui nécessite de prendre en compte la QoS au niveau du routage. Par conséquent, toute approche de routage adaptatif doit être suffisamment réactive et robuste pour prendre en compte toute modification des conditions de trafic tout en minimisant le temps d'acheminement de bout en bout. Ces dernières années, des approches de routage basées sur l'apprentissage par renforcement ont été proposées. Ce type d'apprentissage est bien adapté à la problématique du routage puisque le modèle représentant l'environnement (le réseau) dans lequel se situe le routeur est a priori inconnu.

Cependant, l'efficacité de ces approches dépend fortement des informations sur la charge du réseau. Ces informations doivent être suffisantes et pertinentes et en même temps refléter de manière fiable la charge réelle du réseau au moment de la prise de décision de routage. L'objectif de ce travail de thèse est de proposer deux nouvelles approches algorithmiques améliorant les techniques de routage adaptatif basé sur l'apprentissage par renforcement. Ces approches permettent de remédier aux inconvénients des techniques utilisant le Q-Routing des points de vue de la rapidité de convergence vers la solution optimale lorsque le réseau est soumis à un trafic élevé et de la réduction de l'espace mémoire occupé par la table de routage.

Le premier algorithme appelé Q-Neural Routing, est basé sur une approche neuronale permettant une modélisation de la dynamique du réseau. Cette méthode permet un apprentissage discriminant et une prise en compte fiable du contexte dans lequel se trouve le réseau, pour l'estimation des temps d'acheminement des paquets. L'algorithme proposé a pour objectif d'une part, de prendre en compte en plus du temps de bout en bout, d'autres paramètres tels que l'état des files d'attentes ou la nature des flux et d'autre part, de minimiser l'espace mémoire occupé par la table de routage.

Le second algorithme appelé K-Shortest paths Q-Routing, combine la technique de recherche des K plus courts chemins et l'algorithme Q-Routing. En plus de la minimisation de l'espace mémoire occupé par la table de routage, un des objectifs visés est de réduire l'espace d'exploration aux K plus courts chemins afin de minimiser le temps de convergence. Il s'agit également d'introduire dans la prise de décision d'autres paramètres tels que le nombre de sauts ou la bande passante.

Dans le but d'aborder ces différents aspects, nous avons divisé ce mémoire en cinq chapitres.

Le premier chapitre a pour objectif de présenter la problématique générale de la qualité de service. Après une brève introduction aux réseaux de télécommunication, nous décrivons les paramètres caractérisant la QoS. Nous présentons et analysons ensuite les différentes techniques permettant d'intégrer cette dernière dans les réseaux IP.

Le second chapitre est consacré à la présentation et à l'analyse des algorithmes prenant en compte des contraintes de qualité de service dans le routage. Après une brève description de la fonction de routage et des algorithmes classiques implémentant celle-ci, nous présentons

les différentes approches de routage avec qualité de service proposées dans la littérature. L'analyse de ces travaux nous permet d'une part, de retenir le principe du routage adaptatif basé sur l'apprentissage comme base de travail et d'autre part, de proposer deux nouvelles approches algorithmiques permettant d'améliorer les solutions actuelles des points de vue du temps de convergence et de l'occupation mémoire.

Dans le troisième chapitre, nous présentons les concepts et outils mathématiques sur lesquels nous nous sommes appuyés pour développer les approches de routage adaptatif que nous proposons. Nous développons tout d'abord le modèle connexionniste puis les différentes méthodes d'apprentissage. Notre objectif est de définir la méthode d'apprentissage qui répond le mieux à la problématique du routage adaptatif à QoS, pour un réseau à trafic irrégulier.

Dans le quatrième chapitre, nous présentons le premier algorithme de routage adaptatif, appelé Q-Neural Routing, dérivé de l'algorithme Q-Routing. Nous décrivons en particulier le mécanisme d'exploration utilisé pour la mise à jour des paramètres de routage ainsi que le modèle neuronal utilisé pour l'estimation des Q-valeurs. Nous présentons et analysons ensuite les performances de l'algorithme proposé en termes de délai moyen d'acheminement des paquets, pour différentes conditions de trafic ainsi que son comportement face à un changement de topologie. Pour cette validation, nous nous appuyons sur l'outil de simulation OPNET. Nous développons en particulier l'implémentation de l'algorithme Q-Routing sur lequel s'appuient nos deux approches algorithmiques.

Le cinquième chapitre est consacré au deuxième algorithme proposé, appelé K-Shortest path Q-Routing. Ce dernier est basé sur la technique du routage multi chemin combiné avec l'algorithme Q-Routing. Pour ce faire, nous nous appuyons sur l'algorithme de Dijkstra généralisé, auquel on adjoint un mécanisme de suppression de boucles. A partir d'un mécanisme d'exploration hybride, nous formulons l'algorithme de mise à jour des paramètres de routage. Les performances de l'algorithme sont ensuite évaluées et comparées à celles des algorithmes RIP, Q-Routing et K-Shortest path Routing.

Enfin, une conclusion générale et quelques perspectives de recherche sont exposées dans la dernière partie du manuscrit.

Chapitre I

Réseaux et Qualité de Service

I.1 Introduction

L'objectif de ce chapitre est de présenter la problématique générale de la qualité de service dans les réseaux de télécommunication. Après une brève introduction aux réseaux de télécommunication, nous décrivons les paramètres caractérisant la QoS. Nous présentons et analysons ensuite les différentes techniques permettant d'intégrer cette dernière dans les réseaux IP.

I.2 Introduction aux réseaux de télécommunication

Au début des années 70, chaque constructeur a développé sa propre solution réseau autour d'architectures et de protocoles privés (SNA d'IBM [CIS], DECnet de DEC[CIS], DSA de Bull, TCP/IP du DoD,...) et il s'est vite avéré qu'il serait impossible d'interconnecter ces différents réseaux si une norme internationale n'était pas établie. Cette norme a été introduite par l'ISO [ISO], (*International Standard Organization*), il s'agit du modèle OSI [OSI94] [CAS93], *(Open System Interconnection*). Ce dernier décrit la manière dont, matériels et logiciels coopèrent selon une architecture en couches qui permet la communication. Ce modèle constitue également une aide pour le dépannage, car il fournit un cadre de référence qui décrit la façon dont les composants sont censés fonctionner. Le modèle OSI comporte sept couches (Figure I.1). Il s'étend des niveaux les plus bas des techniques de transmission jusqu'aux interactions de haut niveau entre applications spécifiques. Ces sept couches sont:

- **La couche physique**: Elle établit la connexion physique entre un système et le réseau, et concerne la manière dont le message est transporté en fonction du support (câbles, ondes, etc.). A ce niveau, l'unité d'information est le **Bit**.

La couche de liaison de données : Cette couche "fait" et "défait" les paquets de données à transmettre. Elle assure le contrôle et la correction des données, le contrôle des accès vers la carte réseau. Sur cette couche, le message est appelé une **trame**.

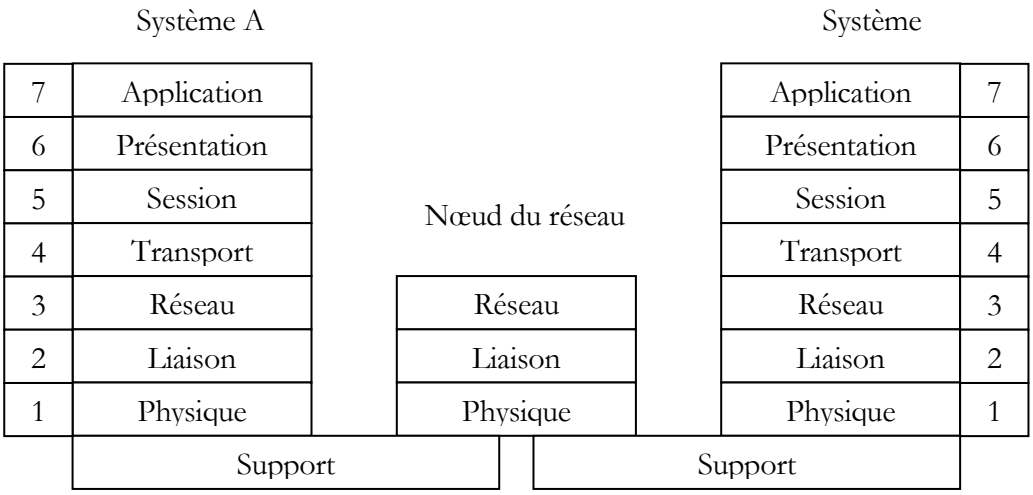


Figure I.1: Le modèle OSI : l'interconnexion de systèmes ouverts

- **La couche réseau**: Elle assure le routage des paquets et gère la relation entre les adresses logiques utilisées par l'utilisateur et les adresses physiques implantées dans les machines. Elle se charge aussi des problèmes issus des éventuelles incompatibilités entre les couches basses des réseaux traversés (taille de trame, adressage physique différent, etc.).

- **La couche transport**: Elle reçoit les données de la couche session et assure leur transport de bout en bout. Elle peut éventuellement fragmenter les données en plusieurs paquets afin de faciliter leur transmission et s'assure de leurs arrivées à destination. Pour améliorer la rapidité de la transmission, la couche transport peut créer plusieurs connexions simultanées sur lesquelles elle repartit les paquets (principe de multiplexage). Elle assure la transparence des réseaux traversés vis à vis des couches hautes.

- **La couche session**: Il s'agit de la première couche qui s'intéresse à la communication proprement dite. Elle ouvre une session avec la station de destination. Cette session reste ouverte tant que dure la communication. Lorsque la station de destination envoie un accusé de réception global, il y a fermeture de la session. La couche offre des services de synchronisation et de rattrapage d'erreurs.

- **Couche présentation**: Celle-ci traite de la mise en forme de l'information. Au-dessus d'elle, l'information est codée sous une forme plus appropriée pour son

transport. Ainsi, la compression des données et le cryptage sont du ressort de la couche 6.

- **La couche application**: Elle est chargée d'offrir à l'utilisateur les fonctions de communication. Ces fonctions sont le transfert de fichier, la messagerie, l'émulation de terminal virtuel, l'exécution de travaux à distance, etc. La couche 7 représente l'interface utilisateur pour les fonctions de communication.

Les couches 1 à 4 sont appelées les couches de communication, tandis que les couches 5 à 7 sont les couches de service.

Pour chaque couche, les fonctionnalités sont assurées par des protocoles. Par exemple, dans l'Internet, on peut trouver le protocole IP au niveau de la couche 3, et le protocole TCP au niveau de la couche 4. L'architecture du réseau réside donc dans la fonctionnalité des protocoles.

Dans la couche réseau, le protocole IP (Internet Protocol) [POS81] est au cœur du fonctionnement des réseaux IP. Les informations devant être transmises d'un point à un autre du réseau, sont fragmentées en paquets (datagramme IP (Annexe 1)). IP est chargé d'acheminer ces paquets jusqu'à leur destination.

IP assure un service *non fiable* car il n'existe aucune garantie pour que les datagrammes IP arrivent à destination. Certains peuvent être perdus, dupliqués, retardés, altérés ou remis dans le désordre. On parle alors de remise au mieux (*best effort delivery*) et ni l'émetteur ni le récepteur ne sont informés directement par IP des problèmes rencontrés. Le mode de transmission est non connecté car IP traite chaque datagramme indépendamment de ceux qui le précèdent et le suivent. Ainsi en théorie, au moins, deux datagrammes IP issus de la même machine et ayant la même destination, peuvent ne pas suivre obligatoirement le même chemin. C'est donc IP qui conditionne le chemin parcouru par un paquet de sa source jusqu'à sa destination. Cette fonctionnalité s'appelle le routage.

Depuis quelques années, nous assistons à un développement rapide des applications communicantes. Outre celles qui ont contribué à la popularité de l'Internet à ses débuts (messagerie électronique, transfert de fichiers, etc.), on trouve désormais des applications qui confient au réseau des données plus sensibles. Parmi celles-ci, on peut citer les applications

permettant de faire transiter du son (voix, programmes musicaux, etc.), de l'image (programmes de télévision, vidéoconférence, vidéo à la demande, etc.) ou des informations urgentes (ordres de bourse). Les perspectives de ces nouvelles applications sont riches et nombreuses. On peut citer à titre d'exemple, les recherches de la médecine en matière de télé chirurgie, où il est question de commander des bras articulés à distance afin d'effectuer des opérations chirurgicales. La fragilité des données échangées par ces applications peut s'exprimer de différentes manières :

- **L'intégrité de l'ensemble :** La perte d'images lors de la transmission d'un flux vidéo dégrade la visualisation.
- **Le temps de réponse :** Un temps de transit dans le réseau qui serait trop grand peut rendre une information urgente obsolète ou une conversation incompréhensible.
- **La régularité :** La qualité de réception doit être constante.

Par conséquent, suivant la nature des informations transmises, le réseau doit pouvoir apporter une qualité de service appropriée, que l'on peut exprimer sous la forme d'une liste de contraintes.

I.3 Qualité de service dans les réseaux

I.3.1 Paramètres de la Qualité de Service

La QoS comporte une série de paramètres qui permettent de caractériser les garanties qui peuvent être fournies à chaque flux (ou connexion) transitant par un réseau de télécommunication. Ces paramètres sont généralement la bande passante, le délai de bout en bout, la gigue, la perte de données.

I.3.1.1 La bande passante

La bande passante est la quantité maximale de données pouvant être transmise d'une source vers une destination en une unité de temps. Il s'agit en fait du minimum des bandes passantes disponibles sur les liens composant le chemin de la source à la destination. La

bande passante disponible sur un chemin dépend donc du support des liaisons, mais aussi du nombre et du débit des flux qui partagent ces liaisons.

A titre d'exemple, les applications vidéo requièrent généralement une bande passante élevée car d'une part, une image représente une grande quantité d'informations et d'autre part, le nombre d'images envoyées par unité de temps doit être suffisant pour obtenir une visualisation satisfaisante.

I.3.1.2 Le délai de bout en bout

Appelé aussi latence ou temps de réponse, il s'agit de la durée nécessaire à l'acheminement d'un paquet de données de bout en bout. Cette durée dépend de la qualité du support des liaisons, mais aussi du temps passé dans les différentes files d'attente du chemin. Par conséquent, le délai augmente avec la charge du réseau. Par exemple, les applications de conversation (voix sur IP) sont particulièrement sensibles au délai. On estime qu'une conversation devient désagréable si le temps d'acheminement dépasse 28 ms[PUJ02].

I.3.1.3 La gigue

La gigue est la variation du délai de bout en bout, issue des congestions instantanées lorsque plusieurs flux de données sollicitent au même moment le même port de sortie. Un bon contrôle de la gigue est souhaité dans les applications de type vidéo à la demande, car une forte gigue entraîne des distorsions lors de l'écoute ou de la visualisation des médias.

I.3.1.4 La perte de données

Le taux de perte est la proportion des paquets qui ne parviennent pas à leur destination. Ces pertes dépendent de :

- La fiabilité du support des liaisons : un support peu fiable entraîne une corruption fréquente des paquets. Si un paquet s'avère corrompu à l'issue de sa vérification grâce aux bits de contrôle (checksum), il sera détruit. On peut toutefois estimer qu'actuellement la fiabilité des liens dans le réseau Internet est relativement élevée, et que par conséquent, les pertes de paquets dues à des défauts de support sont très faibles.

- L'occurrence de surcharges locales (congestions) dans le réseau : les paquets devant être placés dans une file d'attente pleine sont détruits.

Lorsque le protocole de transport garantit l'arrivée de l'information transmise, les paquets perdus doivent être renvoyés. Les pertes ont donc une influence directe sur l'augmentation du délai et sur la gigue.

I.3.2 Intégration de la QoS dans les réseaux

Le problème de la qualité de service dans les réseaux constitue un vaste sujet de recherche et a fait l'objet de nombreuses techniques proposées dans la littérature. En dehors des techniques tendant à organiser l'architecture de communication de façon à réduire les délais d'acheminement des messages tout en garantissant une bonne disponibilité du support de transmission (en travaillant notamment sur le déploiement du réseau en utilisant des algorithmes de placement appropriés [RON01] ou la prise en compte de la QoS au niveau de la couche liaison), les autres techniques, qui vont par ailleurs constituer notre centre d'intérêt, sont classées dans les catégories suivantes : Le contrôle de congestion, le contrôle de débit, la réservation de ressources, la différenciation de services et l'intégration de la QoS dans les décisions de routage des paquets.

I.3.2.1 Le contrôle de congestion

Dans les réseaux de télécommunication, les protocoles de routage généralement utilisés ne tiennent pas compte de la charge du réseau. Par conséquent, il arrive que le débit des flux partageant une liaison soit trop grand, entraînant une surcharge locale du réseau. Ces surcharges sont appelées congestions.

I.3.2.1.1 Définition de la congestion

Lorsqu'un paquet arrive dans un routeur, il est placé dans une file d'attente. Si le débit des paquets en entrée d'un routeur est plus grand que le débit de sortie, le nombre de paquets dans la file d'attente augmente. La taille des files étant limitée, une congestion survient lorsque la file est pleine. Les paquets devant être placés dans la file pleine sont alors détruits. Il est donc essentiel de limiter d'une part, l'apparition de congestions et d'autre part, leur durée. De nombreux travaux se sont orientés dans ce sens, notamment ceux de Van

Jacobson [JAC88], [JAC99]. Ils ont donné naissance à un certain nombre de techniques permettant de moduler le débit des sources, et ainsi de limiter l'aspect néfaste des congestions [STE97b]. Nous présentons ces techniques dans les paragraphes suivants.

I.3.2.1.2 Le démarrage lent (Slow Start)

Dans sa version originale [DAR81], le protocole TCP permet d'envoyer dès le début de la transmission, un nombre de paquets correspondant à la taille de sa fenêtre d'émission. La taille de cette fenêtre est déterminée à l'issue d'une négociation entre la source et la destination des données. [JAC88] montre que cela peut être la cause de congestions dans le réseau. Van Jacobson propose donc d'utiliser une fenêtre supplémentaire, **la fenêtre de congestion**. Lorsqu'une nouvelle connexion est établie, la taille de cette fenêtre (cwnd) est initialisée à un segment. Chaque fois qu'un acquittement est reçu par la source, la taille de la fenêtre de congestion est incrémentée d'un segment. Le nombre de segments transmis (taille de la fenêtre d'émission), est borné par le minimum de la taille de la fenêtre de réception de la destination et de la fenêtre de congestion. Ainsi, la source commence par envoyer un segment, et attend son acquittement. Lorsque l'acquittement est reçu, la fenêtre de congestion est incrémentée d'un segment. La source émet donc deux segments. Pour chacun des deux acquittements reçus pour ces segments, la fenêtre de congestion est à nouveau incrémentée d'un segment, c'est-à-dire passe à 4 segments, et ainsi de suite (Figure I.2).

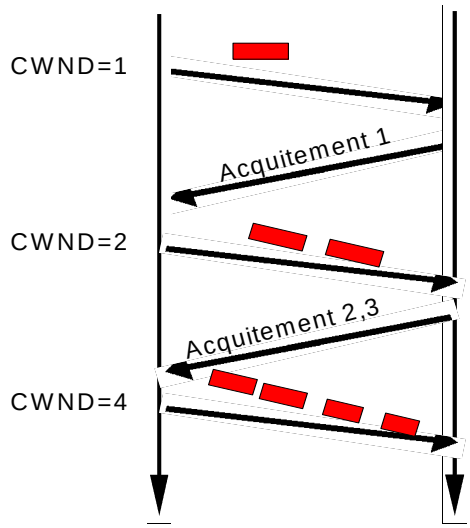


Figure I.2 : Démarrage lent

La taille de la fenêtre de congestion augmente ainsi exponentiellement. Lorsqu'un paquet est perdu, cela signifie que la fenêtre de congestion est devenue trop grande, on peut alors faire appel à la technique de l'évitement de congestion.

I.3.2.1.3 L'évitement de congestion (Congestion Avoidance)

La source détecte qu'un paquet a été perdu lorsqu'elle reçoit des acquittements dupliqués ou bien si elle ne reçoit pas l'acquittement d'un paquet après une période donnée (*time-out*). Le nombre de paquets corrompus à cause d'un défaut de liaison est très faible. On peut considérer que la perte d'un paquet est le signe d'une congestion dans le réseau. Dans ce cas, l'évitement de congestion permet de réduire le débit de la source, en réduisant la taille de sa fenêtre. Pour cela, on applique un seuil au démarrage lent (Slow Start Threshold, *ssthresh*). Lorsqu'une perte de paquet est détectée, on affecte à *ssthresh* une valeur égale à la moitié de la taille de la fenêtre d'émission, et on réduit la taille de la fenêtre de congestion (cwnd) comme suit:

- Si la perte de paquet a été détectée grâce à des acquittements dupliqués, cwnd=ssthresh;
- Si la perte de paquet a été détectée après un *time-out*, cwnd=1.

Ensuite, l'augmentation de la fenêtre de congestion peut être de deux natures :

- Si cwnd<ssthresh, on applique le démarrage lent. Par conséquent, la taille de la fenêtre de congestion augmente exponentiellement ;
- Si cwnd>ssthresh, on applique l'évitement de congestion. A la réception de chaque acquittement, on augmente cwnd d'une taille fixée. La taille de la fenêtre de congestion augmente donc linéairement.

Il est à noter que le démarrage lent et l'évitement de congestion sont actuellement implémentés dans le réseau Internet, dans le protocole TCP.

Cependant, ces deux techniques ne permettent la réduction du débit des sources que lorsqu'une congestion a débuté. L'apparition de congestions n'est donc pas écartée. C'est

pourquoi des algorithmes permettant la prévention des congestions, tels que RED et WRED, ont été proposés.

I.3.2.1.4 RED (Random Early Detection) et WRED (Weighted Random Early Detection)

Proposé en 1993 par Sally Floyd et Van Jacobson [FLO93], le mécanisme RED est une technique préventive permettant d'éviter les congestions. Son principe consiste à détruire arbitrairement des paquets dans les files d'attente, lorsque celles-ci sont remplies au-delà d'un certain seuil. Plus la file se remplit, plus le nombre de flux concernés par la destruction arbitraire de paquets est grand. Grâce au mécanisme d'évitement de congestion, les flux transportés par les paquets détruits voient leur débit réduit.

RED est équitable, puisque les flux qui subissent des destructions de paquets sont choisis arbitrairement. Au contraire, pour privilégier certains flux par rapport à d'autres, on peut appliquer différentes probabilités de destruction. C'est le but des techniques Weighted RED (WRED) et Enhanced RED [FEN97], où les paquets des flux privilégiés, donc plus prioritaires, sont moins exposés à la destruction arbitraire. WRED distingue la priorité des paquets grâce au champ de priorité présent dans chaque en-tête.

Les techniques de régulation des sources que nous avons présentées ci-dessus s'appliquent aux flux individuellement. Une autre démarche consiste à imposer plus généralement des limitations de débit à chaque routeur.

I.3.2.2 Le contrôle de débit

Le contrôle de débit (**traffic shaping**), est une approche globale. Son principe est de limiter le débit des routeurs d'accès, et donc le nombre de paquets injectés dans le réseau, afin de ne pas dépasser un certain seuil en termes d'utilisation des ressources disponibles. Tous les flux ne sont pas sujets à l'évitement de congestion. En effet, si le débit des flux TCP est sensible à la charge du réseau, ce n'est pas le cas pour les flux UDP qui sont émis avec un débit qui peut être très grand, quel que soit l'état du réseau. L'aspect global du contrôle de débit permet, en limitant le débit total des routeurs, d'imposer des limites à ces flux indésirables. Le Leacky Bucket est l'une des techniques permettant d'appliquer le contrôle de débit.

I.3.2.2.1 Le Leacky Bucket

Le Leacky Bucket [TUR86], « seau troué », permet de réguler le trafic émis par les routeurs dans le réseau. Ainsi les paquets devant être émis, sont d'abord placés dans une file d'attente puis envoyés à intervalles réguliers (Figure I.3). Cette méthode permet donc d'éviter les émissions en rafale.

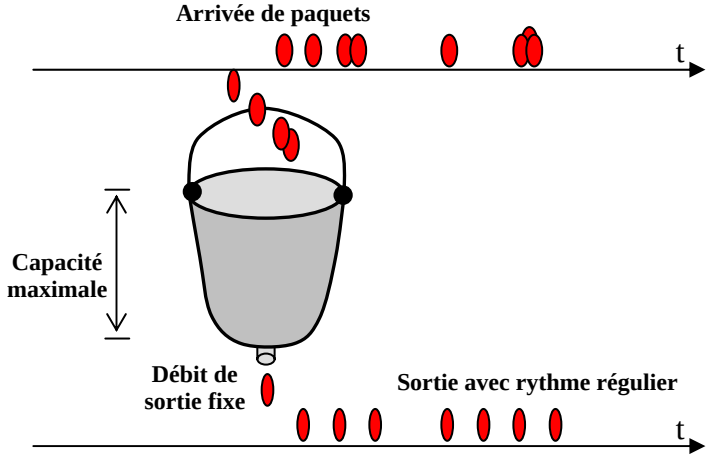


Figure I.3 : Fonctionnement du leacky Bucket

La taille de la file d'attente et le débit d'émission sont configurables. Toutefois, cette méthode ne permet pas de s'adapter à la charge du réseau : le débit d'émission reste constant quelle que soit la charge. Une technique dérivée permet au contraire de tenir compte des ressources disponibles dans le réseau : le Token Bucket.

I.3.2.2.2 Le Token Bucket

Pour une meilleure gestion des ressources, il est préférable de moduler le débit des routeurs en fonction de la charge du réseau. Le Token Bucket [SHE97], "seau à jetons", utilise la notion de jeton pour évaluer la charge du réseau. Le nombre de jetons disponibles est inversement proportionnel à la charge du réseau. Le débit du routeur est conditionné par le nombre de jetons disponibles. Ainsi, le Token Bucket permet l'émission en rafale lorsque le réseau est peu chargé (Figure I.4).

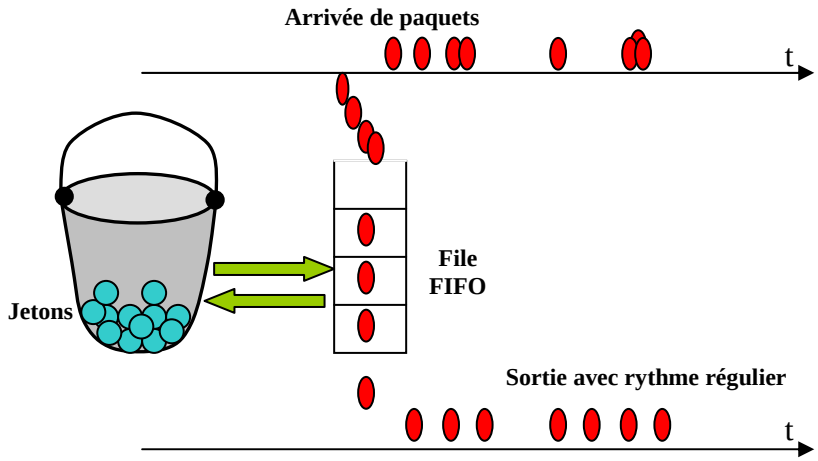


Figure I.4 : Fonctionnement du token Bucket

Un autre moyen d'améliorer la qualité de service dans l'Internet est la réservation des ressources nécessaires à la transmission des informations.

I.3.2.3 La réservation de ressources

I.3.2.3.1 Architecture INTSERV (integrated services)

L'architecture Intserv s'organise autour du concept de flot de données correspondant à un ensemble de paquets résultant d'une application utilisatrice et ayant besoin d'une certaine QoS. Afin de satisfaire la QoS requise, Intserv propose d'effectuer une réservation des ressources nécessaires à l'établissement de celle-ci via le protocole de réservation de ressources nommé RSVP. Le signal RSVP incluant l'information de contrôle de la QoS, propose des directives afin de mettre en place la réservation de ressources mais ne dit pas comment le faire. Ce domaine est réservé aux routeurs du réseau qui prennent en compte la signalisation RSVP. Pour ce faire, les routeurs disposent de quatre fonctions de contrôle du trafic :

1. Le protocole de réservation de ressource qui, de façon implicite, signale le chemin à établir en sollicitant des réservations de bande passante sur chaque routeur traversé.

2. Le contrôle d'admission permet d'autoriser l'arrivée d'un nouveau flot muni de sa QoS sans perturber les QoS des flux existants.

3. Les classifieurs classent les paquets de flux admis dans des classes spécifiques.

4. L'ordonnanceur de paquets détermine l'ordre de service des paquets.

Ainsi RSVP va maintenir un chemin dynamique à l'intérieur du réseau, qu'il rafraîchi par des messages périodiques stipulant l'état du chemin au travers des routeurs.

I.3.2.3.2 RSVP (Resource reSerVation Protocol)

RSVP a été présenté en 1995 à Interop [INT95], une démonstration mettant en oeuvre des ordinateurs recevant du son et des images a été effectuée. Il a été prouvé qu'à travers un réseau Internet, même chargé, la qualité de la transmission de la vidéo et du son était bonne lorsque les données étaient acheminées via des sessions RSVP. Sans les mécanismes de réservation fournis par cette technique, les présentations devenaient inintelligibles. Cette performance était due à RSVP associé aux mécanismes d'ordonnancement des flux implantés dans les routeurs, c'est à dire, Weight Fair Queuing chez Cisco Systems, et Class Based Queuing chez Bay Networks.

L'IETF a conçu le protocole RSVP pour la réservation de ressources au sein d'un réseau Internet. Il peut être utilisé pour assurer la qualité de service et gérer les ressources de transport du réseau pour les sessions point à point (unicast) et point à multipoint (multicast). RSVP est un système de contrôle et de signalisation qui donne la possibilité de réserver la bande passante nécessaire au bon fonctionnement d'une application. Il s'agit d'un besoin qui touche principalement les flux multimédias, plus sensibles aux aléas de l'acheminement que les flux de données pures du fait de leurs contraintes temporelles.

RSVP est basé sur le concept de session [BRA97]. Une session est composée d'au moins un flux de données et est définie par rapport à une "destination" (ou plus précisément par le triplet {adresse de destination, port de destination, identification du protocole}). Comme l'adresse de destination peut être une adresse de groupe, la destination associée à une session peut donc être un groupe de receveurs aussi bien qu'un receveur unique.

Dans RSVP, un flux est défini comme étant n'importe quel sous-ensemble de paquets d'une session ou, en d'autres termes, un sous-ensemble des paquets envoyés à une destination donnée. Un flux est donc unidirectionnel. Théoriquement, le sous-ensemble de

paquets formant un flux peut être arbitraire, mais dans l'état actuel de la spécification de RSVP, un flux est défini comme l'ensemble des paquets émanant d'une même "source" (identifiée par la paire {adresse d'expéditeur, port d'expéditeur}). Une session RSVP fonctionne comme suit [BRA97] [ZHA93] (Figure I.5) :

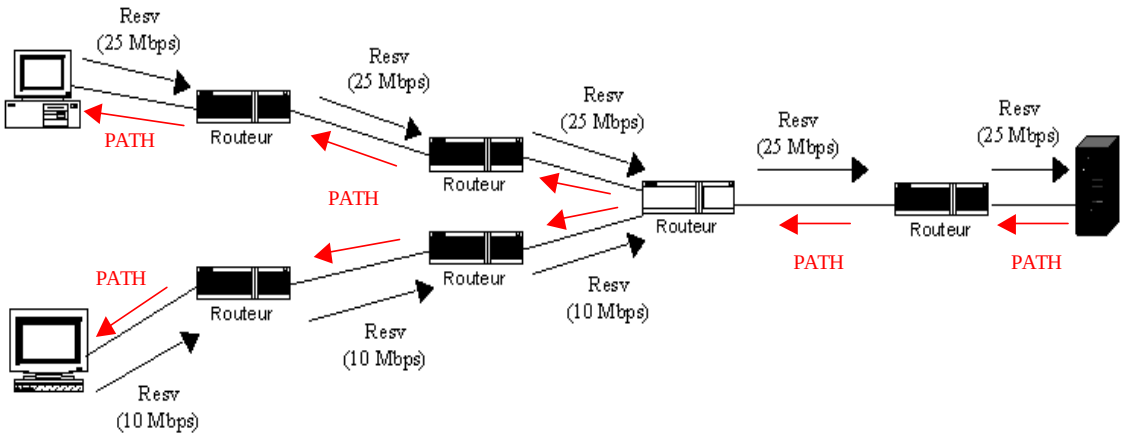


Figure I.5: Fonctionnement du protocole RSVP

- Des messages PATH sont émis périodiquement vers la destination. Ces messages décrivent le flux de données émis par chaque source (en termes de qualité de service) et établissent dans les routeurs un "état de chemin" (*PATH state*) par flux.

- Des messages RESV sont émis périodiquement vers les émetteurs. Ces messages décrivent les réservations à effectuer. Grâce aux états de chemins établis par les messages PATH, les messages RESV suivent le chemin inverse des paquets de données. Ainsi, les réservations sont effectuées dans les routeurs relayant ces données. Comme les ressources sont réservées à la requête des receveurs, RSVP offre donc un style de réservation dit "orienté receveurs". La période est choisie de la même manière que pour les messages PATH.

- Pour des raisons d'optimisation, les messages PATH et RESV émis respectivement en amont et en aval d'un routeur et n'ayant pas d'effet immédiat sur l'état du réseau, sont "bloqués" par ce routeur. Cependant, chaque routeur émet périodiquement ses propres messages PATH et RESV décrivant l'état des flux qu'il sert. Ceci permet aux routeurs RSVP de fusionner plusieurs messages RESV (venant de receveurs différents) en un seul message et ainsi d'assurer qu'il n'y a au plus, par flux, qu'un message PATH et qu'un message RESV émis sur chaque lien, par période de

rafraîchissement. Par contre, lorsqu'un routeur reçoit un message affectant l'état d'un des flux qu'il sert, il retransmet sans délai une copie de ce message.

- Un temps de vie (*timer*) est associé à chaque ressource réservée. La valeur de ce temps de vie est réinitialisée chaque fois qu'un message RESV confirme l'utilisation de cette ressource. Si le temps de vie vient à s'écouler, la ressource correspondante est libérée. Ce type de réservation est dit basé sur le principe d'état éphémère (*soft state*). Celui-ci est aussi appliqué aux états de chemins mémorisés dans les routeurs (et rafraîchis par les messages PATH). Il est à noter que ces temps sont choisis de sorte à tolérer la perte de plusieurs messages de contrôle consécutifs (par défaut, deux messages de contrôle peuvent être perdus sans causer le relâchement impromptu de la ressource associée).

- Pour améliorer le temps de réponse de RSVP aux changements dynamiques du routage dans le réseau, le mécanisme dit de "réparation locale" a été introduit. Lorsqu'un nœud RSVP détecte un changement de route [ZAP98], il envoie, sans délai, un message PATH par flux re-routé le long de la nouvelle portion de route. Par ailleurs, lorsqu'un nœud RSVP reçoit un message PATH pour un flux qu'il sert mais pour lequel l'état de chemin qu'il a mémorisé diffère de celui indiqué par ce message, le nœud met à jour son état de chemin. Il transmet immédiatement, le long du nouveau tronçon de route, un message RESV décrivant la réservation effectuée pour ce flux.

- Des messages de relâchement (*teardown messages*) peuvent être utilisés pour relâcher les états de chemins et les réservations. Les requêtes de relâchement sont initiées soit par l'émetteur, le(s) receveur(s) ou par n'importe quel nœud RSVP intermédiaire (à l'expiration d'un temps de vie).

Dans l'état de spécification actuel, il est à noter que tous les messages de contrôle de RSVP sont échangés de façon non fiable, c'est-à-dire que RSVP n'utilise pas d'accusé de réception. De plus, un flux pour lequel aucune ressource n'a été réservée reçoit de la part d'un routeur un traitement dit du "meilleur effort". Par conséquent, l'échec d'une réservation n'empêche en rien l'envoi de données.

I.3.2.3.3 Les limites de RSVP

RSVP oblige à maintenir l'état des ressources d'un flux. Lorsque le nombre d'usagers augmente, le nombre d'états devient conséquent avec le trafic généré pour les rafraîchissements. Cela nuit aux performances du système dans son ensemble. C'est pourquoi RSVP est plus adapté à des réseaux de petite taille.

Les opérateurs préfèrent augmenter les ressources que de réserver les ressources vu la complexité du système sans oublier que la démarche de réservation des ressources est inégalitaire puisqu'elle favorise les consommateurs payants.

Enfin, il existe d'autres architectures reposant sur l'étiquetage de priorité dans les paquets et la classification des services, plus simples à mettre en oeuvre. Paradoxalement, ces architectures qui peuvent paraître comme concurrentes de RSVP peuvent être ses alliés en allégeant au maximum la tâche de maintien d'états, de rafraîchissement et de classification.

I.3.2.4 Architecture DiffServ (Differentiated Services)

L'architecture à différenciation de services (Differentiated Services ou DiffServ) a été proposée par l'IETF [FER97], [BER98], [NIC98]. Son principe consiste à distinguer et à classer les flux afin de leur appliquer une politique correspondant à leur priorité. On ne distingue plus seulement deux types de flux comme dans l'architecture IntServ, mais plusieurs. A chaque type de flot est associé une priorité. Ainsi, la priorité la plus élevée (trafic premium) correspond aux flux requérant la plus grande qualité de service (flux temps-réel), et la priorité la plus faible, aux flux ne nécessitant pas de contraintes particulières. Les priorités intermédiaires se déclinent suivant les niveaux de qualité de service requis. La classification est effectuée au niveau des paquets en considérant le champ TOS (Type Of Service d'IPv4 que l'IETF a redéfini en champ DS (DiffServ)) de leur en-tête. Ce dernier inclus la priorité, ainsi qu'éventuellement l'identificateur du flux [BAK98]. Les paquets sont ensuite placés dans l'une des files d'attentes existantes dans chaque interface de sortie, chacune d'entre elles correspondant à un service différent. Enfin, c'est la politique d'ordonnancement de ces files d'attente qui détermine l'ordre dans lequel les paquets sont émis vers le routeur suivant. Les composants nécessaires à la mise en place de l'architecture DiffServ doivent donc présenter les fonctionnalités suivantes (Figure I.6):

1. à l'entrée du paquet dans le réseau :

   - Une priorité est affectée à chaque paquet. Cette priorité peut dépendre de l'identité de l'émetteur (par exemple un utilisateur ayant souscrit un abonnement préférentiel) ou de la nature du trafic. Dans ce dernier cas, la priorité dépend des contraintes de qualité de service requises.

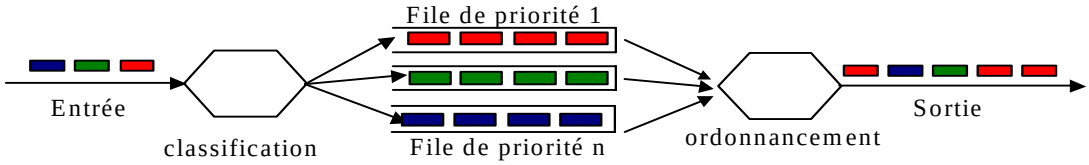


Figure I.6 : Le modèle Diffserv

2. dans les routeurs :

   - Les routeurs périphériques (Edge Router) effectuent les opérations de classification, contrôle et marquage.
   - Les routeurs centraux (Core Router) traitent les paquets en fonction de la classe codée dans l'en-tête de IP (champ DS) selon un comportement spécifique: le PHB (Per Hop Behavior).

Deux comportements de routeurs (PHB) ont été définis:

   - Expedited Forwarding (EF) ou premium service [JAC99]; il a pour but de garantir une bande passante avec des taux de perte, de délai et de gigue faibles.
   - Assured Forwarding (AF) [HEI99] regroupant plusieurs PHB garantissant un acheminement de paquets IP avec une haute probabilité. Cette famille de PHB est scindée en quatre classes garantissant une bande passante et un délai minimum. Chaque classe comprend 3 niveaux de priorité (Drop Precedence).

La classification des paquets dans les routeurs se fait suivant leur priorité, et leur placement dans la file d'attente. On retrouve plusieurs techniques d'ordonnancement des files d'attente dans le modèle Diffserv.

I.3.2.4.1 FIFO

L'ordonnancement FIFO est la politique la plus simple (Figure I.7). C'est celle qui est implémentée de façon standard dans le réseau Internet, ne permettant que le Best Effort.

Le principe FIFO (First In, First Out) garantit que les paquets sont transmis en sortie dans l'ordre dans lequel ils sont arrivés. Toutefois, lorsque la file est pleine, les paquets entrants sont détruits. Par ailleurs, il est possible d'implémenter une différenciation de services avec une seule file, en imposant que les paquets moins prioritaires soient détruits avant que la file ne soit pleine, laissant ainsi leur place aux paquets plus prioritaires.

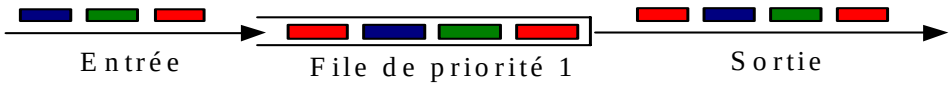


Figure I.7 : Ordonnancement FIFO

L'utilisation de plusieurs files d'attente apparaît toutefois comme un choix plus "pratique" et plus proche des principes de l'architecture DiffServ.

I.3.2.4.2 Files prioritaires

La méthode des files prioritaires utilise donc une file par classe de service. A chacune de ces files est associée une priorité stricte, c'est-à-dire que les paquets d'une file ne sont émis sur l'interface de sortie que lorsque les files plus prioritaires sont vides (Figure I.8).

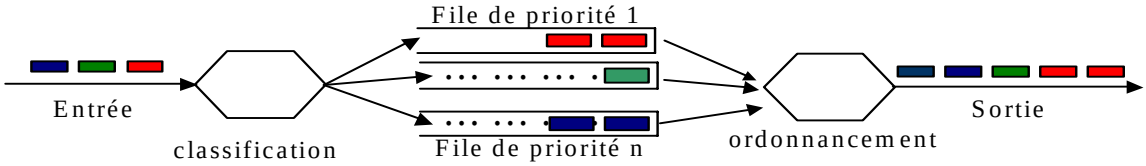


Figure I.8: File prioritaires

La technique des files prioritaires présente toutefois un inconvénient majeur. En effet, un trafic prioritaire important engendre une famine des trafics moins prioritaires. Pour pallier ce défaut, on peut appliquer un contrôle de débit aux flux prioritaires (ce qui n'est pas le but recherché) ou bien imposer un débit minimal aux files de moindre priorité.

I.3.2.4.3 GPS (General Processor Sharing)

La technique GPS associe une priorité et une file d'attente à chaque flot [PAR94]. L'Ordonnanceur retire ensuite de chaque file, à tour de rôle, une quantité de données proportionnelle à sa priorité et l'émet. Les politiques d'ordonnancement décrites ci-après ont été dérivées du GPS.

I.3.2.4.4 RR (Round Robin), WRR (Weighted Round Robin), DRR (Deficit Round Robin) et DWRR (Deficit Weighted Round Robin)

La technique du Round Robin est très simple. Les files d'attente associées à chaque flot sont examinées à tour de rôle. Un paquet est extrait de chaque file non vide, puis est émis sur l'interface de sortie (Figure I.9). Chaque flot est donc traité équitablement.

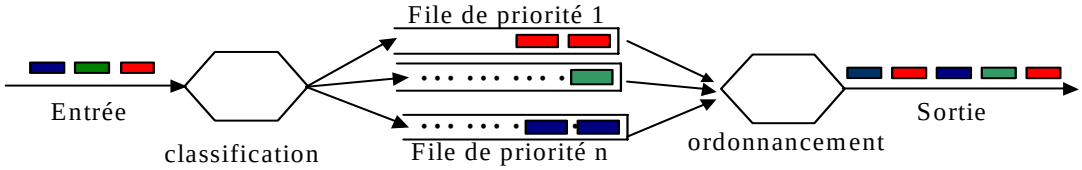


Figure I.9: Round Robin

La méthode du Weighted Round Robin fonctionne sur le modèle du Round Robin, à cette différence près : à chaque flot (et donc à chaque file), est associée une priorité.

L'Ordonnanceur examine alors les files non vides, extrait un nombre de paquets proportionnel à la priorité, puis les émet (Figure I.10). Certains flux sont donc privilégiés par rapport à d'autres.

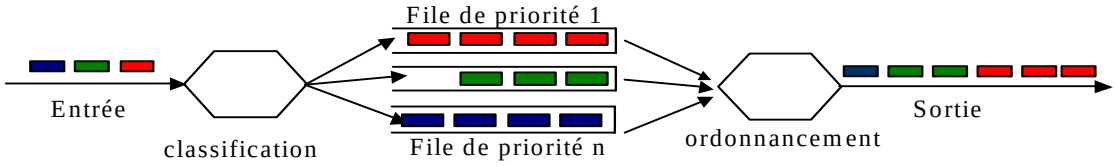


Figure I.10 : Weighted Round Robin

Le principal inconvénient des techniques RR et WRR est qu'elles ignorent la taille des paquets. Un flot composé de paquets de plus grande taille est donc privilégiée, en termes de quantité de données transmise par les routeurs. Les techniques DRR et WDRR ont donc été proposées pour y remédier. Le DRR (Deficit Round Robin) permet de pondérer la quantité

de données extraites des files pour chaque flot par la taille des paquets composant ce flot grâce à la méthode des quantum [SHR95]. De même, WDRR fonctionne sur le même schéma, tout en tenant compte des priorités des flux.

I.3.2.4.5 FQ (Fair Queuing) et WFQ (Weighted Fair Queuing)

Toujours dans le but de tenir compte de la taille des paquets, la technique du Fair Queuing [DEM89] consiste à effectuer un Round Robin sur les files d'attente, mais en simulant l'extraction d'un seul bit de chaque file non vide. En fait, on calcule l'instant t où un paquet sera entièrement extrait avec un tel ordonnancement, et on fixe l'extraction effective du paquet à cet instant t. La technique du Weighted Fair Queuing fonctionne sur le même principe, tout en assignant un poids à chaque file.

Les techniques basées sur le GPS apportent donc un certain nombre d'avantages : partage équitable des ressources, différenciation des services, élimination des phénomènes de famine. Toutefois, ils utilisent une file par flot. Ce principe pose naturellement des problèmes d'échelle. En effet, plus le nombre de flux est grand, plus le nombre de files est grand, et plus les tâches de classification et d'ordonnancement sont coûteuses. Les propriétés remarquables de cette famille de techniques ont été conservées, tout en réduisant le nombre de files à gérer.

I.4 Conclusion

Dans ce chapitre, nous avons passé en revue les différentes techniques permettant de garantir une QoS dans les réseaux. Ces techniques interviennent à différents niveaux. Le premier concerne les **extrémités** d'un lien de communication avec comme exemple le démarrage lent et le contrôle de débit dans le protocole TCP. Le deuxième niveau concerne quant à lui les nœuds intermédiaires du **réseau** où sont proposées des techniques de prévention de congestion (RED, WRED), ou des solutions de gestion explicite de la Qualité de service réseau (IntServ/RSVP, DiffServ). Même si ces techniques apportent des éléments de solution à la problématique de la QoS, elles ne garantissent pas la QoS de bout en bout et par conséquent, elles doivent être complétées par des solutions de routage adaptatif. En effet, un réseau IP est caractérisé par l'hétérogénéité de ses liens et par des conditions de trafic dynamiques, pouvant provoquer des goulots d'étranglement, ce qui nécessite la mise en œuvre de politiques de routage adaptatif permettant une meilleure prise en compte de la QoS.

Chapitre II

Routage avec Qualité de Service

II.1 Introduction

L'objectif de ce chapitre est de présenter la problématique de l'intégration de la qualité de service dans la prise de décision du routage. Tout d'abord, nous présentons la fonction de routage dans les réseaux de télécommunication ainsi que les algorithmes couramment utilisés. Nous développons ensuite les différentes approches de routage avec qualité de service proposées dans la littérature. L'analyse de ces travaux nous permet d'une part, de retenir le principe du routage adaptatif basé sur l'apprentissage comme base de travail et d'autre part, de proposer deux nouvelles approches algorithmiques permettant d'améliorer les approches actuelles du point de vue du temps de convergence et de l'occupation mémoire.

II.2 Routage dans les réseaux de télécommunication

II.2.1 Principe

Le routage est l'une des fonctionnalités principales de la couche réseau qui a la responsabilité de décider sur quelle ligne de sortie, un paquet entrant doit être retransmis. D'une manière générale, on distingue la *remise directe*, qui correspond au transfert d'un datagramme entre deux ordinateurs du même réseau, et la *remise indirecte* qui est mise en oeuvre quant au moins un routeur sépare l'expéditeur initial et le destinataire final.

Par exemple, dans le cas d'un réseau Ethernet, la remise directe consiste à encapsuler le datagramme IP (Annexe 1) dans une trame Ethernet après avoir utilisé le protocole ARP [PLU82] pour faire la correspondance adresse IP / adresse physique, puis à émettre cette trame sur le réseau. L'expéditeur peut savoir que le destinataire final partage le même réseau en utilisant simplement l'adresse IP de destination du datagramme. Il en extrait l'identificateur de réseau et si c'est le même que celui de sa propre adresse IP, alors la remise directe est suffisante. En fait, ce mécanisme, on le retrouve toujours lors de la remise d'un datagramme entre le dernier routeur et le destinataire final.

Pour sa part, la remise indirecte nécessite de déterminer vers quel routeur envoyer un datagramme IP en fonction de sa destination finale. Ceci est rendu possible par l'utilisation d'une *table de routage* spécifique à chaque routeur, permettant de déterminer vers quelle voie de sortie envoyer un datagramme destiné à un réseau quelconque.

II.2.2 La table de routage

L'essentiel du contenu d'une table de routage est constitué de quadruplets *(destination, passerelle, masque, interface)* ou :

- **Destination** est l'adresse IP d'une machine ou d'un réseau de destination.
- **Passerelle** (Gateway) est l'adresse IP du prochain routeur vers lequel envoyer le datagramme pour atteindre cette destination.
- **Masque** est le masque associé au réseau de destination.
- **Interface** désigne l'interface physique par laquelle le datagramme doit réellement être expédié.

Une table de routage (Figure II.1) contient notamment une *route par défaut* qui spécifie un routeur vers lequel sont envoyés tous les datagrammes pour lesquels il n'existe pas de route dans la table.

Tous les routeurs mentionnés dans une table de routage doivent être directement accessibles à partir du routeur considéré. Cette technique, dans laquelle un routeur ne connaît pas le chemin complet menant à une destination, mais simplement la première étape de ce chemin, est appelée routage par sauts successifs (*next-hop routing*).

Exemple de table de routage

Destination	Gateway	Masque	Interface
140.252.13.65	140.252.13.35	255.255.255.255	Eth0
127.0.0.1	127.0.0.1	255.255.255.255	lo0
140.252.13.32	140.252.13.34	255.255.255.224	Eth1
Default	140.252.13.33	0.0.0.0	S1

Figure II.1: Table de routage

- **L'adresse** 127.0.0.1 est celle de lo0, l'interface de bouclage local (loopback), qui sert à pouvoir faire communiquer une machine avec elle-même.

- La destination **default** sert à indiquer la destination de tous les datagrammes qui ne peuvent être « routés » par l'une des autres routes.

La taille d'une table de routage dépend de la taille du réseau. Par conséquent, afin d'accélérer la recherche de correspondances parmi les entrées d'une table, il est préférable que sa taille soit réduite. A ses débuts, l'Internet regroupait un petit nombre de machines isolées et de petits réseaux [LOT92]. Les tables de routage étaient donc de petite taille. Un seul protocole assurait le routage, GGP (Gateway To Gateway Protocol). Avec la fantastique expansion qui s'est produite, l'explosion des tables de routage fut l'une des raisons pour lesquelles la décision de découper l'Internet en un ensemble d'entités indépendantes fut prise.

II.2.3 Algorithme de routage

A la réception d'un paquet par un routeur, ce dernier doit déterminer vers quel routeur le transmettre, à partir de la table de routage construite par l'algorithme de routage utilisé. La formulation générale d'une telle procédure s'écrit :

```
Procédure Routage IP(données Dat : datagramme, Tab : Table de routage)
début
  D := adresse IP de destination de Dat
  si D appartient à l'un des réseaux directement accessibles
    alors // remise directe
          Envoyer Dat vers l'adresse D sur ce réseau
          {il y a résolution de l'adresse IP, en adresse physique, encapsulation de Dat dans une trame physique et
          émission de la trame via l'interface Physique correspondante}
    sinon // remise indirecte
      pour chaque entrée de Tab faire
            N:= (D + du masque) résultat du "et logique" de D et du masque de sous-réseau
            si N=l'adresse réseau de la destination de l'entrée
              alors
                Router Dat vers cette destination
                 Sortir
            finsi
      finpour
    si aucune correspondance n'est trouvée
      alors
      Retourner à l'application d'origine du datagramme une erreur de routage (machine inaccessible, réseau
      inaccessible, etc.).
    finsi
  finsi
fin
```

Le fait de considérer le réseau Internet comme un système homogène et unifié, a rapidement posé des problèmes. Tout d'abord, il est évident que plus le réseau est grand, c'est-à-dire regroupant plus de nœuds et de destinations potentielles, plus la probabilité d'un changement de topologie du réseau est grande (ajout d'un nœud, d'un lien ou panne). Le routage est donc susceptible de changer plus fréquemment. Le choix et la mise à jour des protocoles deviennent particulièrement complexes, surtout si les organisations propriétaires des portions du réseau ne parviennent pas à s'entendre.

Toutes ces raisons ont conduit au découpage du réseau Internet en un ensemble d'entités indépendantes appelées Systèmes Autonomes (AS) [HAW96], que nous désignerons aussi sous le nom de domaines, chacun d'entre eux étant sous une administration unique. Les

protocoles sont unifiés au sein d'un même AS, ce qui permet d'assurer un routage efficace. De plus, l'un des AS du réseau Internet permet d'interconnecter tous les autres. Ce système autonome particulier est appelé épine dorsale (backbone). Ainsi, tous les AS du réseau Internet sont interconnectés (Figure II.2). On nomme les nœuds (ou routeurs) appartenant à un AS des passerelles (ou gateways) internes. Les routeurs permettant l'interconnexion entre deux AS sont appelés passerelles (ou gateways) externes.

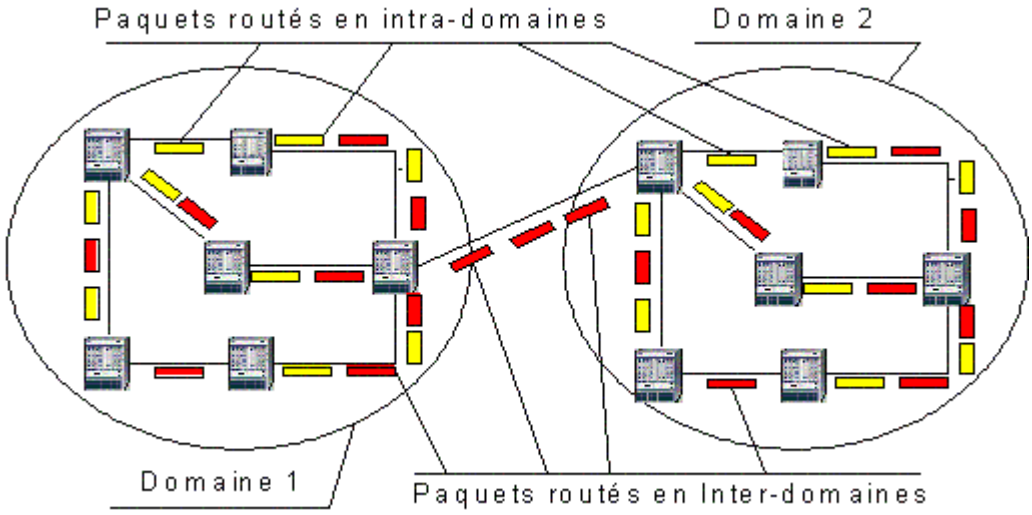


Figure II.2 : Type de routage

Le calcul des tables de routage est effectué indépendamment grâce à un protocole de routage. Ce calcul permet de déterminer le plus court chemin entre deux nœuds du réseau. Il existe de nombreux protocoles de routage, que l'on peut diviser en deux catégories :

- Les protocoles intra-domaines "interior Gateway Protocol (IGP)" (tels que RIP, OSPF [MOY98], IGRP, IS-IS [CAL90]) qui calculent les tables de routage au sein d'un même AS.

- Les protocoles inter-domaine "Exterior Gateway Protocol (EGP)" (tels que EGP [MIL84] ou BGP [LOU91]) qui calculent les tables de routage entre des nœuds appartenant à des AS différents.

Dans ce mémoire, nous nous focaliserons plus particulièrement sur les protocoles intra-domaines, que l'on peut répartir entre deux grandes familles :

- Les protocoles à vecteur de distance.

- Les protocoles à états de liens.

II.3 Les Algorithmes classiques de routage

II.3.1 Le routage à vecteur de distance

L'algorithme de routage à vecteur de distance ou routage Bellman Ford [THO90], est basé sur le principe que chaque routeur dispose d'une table de routage indiquant pour chaque destination, la meilleure distance connue et par quelle ligne l'atteindre. Cet algorithme est ainsi utilisé sur Internet dans le protocole RIP que nous détaillerons un peu plus loin. La distance estimée dans la table de routage suppose que l'on se soit donné une métrique qui doit pouvoir être calculée au niveau local pour chaque routeur. Cela peut être par exemple :

- Le **nombre de sauts** (nombre de routeurs dans un chemin). Dans ce cas, la distance pour chaque routeur voisin sera égale à 1.

- Le **nombre de paquets dans la file d'attente** correspondant à la sortie reliée au routeur voisin.

- Le **temps d'acheminement**, que le routeur calcule en envoyant un paquet spécial, HELLO, au routeur voisin, qui renvoie immédiatement le paquet. Ainsi, le temps d'acheminement d'un paquet est calculé comme la moitié du temps nécessaire à un aller-retour.

- Le **coût** associé à chaque lien (bande passante, taux d'erreur, distance).

Les seuls critères possibles pour le choix de la métrique sont d'une part, la pertinence pour le problème posé et d'autre part, la nécessité de posséder une certaine linéarité, c'est-à-dire qu'il soit possible de calculer le coût d'un chemin en entier comme la somme des coûts des sauts élémentaires qui le composent.

L'algorithme de routage à vecteur de distance vient du fait qu'il est possible de calculer la route optimale en échangeant comme seule information la liste des métriques ou distances. De plus, cette information n'est échangée qu'entre routeurs voisins. Tout d'abord, chaque routeur commence à enregistrer dans sa table de routage, les liaisons avec les routeurs voisins en indiquant la ligne et la distance estimée. Ensuite, périodiquement, chaque routeur envoie à

tous ses voisins une table extraite de sa table de routage contenant pour chaque destination la distance évaluée, ce qui permet de mettre à jour la table de routage. Ainsi, supposons qu'un routeur K ait dans sa table de routage la destination X avec une distance d1. Il envoie cette information à son routeur voisin J, qui possède dans sa table de routage une entrée pour le routeur *K,* avec une distance *d2* et une ligne de sortie associée s. Le routeur *J* sait alors à la réception de cette information que la distance pour la destination *X* en passant par le routeur *K* est *d=d1+d2*. Si la distance inscrite dans la table de routage est plus grande que *d*, ou si la destination *X* n'apparaît pas encore dans la table de routage, alors cette dernière est mise à jour en inscrivant que pour la destination *X*, la distance est *d*, et la sortie de ligne est *s*.

Le principe du routage à vecteur de distance permet a priori de trouver le chemin le plus court suivant la métrique imposée.

II.3.1.1 Exemple de protocole de routage à vecteur de distance : RIP

Le protocole RIP (Routing Information Protocol) est un protocole intra-domaine (IGP Interior Gateway Protocol) [HED88]. Il est basé sur un algorithme de routage à vecteur de distance, dont la métrique est le nombre de sauts. Il est prévu pour des réseaux dont la distance entre 2 points est au maximum de 15 sauts. Le premier mécanisme implémenté dans RIP pour prendre en compte la topologie du réseau est le suivant :

- Chaque routeur possède une table de routage avec une entrée pour chaque destination ;
- Toutes les 30 secondes, chaque routeur envoie à chacun de ses voisins le contenu de sa table de routage ;
- La réception d'un de ces messages par un routeur entraîne la mise à jour de la table de routage. Ainsi, si la nouvelle distance estimée est plus petite que celle présente dans la table, alors cette dernière est modifiée. Si le routeur voisin qui a envoyé le message à un routeur A est B, alors on force la mise à jour dans la table de A de toutes les destinations qui indiquent que le routeur voisin à qui il faut envoyer le paquet est B, et ceci, même si les nouvelles distances calculées sont plus grandes que les anciennes.

Ce mécanisme évite la synchronisation des routeurs entre eux, sans pour autant améliorer la vitesse de réaction comme dans le cas de la panne d'un routeur. Par conséquent, les améliorations suivantes sont nécessaires :

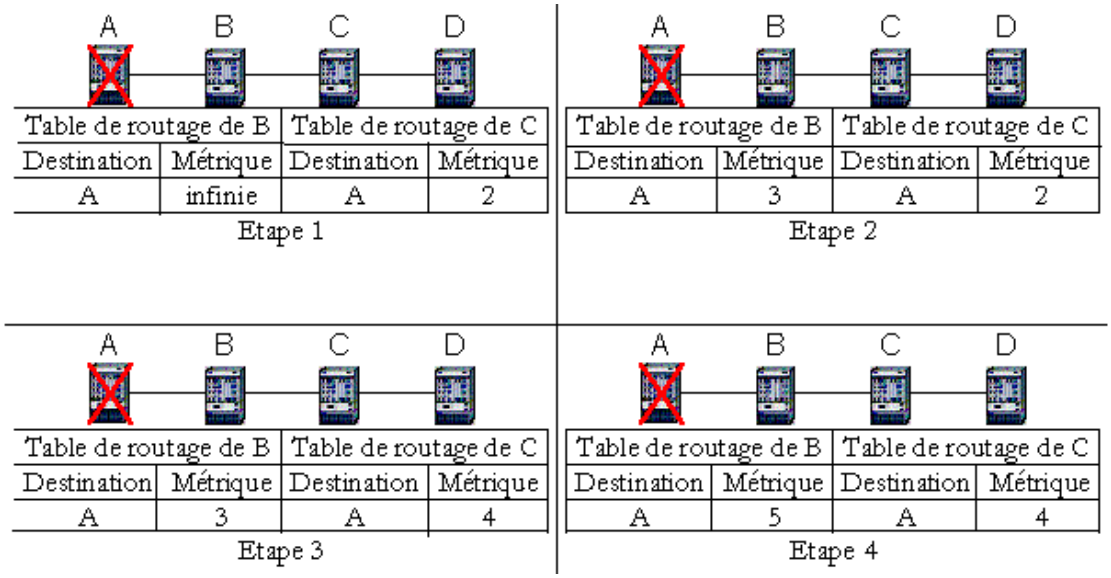


Figure II.3 : Rupture d'un routeur

- Afin de détecter la rupture d'un routeur ou d'un lien, la distance pour atteindre un routeur est estimée à 16 si on ne reçoit plus de nouvelles de ce routeur pendant 180 secondes.

- Certains problèmes de boucles peuvent apparaître lors, par exemple, de la rupture d'un routeur, parce que les routeurs continuent à s'échanger des paquets et ainsi à garder en mémoire une liaison qui n'existe plus. Pour y remédier, on peut utiliser la méthode de **l'horizon partagé** [HUI95] : Quand un routeur apprend d'un voisin une nouvelle route, la première table qu'il lui renvoie correspond à sa table de routage mais avec une valeur infinie pour les routes nouvellement apprises, ce qui évite des aller-retour inutiles. Cependant, cette méthode présente l'inconvénient de garder dans les tables des routes avec des distances infinies, ce qui peut être gênant pour des raisons de taille de table, et par conséquent de bande passante lors des échanges de messages périodiques. Généralement, une durée (timer) est utilisée pour effacer les routes dont la distance est restée égale à une valeur infinie plus de 120 secondes.

- De plus, pour accélérer la convergence de l'algorithme, un routeur transmet directement ses modifications à ses voisins, dès qu'il reçoit une information changeant sa table de routage. Cette procédure correspond à une **mise à jour déclenchée** [HUI95].

Pour mettre en évidence les limites de l'algorithme de routage RIP, nous l'avons évalué en simulation sous le logiciel OPNET (paragraphe IV.6.2). Pour cela, nous avons opté pour un réseau irrégulier comprenant 32 nœuds (Figure II.4). Un flux de paquets est envoyé depuis le nœud source1 au nœud destination1 auquel s'ajoute, au bout d'une heure de simulation, un important pic de flux du nœud source2 au nœud destination2, afin de saturer les routeurs 21 et 22.

Dans le réseau illustré Figure II.4, il existe deux voies possibles pour acheminer les paquets entre la partie gauche et la partie droite du réseau : l'itinéraire incluant les routeurs 21 et 22 (R1) et celui incluant les routeurs 29 et 30 (R 2).

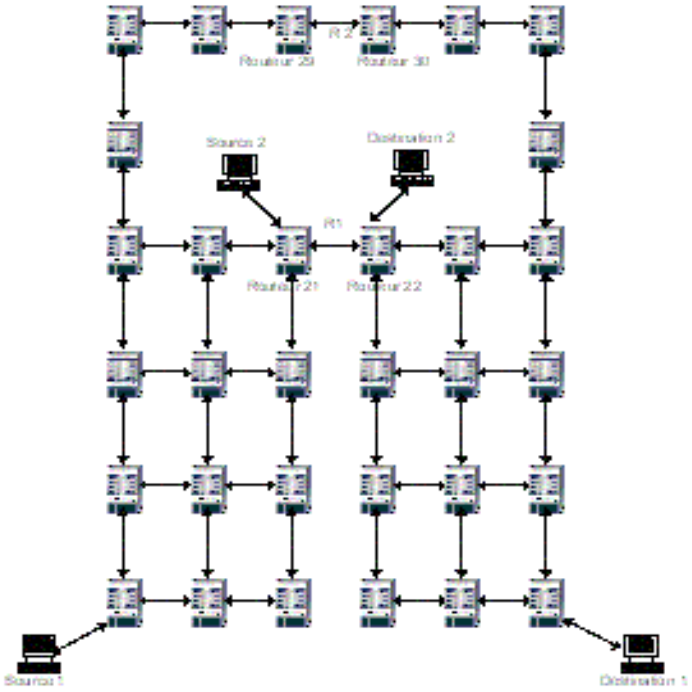


Figure II.4 : Réseau utilisé pour la simulation.

Le résultat illustré Figure II.5, montre clairement, qu'après avoir surchargé la route optimale R1 (début de congestion), le temps moyen d'acheminement des paquets se dégrade sensiblement. En effet, à mesure que la charge du réseau augmente les paquets sont retardés

dans les files d'attente des routeurs car RIP garde la même route tant qu'il ne détecte pas de changement dans la topologie du réseau.

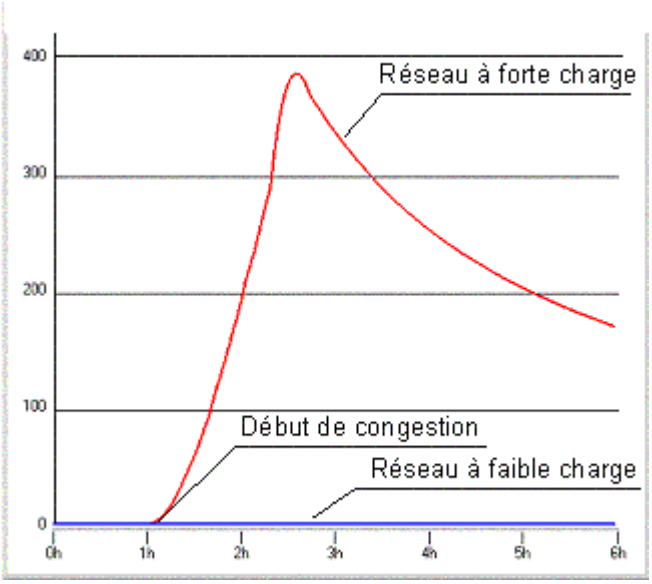


Figure II.5: Comportement du protocole RIP suite à la congestion du chemin optimal

De nombreuses améliorations à RIP ou plus généralement aux protocoles à vecteur de distance ont été apportées. On peut citer notamment la version 2 du protocole [MAL93], solutionnant le problème de la taille des tables de routage dans RIP-version 1 d'une part, en permettant le routage par sous réseau et d'autre part, en traitant les communications multipoints. Ce protocole offre également une meilleure authentification des échanges entre routeurs. D'autres contributions ont également été proposées pour limiter les phénomènes de synchronisation [FLO93] et de détection des boucles. Enfin, des protocoles dérivés de RIP (IGRP et IGRP étendu), apportant de nouvelles solutions pour la suppression des boucles, sont apparus. Toutefois, la majorité de la communauté Internet semble aujourd'hui considérer les protocoles à vecteur de distance comme obsolètes par rapport aux protocoles à états de liens car ils n'ont qu'une vision locale de la topologie du réseau et ne permettent de prendre en compte qu'un seul critère dans la recherche des routes optimales.

II.3.2 Le routage par information d'états de liens

Dans un algorithme de routage par information à états de liens, chaque routeur maintient une même base de données décrivant la topologie du réseau. Ensuite, chacun exécute le même algorithme en parallèle, en construisant un arbre, dont il sera la racine, pour trouver le

plus court chemin. Cet arbre du plus court chemin permet d'établir la route entière pour chaque destination du réseau, mais seul le prochain saut est sauvegardé dans les tables de routage. Suite au changement d'état d'un lien, le routeur détectant cet événement, transmet l'information par un mécanisme d'inondation (diffusion) à tous les autres routeurs.

Comme dans le cas du routage à vecteur de distance, la notion de chemin le plus court fait appel à une distance définie par une métrique pouvant être le nombre de sauts, le nombre de paquets dans la file d'attente, etc. L'initialisation commence par la recherche des voisins et l'évaluation de la distance pour les atteindre. Ensuite, chaque distance estimée est envoyée à tous les routeurs du réseau. Chaque routeur peut donc inscrire dans sa base de données la topologie complète du réseau, et ensuite en se posant en racine, construire l'arbre pour calculer le chemin le plus court. Ce calcul demande une certaine capacité, qui limite donc les dimensions des réseaux fonctionnant avec des algorithmes de ce type : chaque routeur devant recalculer le chemin en entier sur l'arbre, à chaque changement d'état.

II.3.2.1 Exemple de routage par information d'état de lien : le protocole OSPF

Le protocole OSPF (Open Shortest Path First) est un protocole de routage intra-domaine (IGP Interior Gateway Protocol) [MOY89] [MOY98]. Il ne diffuse les informations qu'entre les routeurs appartenant à un même système autonome afin de réduire la charge de calcul de chaque routeur, de limiter le trafic de routage et d'accélérer la convergence de la table de routage.

OSPF est donc un protocole basé sur un algorithme d'information d'état de lien, utilisant l'algorithme SPF (Short Path First) ou l'algorithme de Dijkstra [TCC90]. Sa métrique par défaut est proportionnelle à l'inverse de la bande passante du lien. Quand un routeur démarre, il initialise tout d'abord une structure de données pour le protocole de routage, et attend ensuite que les couches plus basses lui indiquent que ses interfaces sont fonctionnelles. Le routeur utilise ensuite le protocole Hello pour détecter ses voisins et créer entre eux des adjacences. Deux routeurs voisins sont considérés comme adjacents s'ils ont synchronisé leurs bases de données topologiques. Le protocole Hello permet aussi d'assurer une communication bidirectionnelle avant d'échanger des informations d'état de lien. Des paquets Hello sont envoyés aussi périodiquement (par défaut toutes les 10 secondes) par les routeurs pour vérifier que les liaisons sont opérationnelles.

OSPF utilise 2 mécanismes pour détecter les changements de topologie :

- Les changements d'états d'interface.
- L'expiration d'un temporisateur (réglé à 40 secondes) à la suite de l'envoi d'un paquet Hello, indiquant que le routeur voisin est inactif.

Lors d'un changement de topologie, tous les routeurs en sont prévenus par un système d'inondation. Chaque routeur remet alors à jour sa base de données topologiques, puis recalcule à partir de celle-ci son arbre du plus court chemin.

OSPF possède aussi certaines options :

- Il permet de calculer indépendamment une route pour chaque type de service, si l'on différencie les services par le champ "type de service" dans l'entête IP (TOS). Les paquets pour les informations de routage sont affectés d'un champ TOS égale à 0.
- Il permet, quand plusieurs routes ont le même coût, de répartir le trafic sur différentes routes.
- Il permet, pour le calcul de la meilleure route, de combiner plusieurs critères.

II.3.3 Synthèse sur les protocoles classiques de routage

Les familles de protocoles de routage que nous avons présentées jusqu'ici permettent de calculer les tables de routage utilisées par IP pour acheminer des paquets d'une source vers une destination. Qu'il s'agisse des protocoles à vecteurs de distance (comme RIP) ou à états de liaisons (comme OSPF), leur objectif reste le même. Il est de calculer, en associant un coût (métrique) à chaque lien, le plus court chemin entre chaque couple source-destination. Actuellement, le routage intra-domaines utilise une métrique unitaire pour tous les liens. Le plus court chemin est donc celui qui comporte le nombre de liens minimal entre la source et la destination (plus court chemin en termes de nombre de sauts ou hop count). Il s'agit d'un choix simpliste et inadapté au besoin de qualité de service car le routage "hop count" ne peut permettre de respecter des contraintes de bande passante ou de délai. En effet, le chemin le plus court en nombre de sauts peut comporter des liens de faible capacité. Dans le cas d'un

trafic important, il peut alors être sujet à l'apparition de congestions et à l'augmentation du délai.

Le choix de la métrique est donc déterminant. On peut, par exemple, choisir une métrique proportionnelle à la bande passante des liens. Dans ce cas, la bande passante d'un chemin est égale à la plus petite bande passante des liens le composant. Le meilleur chemin est alors celui présentant la bande passante maximale. Toutefois, le fait d'adopter une telle métrique permet d'orienter le trafic principalement sur les liens les plus performants, entraînant ainsi une mauvaise répartition de la charge dans le réseau : on risque de ne jamais utiliser les ressources offertes par les liens de moindre capacité. De plus, si le choix de cette métrique permet d'utiliser le chemin pouvant accepter le plus grand trafic, il ne peut déboucher sur aucune garantie effective. En effet, en forçant l'utilisation de certains liens (en l'occurrence ceux de forte capacité), on risque de voir passer la majorité du trafic sur ces liens. La bande passante résiduelle est alors très éloignée de la bande passante totale, sur laquelle on a basé le routage. De plus, l'inconvénient majeur de ces protocoles de routage est leur absence de réactivité. RIP, par exemple, a une constante de rafraîchissement de 30 secondes, et ne gère par conséquent que des objets statiques, puisqu'il ne prend comme métrique que le nombre de routeurs à traverser, les routes choisies ne pouvant changer que si un routeur ou un lien "tombe", ce qui arrive assez peu souvent [HUI95]. OSPF, quant à lui, constitue un algorithme réactif mais au prix d'un coût de calcul non négligeable, et sa réactivité n'est pas suffisamment élevée pour s'adapter en permanence à la taille des files d'attente et aux changements imprévisibles de trafic.

II.4 Routage avec Qualité de Service

Les besoins en télécommunications des utilisateurs se caractérisent de plus en plus par une augmentation des besoins en débit (services télévisuels du type VoD, accès à Internet, télétravail, visiophonie). L'utilisation des nouvelles générations de réseau dans le cadre d'applications multimédia, de services à qualité garantie, de services à diffusion, services mobiles, etc. impose que l'acheminement soit assuré avec une Qualité de Service (QoS : débit, délai, gigue, fiabilité, etc.) maîtrisée. Les techniques de routage présentées dans le paragraphe précédent ne répondent pas à ces caractéristiques. Ainsi, il est nécessaire de concevoir de nouveaux mécanismes de routage et par conséquent de repenser les algorithmes correspondants. Le routage doit alors prendre en compte l'état réel des liens et des routeurs

afin d'effectuer un routage par "qualité de service". La prise en compte de la qualité de service dans le routage s'organise donc autour de trois étapes:

1. L'estimation de l'état des liens du réseau ;
2. La collecte de ces informations par les routeurs ;
3. La détermination des routes permettant de respecter les contraintes de qualité de service.

Les approches de routage avec QoS peuvent êtres classées en quatre catégories :

- Celles basées sur le principe de commutation (MPLS).
- Celles dérivées des protocoles existants (QOSPF, multi chemin).
- Celles basées sur plusieurs métriques (SW, WS).
- Celles basées sur les techniques d'apprentissages (Q-Routing, Ants,…).

II.4.1 Protocole à base de commutation : MPLS (Multiprotocol label switching)

MPLS [ROS99], [STA01] est un protocole qui permet d'imposer une route fixe aux différents flux pour arriver à leurs destinations. Il est basé sur le concept de commutation de label (label switching). Une caractérisation de trafic (traffic characterisation [PAR92]), définissant la QoS requise [SHE97] est associée à chaque flux. Il n'est pas à proprement parler un protocole de niveau 3 (couche réseau), mais plutôt intermédiaire entre le niveau 2 (couche liaison) et le niveau 3 (couche réseau) du modèle OSI.

Paradoxalement, pour une couche située sensiblement au niveau inférieur à IP, MPLS a besoin de IP et des protocoles de routage associés pour exister. MPLS est une technologie permettant d'offrir à IP un mode circuit, à l'image de X25 ou ATM.

Les avantages de MPLS peuvent ce résumer ainsi :

- Rapidité de commutation au niveau des équipements de cœur.
- Possibilité d'associer des politiques de routage spécifiques pour certains flux.

Le principe de la commutation de label consiste à remplacer la recherche de la plus longue correspondance entre l'adresse de destination des paquets IP, et les préfixes présents dans les tables de routage en insérant un label de longueur fixe entre l'en-tête réseau et l'en-tête liaison des paquets. La détermination du prochain est alors effectuée grâce au label. Cette solution est plus avantageuse, puisqu'elle ne s'appuie plus sur la recherche de la plus longue correspondance avec des préfixes de longueur variable, mais sur la recherche d'un label de longueur fixe.

Le routage MPLS peut être effectué localement. En effet, il est possible que dans un réseau IP, certaines zones utilisent MPLS et d'autres non. La distribution des labels est alors effectuée par les routeurs d'accès au "domaine MPLS", grâce à un algorithme dynamique qui associe des labels aux adresses de destination des paquets entrants. Pour cela, les routeurs d'extrémité du domaine MPLS communiquent avec leurs voisins non MPLS. Ils échangent avec eux des informations de routage, et distribuent les labels aux routeurs internes pour effectuer un routage cohérent. Le routage est donc explicite, défini de bout en bout au sein du domaine MPLS (Figure II.6).

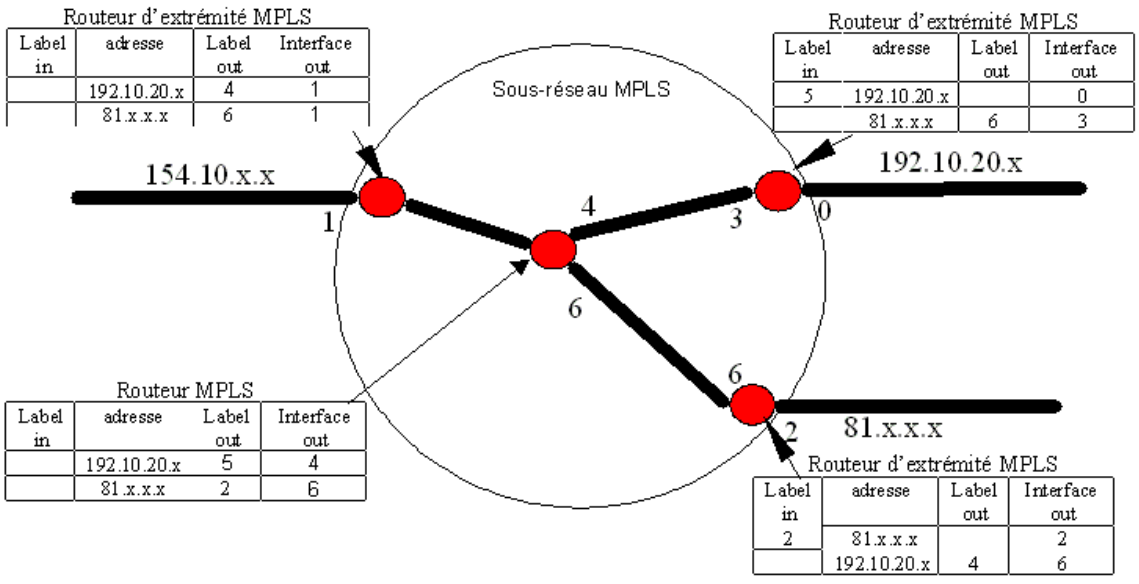


Figure II.6: Routage MPLS

Les perspectives de MPLS pour améliorer la qualité de services sont nombreuses. Cisco a contribué à la standardisation de MPLS en proposant le Tag Switching. Il s'agit d'utiliser les 3 bits IP de précédence (trois bits inclus dans le champ Type de service de l'en-tête IPv4) des paquets IP entrants pour les placer dans le champ CoS (Class of Service) du label MPLS.

Ainsi le routage peut être effectué de manière à privilégier, dans le domaine MPLS, les flux dont le CoS est le plus élevé. Par ailleurs, l'un des aspects les plus intéressant de MPLS est la possibilité de construire des routages explicites de bout en bout, les LSP (Label-Switched Paths). Ces chemins peuvent être déterminés afin de respecter des contraintes de qualité de service. Ainsi, plusieurs chemins parallèles peuvent coexister entre deux routeurs d'extrémité du domaine MPLS, chacun d'entre eux offrant des caractéristiques propres en termes de délai ou de bande passante.

Lorsqu'un flux arrive dans le domaine MPLS, le choix du chemin peut être fondé sur le CoS (un flux avec un CoS élevé sera envoyé sur un meilleur chemin, alors qu'un flux avec un CoS plus faible sera envoyé sur une route moins bonne).

II.4.2 Protocoles dérivés des protocoles existants

II.4.2.1 QOSPF

QOSPF (Quality Of Service Path First) [CRA98], [ZHA97a], est une extension d'OSPF. Combiné à un protocole de réservation, ce protocole de routage avec qualité de service permet d'annoncer à tous les routeurs, la capacité des liens à supporter des contraintes de qualité de service. QOSPF se base sur l'utilisation de deux messages, le RES-LSA (Link Resource Advertisement message) et le RRA (Resource Reservation Advertisement message). Le but des messages RES-LSA est d'informer l'ensemble des routeurs de l'état des liens. Ces informations sont utilisées pour calculer les chemins. Ce type de message est ainsi utilisé lorsqu'un nouveau routeur est ajouté au réseau ou lorsque la charge d'un lien varie.. Le rôle d'un message RRA est d'indiquer les ressources utilisées par un flux (ou plutôt réservées pour ce flux). Chaque routeur est averti des ressources utilisées et du chemin emprunté par chaque flux. Le nombre de messages RRA peut donc devenir très élevé si le nombre de flux augmente, ce qui pose un problème d'échelle. Cependant, on peut y remédier en effectuant un routage explicite : il s'agit de pré-calculer de manière centralisée la route permettant de satisfaire les contraintes de qualité de service, et d'imposer ce chemin à chaque routeur. Typiquement, ce calcul est effectué par la source (routage par la source), QOSPF n'est pas autorisé à remplacer un itinéraire existant sous prétexte qu'un chemin s'avère meilleur. C'est donc la stabilité qui est privilégiée par rapport à la performance immédiate.

II.4.2.2 Le routage multi chemin

Les algorithmes de routage classiques utilisent une route unique pour acheminer les paquets de leur source à leur destination. Pour utiliser au mieux les ressources du réseau, il peut être intéressant d'effectuer un routage multi chemin. En effet, les algorithmes de routage peuvent trouver plusieurs chemins ayant le même coût minimal ou des coûts faibles. Le choix de l'un des ces chemins est actuellement arbitraire. Des techniques, regroupées sous le nom de Equal Cost Multipath (ECMP), ont été proposées afin de répartir équitablement le trafic sur ces chemins. D'autres proposent une répartition inégale sur les différentes routes (Optimized Multipath, OMP). Il existe aussi des extensions permettant à OSPF d'effectuer un routage sur plusieurs routes. OSPF-ECMP [MOY98] permet de diviser le trafic en parts égales sur les chemins de coût minimum.

Puisque ces techniques de routage ne sont pas sensibles à la charge, il n'y a pas de risque d'oscillations. Toutefois, Curtis Villamizar [VIL99a] a mis en évidence les défauts d'OSPF-ECMP, qui provoquent des congestions dans certains cas. Il décrit OSPF-OMP, qui utilise les LSA utilisés par OSPF pour informer les routeurs des changements de topologie, afin de diffuser, grâce à un processus d'inondation, des informations concernant la charge des liens dans le réseau. De même, Villamizar décrit dans [VIL99b] une extension de MPLS permettant le multiroutage. MPLS-OMP utilise les informations diffusées par OSPF-OMP pour créer de nouveaux LSP. La charge envoyée sur ces chemins est ensuite ajustée graduellement.

II.4.3 Protocoles de routage utilisant plusieurs métriques

L'utilisation d'une métrique unique peut s'avérer insuffisante pour satisfaire les besoins de certaines applications. Par exemple, une application de visioconférence peut requérir d'une part, une certaine bande passante afin que la qualité de la retransmission vidéo soit correcte et d'autre part, un délai de bout en bout borné pour que les conversations ne soient pas entrecoupées de « blancs » désagréables pour les utilisateurs. Pour cela, Wang et Crowcroft proposent deux alternatives [WAN95].

La première possibilité est d'utiliser une métrique mixte, composée de plusieurs métriques primitives (telles que la bande passante, le délai, la gigue, etc.). Ainsi, pour tenir compte de la bande passante B, du délai D, et du taux de perte T, on peut construire la

métrique f, qui associe au chemin p la valeur $f(p)=\frac{B(p)}{D(p)\times T(p)}$. Il est toutefois difficile de considérer une telle métrique pour savoir si un chemin dispose des capacités à accueillir un flux requérant telle bande passante ou telle contrainte de délai. La valeur de cette métrique est en fait une heuristique rapportant globalement l'état d'un chemin.

La seconde possibilité est de considérer plusieurs métriques séparément. Cependant, il est prouvé dans [WAN95] que le problème consistant à trouver un chemin satisfaisant à deux contraintes parmi les suivantes : délai, taux de perte, coût et gigue, est NP-complet. Les combinaisons restantes sont donc celles associant la bande passante au délai, au taux de perte ou à la gigue. Dans ce cadre, les algorithmes suivants ont donc été proposés :

4. Widest-Shortest Path (WS) [APO99]: Cet algorithme détermine le chemin ayant un nombre de sauts minimum satisfaisant le débit du flux à acheminer. S'il existe plusieurs chemins faisables de même longueur, celui ayant la bande passante résiduelle la plus importante est sélectionnée.

5. Shortest-Widest Path (SW) [GUE96] : cet algorithme détermine le chemin le plus court parmi les chemins offrant la bande passante disponible la plus large.

II.4.4 Protocoles de routage basés sur l'apprentissage

II.4.4.1 Routage utilisant un réseau de neurones

Les algorithmes présentés ci-dessus utilisent l'algorithme de Dijkstra pour le calcul des différents chemins optimaux, mais d'autres algorithmes se basent sur un réseau de neurones pour la recherche du chemin optimal [GEL00], [PIE01]. Par exemple, Samuel Pierre et al [PIE00] utilisent un réseau de neurones de type Hopfield où chaque neurone est physiquement connecté à la totalité des autres neurones [HOP82]). Dans cette approche, la topologie du réseau est modélisée par un réseau de neurones où les coûts des liens représentent les poids des transactions entre les nœuds du réseau de neurones.

II.4.4.2 Routage adaptatif utilisant la technique d'apprentissage par renforcement

II.4.4.2.1 L'algorithme Q-Routing

Cet algorithme introduit en 1994 par Boyan et Littman [BOY94], est basé sur la technique d'apprentissage par renforcement (chapitre III). Cette technique est bien adaptée à la problématique du routage étant donné que le modèle d'environnement de chaque routeur est a priori inconnu. Cet algorithme recherche le plus court chemin en terme de temps d'acheminement des paquets jusqu à leurs destinations.

Pour estimer le temps d'acheminement de bout en bout, un routeur doit être capable de déterminer à tout moment :

- Le **temps de transmission**, c'est à dire le délai mis par un paquet pour atteindre le routeur suivant.
- Le **temps de traitement** à l'intérieur du routeur. Ce dernier est crucial et doit être le plus court possible.
- Le **temps d'attente,** c'est à dire le temps que va passer un paquet dans la file d'attente avant d'être émis. Contrairement aux délais précédents, qui sont à peu de chose près constants, les temps d'attente dans les files d'attente évoluent très rapidement en fonction du trafic.

C'est sur cette dernière estimation que se base le Q-Routing pour répondre aux objectifs de réactivité en détectant rapidement les changements de charge du réseau, l'apparition ou de la disparition d'une communication.

Dans le Q-Routing, le Q-Learning est utilisé pour apprendre une représentation de l'état du réseau en calculant les Q-valeurs qui vont permettre d'obtenir une politique de routage optimal et adaptatif. Pour cela, l'idéal est que chaque routeur dispose d'une vision globale de l'état du réseau à tout moment, c'est-à-dire de toutes les informations sur tous les routeurs. Cependant, on peut se rendre compte que la simple émission de cette information suffirait à saturer le réseau. Une solution possible consisterait à faire transiter le moins d'informations possibles sur le réseau, en limitant la vision et l'échange d'information d'un routeur uniquement à ses voisins.

Pour sauvegarder les informations de routage, chaque nœud x maintient une table des valeurs Q(x,y,d), appelé Q-table, où d est un élément de V, l'ensemble de tous les nœuds dans le réseau. y représente un élément de N(x), l'ensemble de tous les voisins du nœud x. D'après Boyan et Littman, la valeur Q(x,y,d) peut être interprétée comme le meilleur temps estimé par le routeur *x* pour qu'un paquet atteigne la destination *d* en passant par le routeur *y*. Ce temps n'inclut pas le temps d'attente dans la file d'attente de *x* mais inclus le temps de transmission δ, le temps d'attente dans la file d'attente de y, et le temps que le paquet met pour atteindre *d*, à partir du routeur *y* et en passant par le routeur *z* voisin de *y*.

Comme le choix d'une route est basé sur les Q-valeurs et que ces dernières ne représentent qu'une estimation, la décision de routage n'est pas forcement optimale. Il est alors nécessaire de mettre à jour les Q-valeurs afin de prendre en compte l'état réel du réseau. Boyan et Littman [BOY94] ont proposé le mécanisme de la mise à jour suivant :

Dès qu'un routeur x envoie un paquet *P* destiné au nœud *d* via l'un des routeurs voisins *y*, ce dernier envoi un paquet de renforcement (signal de renforcement) au routeur x. Ce paquet contient l'estimation optimale $Q(y,\tilde{z},d)$ du temps restant pour arriver à la destination *d*. Quand le routeur x reçoit cette estimation, il calcule la nouvelle Q-valeur $Q(x,y,d)$ comme suit:

$$\underbrace{Q(x,y,d)}_{nouvelle Valeur} = \underbrace{Q(x,y,d)}_{ancienne\ valeur} + \eta((\underbrace{q_y + \delta + Q(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{Q(x,y,d)}_{ancienne\ valeur}) \qquad \text{(II.1)}$$

où η représente le pas d'apprentissage (valeur comprise entre 0 et 1).

La méthode utilisée dans le Q-Routing, pour la mise à jour des Q-valeurs est connue sous le nom d'exploration avancée (forward exploration), où à chaque saut du paquet P(s; d), une Q-Valeur est mise à jour (Figure II.7).

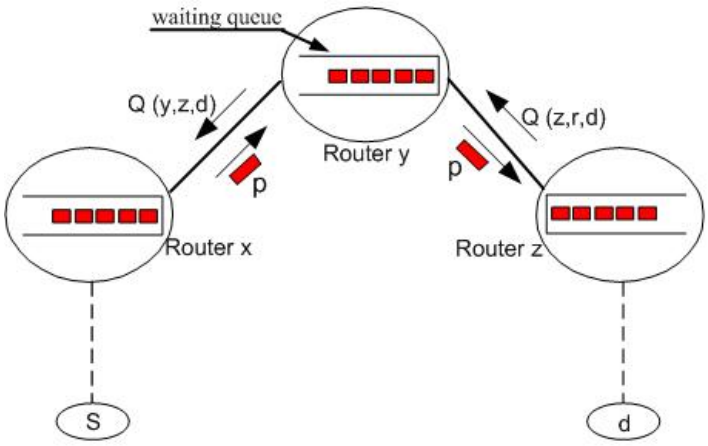


Figure II.7. Mise à jour des Q-Valeurs dans l'algorithme Q-Routing

Les performances de cette politique de routage dépendent principalement des Q-Valeurs estimées qui doivent être les plus représentatives possibles de l'état courant du réseau. Par conséquent, ces valeurs doivent être mises à jour de façon continue. Cependant, pour celles qui ne sont mises à jour que rarement, les décisions de routage sont peu fiables.

II.4.4.2.2 Confidence-based Q-Routing (CQ-Routing)

Cet algorithme est basé sur l'algorithme Q-Routing où la qualité d'exploration est améliorée en attachant des valeurs de confiance (C-valeurs) à chacune des Q-Valeurs dans le réseau [KUM98a], [KUM98b]. Ces C-valeurs sont utilisés pour déterminer le pas d'apprentissage utilisé pour la mise à jour des Q-Valeurs. De plus, ces C-Valeurs sont mises à jour en fonction des Q-Valeurs correspondantes, ce qui permet d'affiner la représentation de l'état du réseau.

Chaque Q-Valeur $Q(x, y, d)$ est associée à une valeur de confiance $C(x, y, d)$ qui est un nombre réel compris entre 0 et 1. Une valeur de confiance égale à 1 signifie que la Q-Valeur correspondante représente l'état courant du réseau avec un degré de fiable maximal, ce qui signifie que cette Q-Valeur a été mise à jour récemment. En revanche, une valeur de confiance égale à 0 signifie que la Q-Valeur correspondante est aléatoire et ne représente pas nécessairement l'état courant du réseau (valeur non fiable), En d'autres termes, cette Q-Valeur n'a jamais été mise à jour.

Dans l'algorithme Q-Routing standard, rien ne permet de dire si les Q-Valeurs utilisées dans la prise de décision du routage sont fiables ou non (si elles ont été mises à jour

récemment ou non). De plus, le pas d'apprentissage est constant tout au long de l'apprentissage. Dans CQ-Routing, les valeurs de confiance réagissent à la précision des Q-Valeurs, et le pas d'apprentissage dépend de la valeur de confiance de la Q-Valeur et de sa nouvelle estimation [KUM98a], [KUM98b]. En effet, quand un routeur x envoie un paquet P(s,d) à son voisin y (Figure II.8), il reçoit en retour non seulement l'estimation $Q(y,\tilde{z},d)$, mais aussi la valeur de confiance $C(y,\tilde{z},d)$ associée à cette Q-valeur. Quand le routeur x met à jour sa valeur $Q(x,y,d)$, il calcule en premier le pas d'apprentissage η qui dépend de $C(x,y,d)$ (ancienne valeur de confiance) et de $C(y,\tilde{z},d)$ (valeur de confiance estimée). La mise à jour de $Q(x,y,d)$ est formulée comme suit:

$$\underbrace{Q(x,y,d)}_{nouvelleValeur} = \underbrace{Q(x,y,d)}_{ancienne\,valeur} + \eta(\underbrace{C(x,y,d)}_{C_{anc}}, \underbrace{C(y,\tilde{z},d)}_{C_{est}})(\underbrace{(q_y + \delta + Q(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{Q(x,y,d)}_{ancienne\,valeur}) \quad \text{(II.2)}$$

La valeur de la fonction du pas d'apprentissage $\eta(C_{anc}, C_{est})$ doit être élevée si la valeur de Confiance correspondant à l'ancienne Q-valeur est basse ou la valeur de Confiance correspondant à la Q-valeur estimée est haute.

Ainsi, la fonction du pas d'apprentissage s'écrit:

$$\eta(C_{anc}, C_{est}) = \max(C_{est}, 1 - C_{anc}) \quad \text{(II.3)}$$

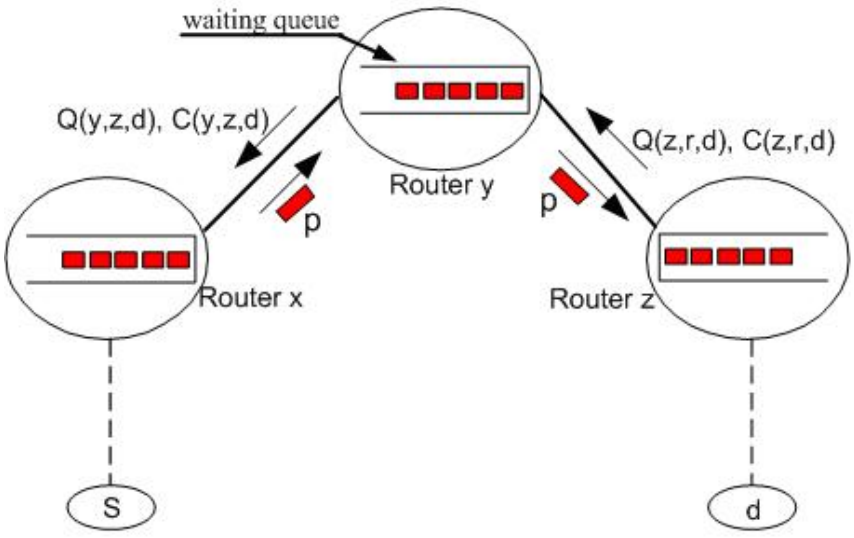


Figure II.8 : Mise à jour des Q-Valeurs dans l'algorithme CQ-Routing

La valeur de confiance (à l'exception des états initiaux) continue à changer avec le temps pour augmenter la précision des Q-valeurs correspondantes (augmenter la fiabilité des

Q-valeurs). Selon qu'une Q-valeur ait été mise à jour ou pas dans l'étape d'apprentissage précédente, les règles de mise à jour de C-valeur différent:

1. Si la Q-valeur correspondante n'est pas mise à jour dans l'étape d'apprentissage précédente alors :

$$\underbrace{C(x,y,d)}_{nouvelle\ valeur} = \lambda \underbrace{C(x,y,d)}_{ancienne\ valeur} \tag{II.4}$$

où λ et une constante d'affaiblissement comprise entre 0 et 1

2. Si une Q-valeur est mise à jour dans l'étape d'apprentissage précédente alors la C-valeur est mise à jour selon la relation suivante:

$$\underbrace{C(x,y,d)}_{nouvelle Valeur} = \underbrace{C(x,y,d)}_{ancienne\ valeur} + \eta(\underbrace{C(x,y,d)}_{C_{anc}}, \underbrace{C(y,\tilde{z},d)}_{C_{est}})(\underbrace{C(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{C(x,y,d)}_{ancienne\ valeur}) \tag{II.5}$$

II.4.4.2.3 Dual Reinforcement Q-Routing (DRQ-Routing)

DRQ-Routing [KUM97], [KUM98b], est basé sur la technique d'apprentissage par renforcement dual (Dual reinforcement learning) [GOE96]. Cet algorithme est une extension de l'algorithme Q-Routing auquel a été ajouté une autre direction d'exploration supplémentaire, « backward exploration » (Figure II.9). Dans la technique d'apprentissage par renforcement simple, l'élément essentiel est le signal de renforcement (section III). Ce dernier est envoyé à chaque fois qu'un paquet est reçu, ce qui augmente la charge du réseau. Ainsi, un tel type d'apprentissage n'est pas suffisamment adapté à la problématique du routage. Goetz et.al. [GOE96] ont développé l'algorithme d'apprentissage par renforcement dual (Dual reinforcement learning) qui au lieu de n'utiliser que le signal de renforcement direct, utilise en plus un signal du renforcement indirect, extrait de l'information envoyée par la source.

Contrairement au Q-Routing, dans le DRQ-Routing, quand un routeur x envoie un paquet P(s,d) à l'un de ses voisins y, le paquet contient l'information relative à la Q-valeur du routeur x. Quand le routeur y reçoit ce paquet, il utilise cette information pour mettre à jour sa Q-valeur qui concerne son voisin x. La Figure II.9 résume les deux explorations à chaque saut de paquet. Quand le routeur y doit prendre une décision de routage, il utilise les Q-valeurs correspondant au routeur voisin x.

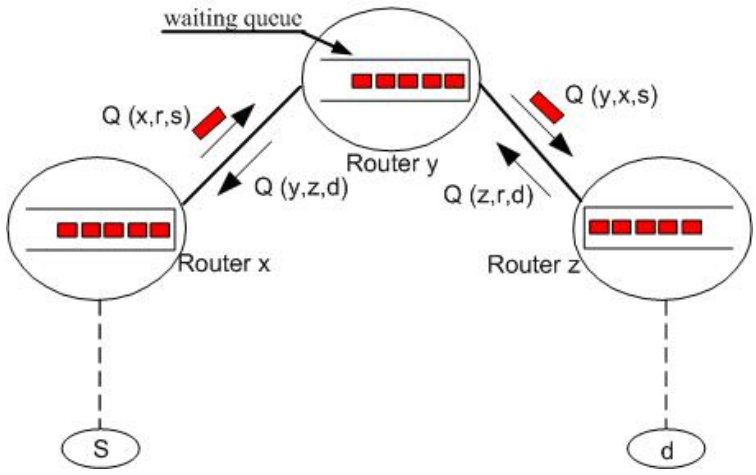


Figure II.9 : Mise à jour des Q-Valeurs dans l'algorithme DRQ-Routing

L'information contenue dans le paquet P(s; d) est le temps estimé $Q(x,\tilde{z},s)$ avec:

$$Q(x,\tilde{z},s) = \min_{z \in \text{voisins de } y} Q(x,z,s) \qquad \text{(II.6)}$$

Cette valeur représente une estimation du temps minimum qu'un paquet prendrait pour atteindre la source s, à partir du routeur x. Quand le routeur y reçoit cette évaluation, il calcule la nouvelle évaluation $Q(y,x,s)$ comme suit:

$$\underbrace{Q(y,x,s)}_{nouvelle Valeur} = \underbrace{Q(y,x,s)}_{ancienne\ valeur} + \eta_1((\underbrace{q_x + \delta + \underbrace{Q(x,\tilde{z},s)}_{nouvelle\ estimation}}_{} - \underbrace{Q(y,x,s)}_{ancienne\ valeur}) \qquad \text{(II.7)}$$

où $\eta_1$est le taux d'apprentissage pour l'exploration « backward ».

II.4.4.2.4 Confidence-based Dual Reinforcement Q-Routing (CDRQ-Routing)

L'algorithme CDRQ-Routing [KUM99], [KUM98b] combine les techniques des algorithmes CQ-Routing et DRQ-Routing. Ainsi, à chaque saut d'un paquet P(s; d), du routeur x au routeur y, les Q-valeurs et C-valeurs sont mises à jour lors des phases d'exploration forward et backward (Figure II.10).

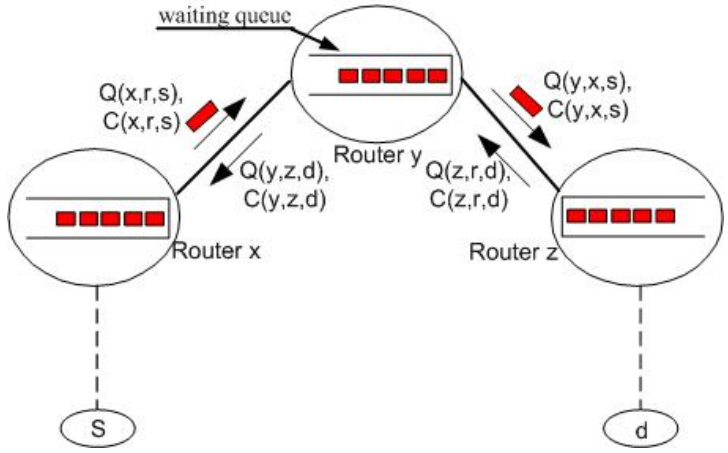


Figure II.10 : Mise à jour des Q-Valeurs dans l'algorithme CDRQ-Routing

Mise à jour de la Q-valeur du routeur x (forward exploration):

$$\underbrace{Q(x,y,d)}_{nouvelleValeur} = \underbrace{Q(x,y,d)}_{ancienne\,valeur} + \eta(\underbrace{C(x,y,d)}_{C_{anc}}, \underbrace{C(y,\tilde{z},d)}_{C_{est}})(\underbrace{(q_y + \delta + Q(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{Q(x,y,d)}_{ancienne\,valeur}) \quad \text{(II.8)}$$

Mise à jour de la C-valeur du routeur x (forward exploration):

$$\underbrace{C(x,y,d)}_{nouvelleValeur} = \underbrace{C(x,y,d)}_{ancienne\,valeur} + \eta(\underbrace{C(x,y,d)}_{C_{anc}}, \underbrace{C(y,\tilde{z},d)}_{C_{est}})(\underbrace{C(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{C(x,y,d)}_{ancienne\,valeur}) \quad \text{(II.9)}$$

Mise à jour de la Q-valeur du routeur y (backward exploration):

$$\underbrace{Q(y,x,s)}_{nouvelleValeur} = \underbrace{Q(y,x,s)}_{ancienne\,valeur} + \eta_1(\underbrace{C(y,x,s)}_{C_{anc}}, \underbrace{C(x,\tilde{z},s)}_{C_{est}})(\underbrace{(q_x + \delta + Q(x,\tilde{z},s)}_{nouvelle\ estimation} - \underbrace{Q(y,x,s)}_{ancienne\,valeur}) \quad \text{(II.10)}$$

Mise à jour de la C-valeur du routeur y (backward exploration):

$$\underbrace{C(y,x,s)}_{nouvelleValeur} = \underbrace{C(y,x,s)}_{ancienne\,valeur} + \eta(\underbrace{C(y,x,s)}_{C_{anc}}, \underbrace{C(x,\tilde{z},s)}_{C_{est}})(\underbrace{C(x,\tilde{z},s)}_{nouvelle\ estimation} - \underbrace{C(y,x,s)}_{ancienne\,valeur}) \quad \text{(II.11)}$$

II.4.4.2.5 Ant-based routing

Dans les algorithmes basés sur le Q-Routing, la mise à jour des tables de routage dépend de l'envoi de paquets entre une source et une destination. Pour cela, Subramanian et al. ont proposé l'algorithme Ant-based routing [SUB97], qui est inspiré de l'observation des fourmis apprenant le plus court chemin de la fourmilière à une source de nourriture (Figure II.11). À la différence du Q-Routing, cet algorithme emploie des tables probabilistes qui maintiennent pour chaque destination d une entrée ayant la forme $(d,(y_1,p_1),(y_2,p_2)\ldots(y_n,p_n))$, où p_i est la

probabilité pour choisir le routeur voisin y_i. Dans Ant-based routing, des messages appelés Ants sont utilisés pour fournir un signal de renforcement qui permet la mise à jour de la table de routage et de la table des probabilités. Ces messages doivent contenir trois informations : le nœud source s, le nœud de destination d et le coût c du chemin parcouru.

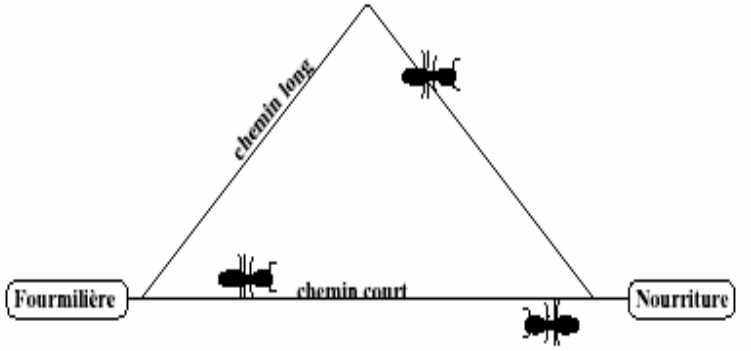


Figure II.11 : Recherche du plus court chemin chez les fourmilles

Pour explorer le réseau, un nœud S envoie un message Ants à une destination choisie aléatoirement. Un routeur qui reçoit ce message, met à jour en premier l'entrée correspondant au nœud s dans la table de routage, puis met à jour les probabilités concernant le même nœud. C'est une forme de backward exploration qui permet de minimiser la circulation des messages Ants. Spécifiquement, quand un message Ants M(C,S,D) arrive le long de l'interface I_k au routeur r, ce dernier met à jour c en premier (le coût accumulé par le message Ants), inclue le coût de l'interface I_k et envoie le message au routeur suivant. Cependant, r met à jour aussi la probabilité correspondant au nœud s, en augmentant légèrement la probabilité pour l'interface I_k et en diminuant les probabilités pour les autres interfaces. En particulier, la mise à jour de la probabilité p_k correspondant à l'interface I_k s'écrit:

$$P_k = \frac{P_k + \Delta P_k}{1 + \Delta P_k} \qquad \text{(II.12)}$$

alors que les autres probabilités sont ajustées selon la relation suivante :

$$P_j = \frac{P_j}{1 + \Delta P_k} \qquad \text{(II.13)}$$

Où $\Delta P_k = 1 / f(c)$

II.4.4.2.6 Les paquets cognitifs

De manière similaire, l'approche développée par Gelenbe et al. [GEL04], suggère de déployer une architecture réseau basée sur un concept cognitif : les informations concernant le routage ne seront plus stockées au niveau des nœuds de transmission mais acheminées au travers de paquets appelés paquets cognitifs (PC). Ces derniers sont de trois types :

- Des paquets d'exploration : Ces paquets sont amenés à tracer un circuit dans un réseau IP répondant à une qualité de service donnée. Ils fonctionnent pour chaque couple source-destination.

- Des paquets d'acquittement : Chaque fois qu'un paquet d'exploration arrive à destination, celle-ci génère un paquet d'acquittement. Celui-ci est acheminé au nœud source en suivant le chemin inverse étiqueté par le paquet d'exploration, il informe celui-ci sur le chemin que peuvent suivre les paquets de données.

- Des paquets de données : Ils servent à transporter les différents flux d'information.

L'approche développée par Gelenbe et al. reprend un concept proche du mode orienté connexion. L'échange des flux est précédé par l'établissement d'un circuit entre la source et la destination sauf que celui-ci n'est pas le même tout au long de la connexion. On parle donc d'un circuit adaptatif. Les paquets d'exploration servent à remonter l'information sur l'état du trafic dans le réseau faisant ainsi un routage adaptatif. Ils utilisent la technique de l'apprentissage par renforcement par réseaux de neurones que nous verrons dans le chapitre III pour la mise à jour des paramètres du modèle.

II.5 Analyse critique et conclusion

Nous avons montré dans le chapitre I la nécessité d'introduire des solutions de routage adaptatif dans un réseau IP, permettant de prendre en compte des contraintes de QoS. Ce type de réseau est caractérisé par l'hétérogénéité de ses liens, pouvant provoquer des goulots d'étranglement, ainsi que par des conditions de trafic dynamiques. Pour ce faire, une approche de routage adaptatif doit pouvoir prendre en compte rapidement toute modification des conditions de trafic tout en minimisant le temps de transfert de bout en bout. En effet, ce dernier fait intervenir des temps de transmission qui sont connus et

dépendant des liens entre les routeurs, des temps de traitement a priori connus et des temps d'attente des paquets dans les routeurs. Les temps d'attente sont a priori inconnus et évoluent très rapidement en fonction de la dynamique du réseau.

Dans ce chapitre, nous avons présenté les différentes approches de routage avec qualité de service. Les algorithmes correspondant ont été classés en quatre catégories : Ceux basés sur le principe de la commutation, ceux dérivés des protocoles classiques de routage, ceux utilisant plusieurs métriques et enfin ceux basés sur l'apprentissage. Les approches MPLS et QOSPF sont d'une part, des algorithmes non adaptatifs puisqu'ils ne tiennent pas compte des fluctuations de la charge du réseau et d'autre part, ne déterminent pas systématiquement le chemin optimal minimisant le temps de bout en bout. En effet, MPLS s'appuie sur des algorithmes classiques de routage pour déterminer la route que les paquets doivent emprunter pour atteindre leur destination. QOSPF quant à lui n'autorise pas le remplacement de l'itinéraire choisi car ce dernier est imposé par la source et cela même si un autre chemin s'avère meilleur. A l'opposé, les approches de routage basées sur l'apprentissage par renforcement sont adaptatifs et prennent en compte les fluctuations de la charge du réseau. Ce type d'apprentissage est bien adapté à la problématique du routage puisque le modèle représentant l'environnement (le réseau) dans lequel se situe le routeur est a priori inconnu. Cependant, l'efficacité de ces approches dépend fortement des informations utilisées par chaque routeur concernent la charge du réseau. Ces informations doivent être suffisantes et pertinentes pour refléter la charge réelle du réseau au moment de la prise de décision de routage.

Dans ce cadre, les algorithmes Q-Routing et DRQ-Routing utilisent une technique de mise à jour qui consiste à récupérer les informations de l'état du réseau à chaque fois qu'un routeur envoi des paquets à l'un de ces voisins. Boyan et Littman [BOY94] ont montré que cette technique permet de s'adapter aux variations de charge. Cependant, on constate que cette technique ne mène pas forcement à une politique de routage optimal et cela pour plusieurs raisons :

- Le changement de route ne se fait que si la route initialement choisie se détériore. Dans le cas où cette dernière redevient optimale, le routeur n'en tient pas compte.

- Elle ne permet pas l'exploration des autres chemins, ce qui laisse penser que les routeurs ne choisissent pas forcement le chemin optimal.

- Certaines informations ne sont mises à jour que rarement, ce qui les rend peu fiables lorsqu'il s'agit de les utiliser pour une décision de routage.

Les algorithmes CQ-Routing et CDRQ-Routing ont été proposés pour apporter des améliorations à la technique de mise à jour du Q-Routing en introduisant une valeur de confiance associée à l'état du réseau. Ceci permet d'une part, de fiabiliser les informations utilisées pour le routage et d'autre part, d'explorer tous les chemins possibles. Cependant, cette approche exhaustive ne constitue pas forcement un choix judicieux car dans certains cas, certains chemins sont explorés inutilement, ce qui ralentit la convergence vers un routage optimal. En effet, dans un réseau, plusieurs chemins peuvent comporter des routeurs constituant des goulots d'étranglement ou contenir des boucles. De plus, pour le choix du chemin optimal, tous ces algorithmes sont basés sur une métrique simple qui est le temps de bout en bout. L'ajout d'autres métriques s'avère complexe aussi bien du point de vue de la formulation que de la mise en oeuvre.

L'objectif de ce travail de thèse est de proposer deux nouvelles approches algorithmiques pour le routage adaptatif basé sur l'apprentissage. Ces approches permettent de remédier aux inconvénients des techniques utilisant le Q-Routing du point de vue de la minimisation du temps de converge vers la solution optimale de routage et de la réduction de l'espace mémoire occupé par la table de routage.

Le premier algorithme que nous appelons Q-Neural Routing, est basé sur un modèle neuronal permettant une modélisation de la dynamique du réseau. Cette méthode permet un apprentissage discriminant et une prise en compte fiable du contexte du réseau, pour l'estimation des temps d'acheminement des paquets. L'algorithme proposé a pour objectif d'une part, de prendre en compte en plus du temps de bout en bout, d'autres paramètres tels que l'état des files d'attentes ou la nature des flux et d'autre part, de minimiser l'espace mémoire occupé par la table de routage.

Le second algorithme que nous appelons K-Shortest paths Q-Routing, combine la technique de recherche des K plus courts chemins et l'algorithme du Q-Routing. En plus de la minimisation de l'espace mémoire occupé par la table de routage, un des objectifs visés est

de réduire l'espace d'exploration aux K plus courts chemins afin de minimiser le temps de convergence. Il s'agit aussi d'introduire dans la prise de décision, de nouveaux paramètres tels que le nombre de sauts ou la bande passante.

Chapitre III

Réseau de neurones et apprentissage par renforcement

III.1 Introduction

Dans ce chapitre, nous présentons les concepts et outils mathématique sur lesquels nous nous sommes appuyés pour développer les approches de routage adaptatif que nous proposons. Nous développerons tout d'abord le modèle connexionniste puis les différentes méthodes d'apprentissage. Notre objectif est de définir la méthode d'apprentissage qui répond le mieux à la problématique du routage adaptatif à QoS dans un réseau à trafic irrégulier.

III.2 Réseaux de neurones

L'essor des neurones artificiels a débuté par la proposition du premier neurone formel par McCulloch et Pitts [MCC43]. Puis les travaux de Rosenblatt, ont concerné un neurone formel doté de caractéristiques d'adaptabilité, appelé "Perceptron". Ce perceptron a permis de comprendre que les réseaux de neurones artificiels possèdent d'intéressantes capacités d'apprentissage qui peuvent être utilisées pour la résolution de certains problèmes.

Les années soixante et soixante-dix ont constitué une période extrêmement riche où des chercheurs comme Minski et Papert [MIN68], et bien d'autres ont mené une étude détaillée sur le perceptron. Le renouveau apparaît dans les années quatre-vingt grâce aux théories de Hopfield sur les neurones artificiels [HOP82]. Ces dernières années, les réseaux de neurones artificiels ont suscité un intérêt grandissant dans tous les domaines où interviennent les notions de décision, de classification, de reconnaissance des formes et d'optimisation.

III.2.1 Réseaux de neurones réels

Le neurone (cellule nerveuse) est l'élément de base du cerveau. Il reçoit l'influx nerveux par des ramifications courtes et nombreuses (dendrites), et transmet à son tour les influx nerveux par une fibre unique ramifiée à son extrémité (axone). L'information est transmise d'une cellule à l'autre en des points de contacts spécialisés (synapses). Les neurotransmetteurs chimiques, libérés dans la synapse, modifient le potentiel intracellulaire, augmentant ou diminuant ainsi la probabilité que celui-ci donne naissance à l'influx nerveux. Si le potentiel intracellulaire est en dessous d'un certain seuil, le neurone est au repos. Dans le cas contraire, le neurone envoie des signaux électriques. Typiquement, le neurone biologique comprend les éléments suivants (FigureII.1):

6. Un corps cellulaire contenant le noyau ;

7. Des dendrites, ou neurorécepteurs ;

8. Un axone qui permet le transfert de signaux à travers les synapses.

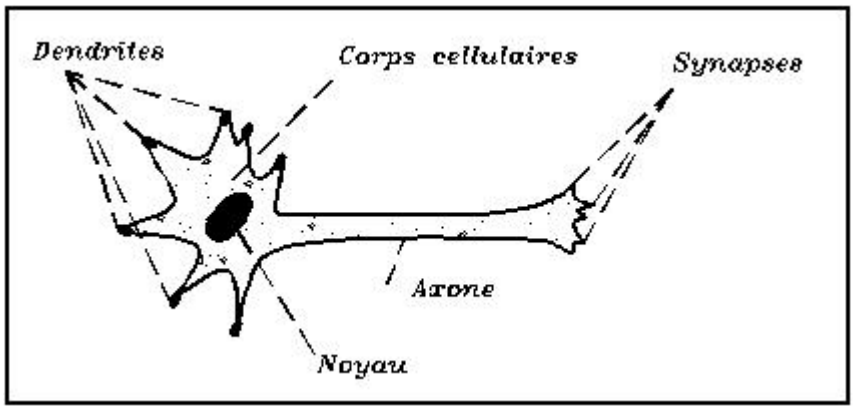


Figure III.1 : Schéma d'un neurone réel

III.2.2 Modèle d'un neurone formel

Un neurone formel (Figure II.2) est une entité mathématique simple qui calcule son potentiel V en déterminant la somme pondérée de ses entrées x_j [MCC43]. Sur la Figure III.2, les $\{w_{ij}\}$ représentent les coefficients de pondération, appelés aussi poids synaptiques. La sortie S du réseau est en général une fonction non linéaire du potentiel $S=f(v)$.

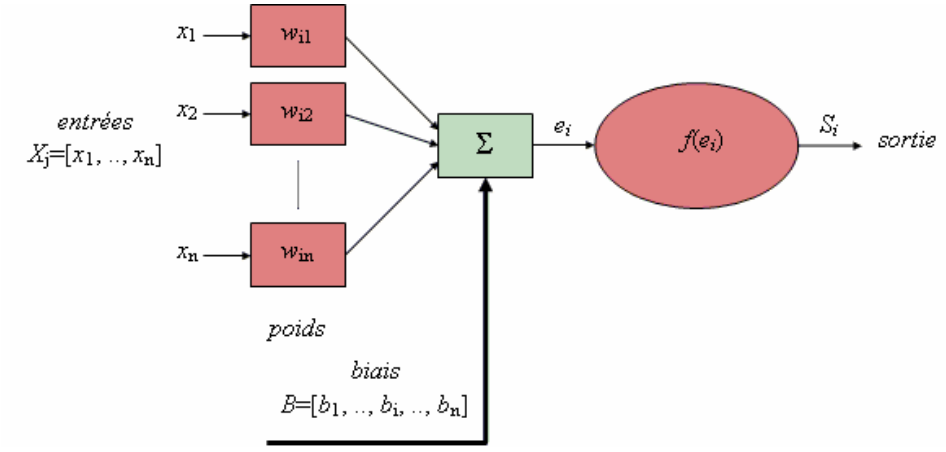


Figure III.2 : Structure interne d'un neurone formel

L'apprentissage consiste à trouver les valeurs des coefficients $\{w_{ij}\}$ afin que le réseau de neurones remplisse la fonction qui lui est demandée : classification, prédiction, etc.

III.2.3 Domaines d'application des réseaux de neurones

Les réseaux de neurones sont utilisés pour modéliser un processus par apprentissage des paramètres internes. Du fait de la non-linéarité des réseaux, différentes techniques itératives sont souvent employées pour accomplir l'adaptation des paramètres. Une fois cette phase terminée, s'ensuit la phase de généralisation ou de reconnaissance. Il s'agit d'exploiter le réseau en lui présentant des données inconnues. Sa tâche consiste à fournir une sortie en fonction des entrées et des paramètres initialement calculés. Notons que les phases d'apprentissage et de reconnaissance peuvent être théoriquement entrelacées.

Dans la phase d'apprentissage, selon la nature des entrées, des sorties désirées et des techniques d'adaptation utilisées, nous distinguons plusieurs types de tâches accomplies par le modèle.

Auto-association : La tâche consiste à effectuer des associations entre un vecteur d'entrée x et un vecteur de sortie y (Figure III.3). Si de plus les sorties sont associées à elles-mêmes, l'association est dite auto-associative. L'intérêt d'une telle mémoire réside d'une part, dans sa résistance au bruit (On espère ainsi que présenter une entrée bruitée produira en sortie une version non bruitée des entrées), et d'autre part, dans sa compression de l'information. Cette forme est très utilisée dans le traitement des images [LON90].

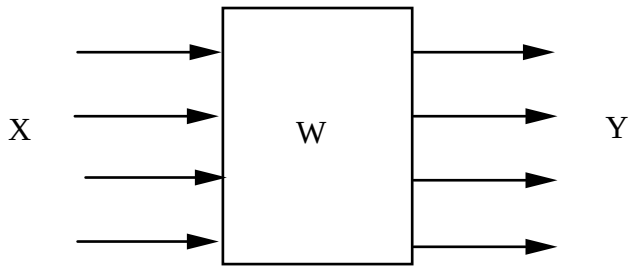


Figure III.3: Mémoire associative

- **Classification** : C'est la tâche pour laquelle on note le plus d'applications des modèles connexionnistes. Il s'agit ici d'un cas d'hétéro-association, où on entraîne un réseau à associer entre eux, des vecteurs différents. Le but est de faire étiqueter un vecteur d'entrée par une classe choisie parmi un ensemble fini. Ainsi par exemple, pour une application de reconnaissance automatique de caractères, indépendamment de la police utilisée, on fait correspondre à chacune des lettres de

l'alphabet la même classe quelle que soit la police utilisée. La représentation la plus communément utilisée consiste à décrire les classes par des vecteurs dans l'espace $\{0, 1\}^p$ ou $\{-1, 1\}^p$, p étant le nombre total de classes.

- **Prédiction :** Cette tâche consiste à former un modèle interne du processus, de façon à générer les points futurs d'une série de données. Le modèle apprend à prédire dans le futur, le comportement d'un système en fonction de son comportement passé. Des modèles à prédiction linéaire ont souvent été appliqués aux systèmes d'identification et de contrôle [PUE92]. On peut noter aussi l'application de modèles non-linéaires pour cette tâche dans le domaine de la reconnaissance de la parole [MEL92], [TEB90], [ISO90] ou l'identification du locuteur [ART93], [BEN98].

III.2.4 Réseau de neurones et apprentissage

Soit un réseau de neurones composé d'une couche d'entrée, d'une couche cachée et d'une couche de sortie, et soient :

- x_k : le vecteur à I éléments représentant le $k^{\text{ième}}$ stimulus. La matrice IxK des K stimuli à apprendre est notée X.

- h_k : le vecteur à L éléments représentant la réponse des L cellules de la couche cachée lorsque le $k^{\text{ième}}$ stimulus est présenté en entrée (la couche cachée comporte L cellules).

- o_k : le vecteur à J éléments représentant la réponse des cellules de la couche de sortie pour le $k^{\text{ième}}$ stimulus (la couche de sortie comporte J cellules).

- t_k : le vecteur à J éléments représentant la réponse désirée des cellules de la couche de sortie pour le $k^{\text{ième}}$ stimulus. La matrice des réponses désirées de dimension $J\times K$ est notée T.

- W : la matrice de dimension $L\times I$, des valeurs des connexions reliant les cellules de la couche d'entrée à celles de la couche cachée. L'élément $w_{l,i}$ représente la valeur de la connexion entre la $i^{\text{ième}}$ cellule d'entrée et la $l^{\text{ième}}$ cellule de la couche cachée.

- Z : la matrice de dimension $J \times L$, des valeurs des connexions reliant les cellules de la couche cachée à celles de la couche de sortie. $z_{l,j}$ représente la valeur de la connexion entre la $l^{ième}$ cellule de la couche cachée et la $j^{ième}$ cellule de sortie.

En notant a_n, l'état d'activation de la cellule n (qui peut être une cellule de la couche cachée ou de la couche de sortie), la réponse de la cellule, notée o_n, est :

$$o_n = f(a_n) \tag{III.1}$$

où f représente une fonction de transfert (fonction d'activation). Une des fonctions les plus utilisées est la fonction sigmoïde, définie par :

$$f(x) = \frac{1}{1+e^{-x}} \tag{III.2}$$

La figure ci dessous (Figure II.3), représente un exemple de réseau de neurones à deux entrées, deux sorties et à une couche cachée à trois neurones.

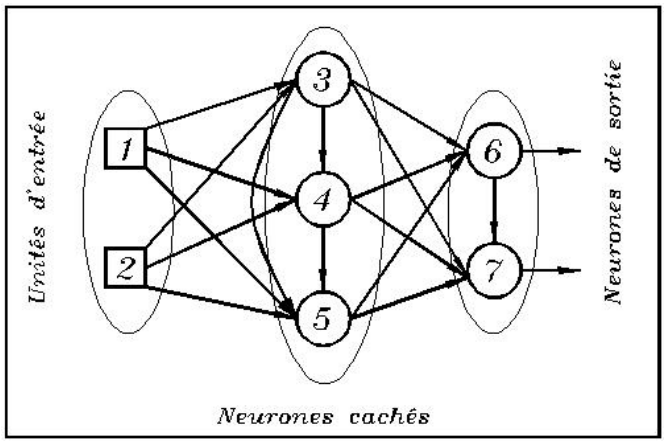


Figure III.4 : Exemple de réseau de neurones

Au cours de l'apprentissage, les calculs relatifs à la mise à jour des poids synaptiques se fait généralement, en utilisant la méthode du gradient, où on distingue deux étapes : la propagation directe et la rétro-propagation.

Hornik [HOR91] et Cybenko [CYB89] ont montré qu'un réseau de neurones à trois couches (couche d'entrée, couche cachée et couche de sortie) peut approcher n'importe

quelle fonction continue, à la seule condition qu'il ait suffisamment de cellules et que la fonction de transition de ses neurones formels satisfasse certaines propriétés. Cependant, le nombre total de cellules requises peut fort bien être plus faible avec un réseau de neurones à plus de trois couches pour approcher la même application avec la même précision. Il n'existe pas d'architecture qui soit optimale pour tous les problèmes, mais plutôt des architectures qui soient statistiquement optimales pour des classes de problèmes. Ajoutons à ceci qu'un très grand nombre d'articles présentent des règles plus ou moins intuitives permettant d'anticiper les capacités du réseau de neurones à représenter tel ou tel type d'application. Un bon exemple de ce genre de présentation est l'article [LIP87] où les arguments avancés sont de nature uniquement expérimentale.

III.2.4.1 Calcul du gradient par propagation directe

Pour répondre à un stimulus, le signal est propagé de la couche d'entrée à la couche de sortie en passant par la couche cachée. Les cellules de la couche cachée calculent leur activation et la transforment en réponse en utilisant leur fonction de transfert. Puis elles transmettent cette réponse aux cellules de la couche de sortie qui, à leur tour, calculent leur activation et la transforment en réponse. Ainsi, lorsque le *k-ième* stimulus est présenté en entrée, le vecteur de sortie des cellules de la couche cachée h_k prend la forme suivante :

$$h_k = f_1(Wx_k) \tag{III.3}$$

La réponse des cellules de la couche de sortie o_k, sera définie par :

$$o_k = f_2(Zh_k) \tag{III.4}$$

où f_1 et f_2 étant respectivement la fonction de transfert des cellules de la couche cachée et de la couche de sortie. Pour simplifier les notations, les cellules de la couche cachée et de la couche de sortie utilisent la même fonction de transfert *f*.

III.2.4.2 Calcul du gradient par la méthode de rétro-propagation

Le comportement du réseau dépend des matrices de connexions *W* et *Z*. Si le réseau ne donne pas la réponse attendue, on peut modifier les matrices de connexions afin d'améliorer la performance du réseau. La méthode de rétro-propagation introduite par Werbos [WER74]

est une technique d'apprentissage qui permet de réaliser cette procédure. Elle modifie l'intensité des connexions de manière à diminuer l'intensité de l'erreur commise par la cellule pour la réponse considérée. La procédure de prise en compte de l'erreur est identique pour toutes les couches, mais l'estimation du signal d'erreur diffère suivant les couches.

Pour les cellules de la couche de sortie, l'erreur est évaluée en comparant la réponse donnée par la cellule avec la réponse théorique désirée. Cette erreur, pour le $k^{ième}$ stimulus, est donné par :

$$e_k = (t_k - o_k) \tag{III.5}$$

Ce vecteur d'erreur prend en compte l'erreur commise par la cellule et son état d'activation.

Concernant la couche de sortie, l'erreur est définie comme suit :

$$\delta_{sortie,k} = f'(Zh_k) \otimes (e_k) = o_k \otimes (1 - o_k) \otimes (t_k - o_k) \tag{III.6}$$

où $\otimes$ symbolise le produit terme à terme des vecteurs (produit de Hadamar) et I un vecteur unité. La matrice de connexions Z est mise à jour à chaque itération selon la relation suivante :

$$Z_{[t+1]} = Z_{[t]} + \alpha \delta_{sortie,k} h_k^T = Z_{[t]} + \Delta_t Z \tag{III.7}$$

où k est un scalaire choisi aléatoirement, et α, un nombre réel positif définissant le pas d'apprentissage.

Pour les cellules de la couche cachée, le signal d'erreur ne peut être évalué par comparaison avec une valeur idéale. Il est estimé comme une fonction du signal d'erreur provenant de la couche de sortie et de l'activation des cellules de la couche cachée. Précisément, le vecteur d'erreur pour les cellules de la couche cachée s'obtient ainsi :

$$\delta_{cachée,k} = f'(Wx_k) \otimes (Z_{[t]}^T \delta_{sortie,k}) = h_k \otimes (1 - h_k) \otimes (Z_{[t]}^T \delta_{sortie,k}) \tag{III.8}$$

Comme on peut le constater, le signal d'erreur s'obtient en propageant l'erreur de la couche de sortie à travers la couche cachée. L'apprentissage pour la couche cachée s'effectue

ensuite de façon similaire à celui de la couche de sortie. La matrice de connexions W est adaptée au cours des itérations comme suit :

$$W_{[t+1]} = W_{[t]} + \alpha\delta_{cachée,k}x_k^T = W_{[t]} + \Delta_t W \qquad \text{(III.9)}$$

III.2.5 Méthodes d'apprentissage

L'apprentissage est un processus qui vise à améliorer les performances d'un système sur la base des expériences passées. Cette amélioration repose, dans un premier temps, sur la capacité d'extraction de l'information à partir de ces expériences, et dans un second temps, sur l'exploitation de cette information en vue d'améliorer une certaine fonction de performance qui est à définir. En général, on entend par apprentissage la modification automatique des paramètres des systèmes ou plus rarement du nombre et de l'organisation des systèmes, afin d'adapter le traitement effectué à une tâche particulière. On distingue trois familles d'apprentissage en fonction de la nature des informations disponibles et du but recherché : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Dans la suite du chapitre, nous désignons par agent, l'entité ou le système soumis à un apprentissage.

III.2.5.1 Apprentissage supervisé

C'est le mode d'apprentissage le plus couramment utilisé. Dans ce cas, il est nécessaire de disposer d'un ensemble de couples de données {entrées du réseau ; sorties désirées}, appelées base d'exemples [PER90] [RUM86]. La différence entre la sortie du réseau et la sortie désirée donne ainsi une mesure d'erreur quantitative sur le calcul effectué par le réseau de neurones. Cette erreur est utilisée pour réaliser l'adaptation. La méthode d'apprentissage la plus utilisée est la rétro-propagation du gradient que nous avons exposée dans le paragraphe III.2.4.2.

III.2.5.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, seules les informations en entrée sont fournies au système. Celui-ci doit donc déterminer ses sorties en fonction des similarités détectées entre les différentes entrées, c'est-à-dire en fonction d'une règle auto - organisatrice. Le système est appelé donc, à découvrir les régularités présentes dans ces configurations qui peuvent servir à les diviser en groupes de configurations semblables.

III.2.5.3 Apprentissage par renforcement

Il existe beaucoup de problèmes où les sorties désirées que l'apprentissage supervisé exige sont difficile à spécifier. On ne dispose souvent que d'une information qualitative permettant l'évaluation de la réponse calculée. L'apprentissage par renforcement utilise cette évaluation pour améliorer les performances du système. Cette forme d'apprentissage constitue une méthode d'apprentissage par essais et erreurs.

III.3 Apprentissage par renforcement (AR)

C'est un apprentissage pour lequel seule une mesure qualitative de l'erreur est disponible [ZHA97] [KAE96b]. Dans ce cas, l'agent reçoit des stimuli de son environnement et réagit en choisissant une action adéquate pour son comportement. Sa réaction est alors jugée par rapport à un objectif prédéfini sous forme de note "récompense". L'agent reçoit cette "récompense" et doit l'intégrer pour modifier ses actions futures et parvenir, ainsi à un comportement optimal. Une action conduisant à une note négative sera, à l'avenir, et dans les mêmes conditions, moins utilisée qu'une action notée positivement.

L'AR résulte de deux principes simples, à savoir :

1. Si pour un état donné, une action cause immédiatement quelque chose de mauvais, alors le système apprend à ne plus faire cette action lorsqu'il se trouve dans cet état.

2. Si dans un état donné, toutes les actions possibles conduisent à quelque chose de mauvais, alors le système apprendra à éviter de se retrouver dans cet état.

L'AR consiste à apprendre pour un agent autonome un comportement à adopter lors de son interaction avec son environnement, afin d'atteindre des objectifs sans aucune intervention extérieure ou d'un professeur. La figure II.4 illustre le principe d'un apprentissage par renforcement [HAR96].

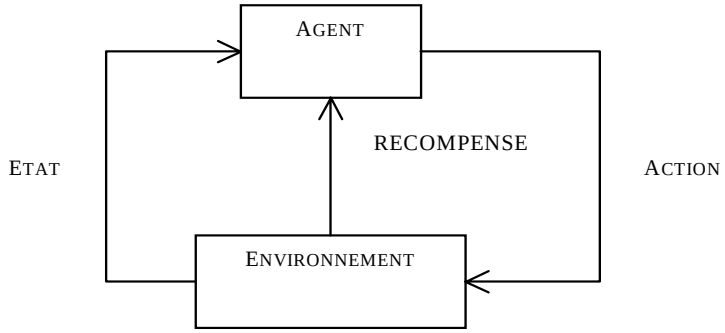


Figure III.5 : Modèle de l'apprentissage par renforcement

L'analyse des différentes méthodes d'apprentissage montre que l'apprentissage par renforcement est celui qui répond le mieux à la problématique du routage dynamique prenant en compte des contraintes de QoS. En effet, dans ce cas, le réseau est soumis à un trafic imprévisible et arrivant par rafale, et il est donc nécessaire qu'il fasse preuve d'une grande réactivité.

III.3.1 Modèle mathématique d'un AR

Soit un système défini par trois états, pour lequel chaque état peut effectuer deux actions possibles, à savoir a_1 et a_2 (Figure II.5) [SUT97].

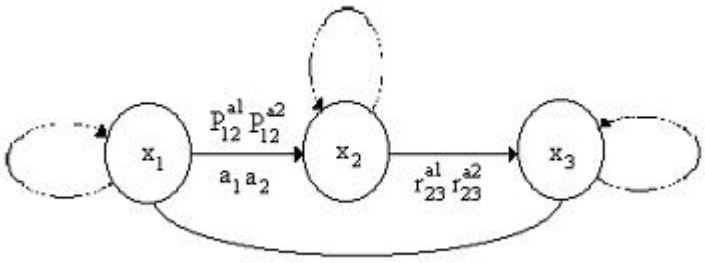


Figure III.6 : Exemple de Processus Markovien à 3 états et 2 actions

La transition d'un état à un autre est effectuée selon la probabilité P_{21}^{a1}, P_{21}^{a2}, aboutissant à un signal de récompense respectivement r_{23}^{a1}, r_{23}^{a2}.

Notations

Soient :

- X : L'ensemble des états du système et $x_t \in X$ un état du système à l'instant t.

- A_x : L'ensemble des actions possibles en se plaçant dans l'état x.

- $A = \bigcup_{x \in X} A_x$: L'ensemble de toutes les actions.

- $a_t \in A_x$: L'action appliquée à l'instant t pendant l'état x_t.

- $P_{xy}^{a} = \Pr\left(x_{t+1} = y \middle| x_t = x, a_t = a\right)$ définit la probabilité de la transition de l'état x à l'état y sous l'action a.

- R : L'ensemble des récompenses que le système peut recevoir de la part de l'environnement, tel que :

$$\begin{aligned} r : (X, X, A) &\to R \\ (x_t, y_t, a_t) &\mapsto r_{x_t y_t}^{a_t} \end{aligned} \qquad \text{(III.10)}$$

où r est la récompense retournée de l'environnement en appliquant l'action a_t lors de la transition d'état de $x_t \to y_t$.

- π : La politique adoptée, (La politique décidant de l'action a dans l'état x : $a = \pi(x)$) telle que :

$$\begin{aligned} \pi \quad : \quad X &\to A \\ x &\mapsto a \end{aligned} \qquad \text{(III.11)}$$

- Π : L'ensemble des politiques (appelées lois de commande), tel que :

V^{π} : La fonction de valeur d'états, telle que :

$$\begin{aligned} V^{\pi} : (X, \Pi) &\to R \\ (x, \pi) &\mapsto V^{\pi}(x) \end{aligned} \qquad \text{(III.12)}$$

- $V^{\pi}(x)$: La mesure de la qualité de la politique π dans l'état x.

- U : L'ensemble des couples état-action.

- Q^{π} : La fonction de valeur du couple état-action, telle que :

$$\begin{aligned} Q^{\pi}&: (U, \Pi) \to R \\ &(x_0, a, \pi) \mapsto Q^{\pi}(x_0, a) \end{aligned} \tag{III.13}$$

La fonction $Q^{\pi}(x_0, a)$ mesure la qualité de la politique π lorsque le système est à l'état x_0 et l'action a est choisie.

La définition des fonctions Q et V (Q-Valeur et V-Valeur) sont fondamentales car la politique optimale sera celle qui conduira à la meilleure qualité (V^* et Q^*), définie par :

$$\pi^* = \arg\max_{a \in A_x} Q^*(x, a) \tag{III.14}$$

La politique optimale est celle qui maximise le critère V^{π}. Les algorithmes d'apprentissage par renforcement ont donc pour objectif de maximiser de manière itérative la fonction π.

III.3.2 Les fonctions de valeurs

Dans ce paragraphe, nous présentons la relation qui existe entre les fonctions de valeur et les politiques de décision.

III.3.2.1 Récompenses

Une récompense à un temps donné r_t, est le signal reçu par l'agent lors de la transition de l'état x_t vers l'état x_{t+1}, pendant la réalisation de l'action a_t, définie par : $r_t = r_{xy}^a$

La récompense globale représente la somme de toutes les récompenses ponctuelles. Elle est utilisée pour l'évaluation du comportement du système à long terme. Lorsqu'un essai est fini, cette somme s'écrit sous la forme:

$$R_{et} = \sum_{k=0}^{T} r_{t+k+1} \tag{III.15}$$

et on définit une somme amortie des récompenses ponctuelles par:

$$R_{et} = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{III.16}$$

où $0 < \gamma < 1$ et T la fin de l'essai.

III.3.2.2 Définition des fonctions V -valeur et Q -valeur

La fonction de valeur, V -valeur, représente l'espérance de la somme globale des récompenses ponctuelles, sous la politique π . Le but de cette fonction est d'évaluer la qualité des états, sous une politique donnée. Elle est définie comme suit :

$$V^{\pi}(x) = E_{\pi}\left[R_{et} \middle| x_t = x\right] = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x\right] \tag{III.17}$$

De la même façon, une fonction Q -valeur, notée $Q^{\pi}(x,a)$, représente l'espérance de la récompense globale lorsque le système réalise l'action a sous la politique π , telle que :

$$Q^{\pi}(x,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x, a_t = a\right] \tag{III.18}$$

L'estimation de ces fonctions de valeur est nécessaire dans la plupart des algorithmes d'AR. Ces fonctions de valeur peuvent être estimées à partir d'expériences [SUT97] [LIT94].

III.3.2.3 Fonction de valeur et politique optimale

Une politique $\pi = \pi^*$ est dite optimale si pour $V^* = V^{\pi^*}(x) \geq V^{\pi^i}(x) \quad \forall x \in X$ et pour toutes les politiques π^i [LIT94].

L'objectif de l'apprentissage par renforcement est de trouver une politique optimale π^* qui maximise l'espérance de la récompense globale :

$$\pi^* = \arg\max_{\pi} V^{\pi}(x) = \arg\max_{\pi} E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| x_t = x\right] \tag{III.19}$$

L'expression de l'équation (III.17) peut se mettre sous la forme:

$$V^{\pi}(x) = \sum_{y \in X} P_{xy}^{\pi(x)} \left[r_{xy}^{\pi(x)} + \gamma V^{\pi}(y) \right] \qquad \text{(III.20)}$$

Cette équation signifie que l'utilité d'un état, sous la politique π, est une espérance sur tous les états disponibles de la somme de la récompense globale reçue en appliquant $\pi(x)$ et l'utilité de y sous la politique π.

Cette équation s'applique aussi pour toutes les politiques, en particulier à la politique optimale π^* :

$$V^*(x) = V^{\pi^*}(x) = \sum_{y \in X} P_{xy}^{\pi^*(x)} \left[r_{xy}^{\pi^*(x)} + \gamma V^{\pi^*}(y) \right] \qquad \text{(III.21)}$$

Par définition, toutes les politiques optimales ont la même fonction de valeur V^*. De ce fait, l'action choisie par la politique optimale doit maximiser l'expression suivante :

$$\sum_{y \in X} P_{xy}^{a} \left[r_{xy}^{a} + \gamma V^*(y) \right] \qquad \text{(III.22)}$$

Ce qui implique :

$$V^*(x) = \max_{a \in A_x} \sum_{y \in X} P_{xy}^{a} \left[r_{xy}^{a} + \gamma V^*(y) \right] \qquad \text{(III.23)}$$

Finalement, on peut déterminer la politique optimale comme suit:

$$\pi^*(x) = \arg\max_{a \in A_x} \sum_{y \in X} P_{xy}^{a} \left[r_{xy}^{a} + \gamma V^*(y) \right] \qquad \text{(III.24)}$$

Ces équations supposent la connaissance du modèle du système $\left(r_{xy}^{a}, P_{xy}^{a} \right)$. Cependant, dans le cas où ce modèle est inconnu, la fonction Q-valeur permet de résoudre ce problème.

L'expression $Q^{\pi}(x,a)$ peut se mettre sous la forme récurrente suivante :

$$Q^{\pi}(x,a) = \sum_{y \in X} P_{x}^{a} \left[r_{xy}^{a} + \gamma V^{\pi}(y) \right] \qquad \text{(III.25)}$$

La fonction $Q^* = Q^{\pi^*}$ correspond à la politique optimale, sans nécessairement choisir l'action $\pi(x)$ par la politique π.

Enfin, l'expression de V^* peut être écrite sous la forme (équation d'optimalité de Bellman) :

$$V^* = \max_{a \in A_x} \left[Q^*(x,a) \right] \tag{III.26}$$

et l'expression de $\pi^*(x)$ devient :

$$\pi^*(x) = \arg\max_{a \in A_x} \left[Q^*(x,a) \right] \tag{III.27}$$

Ces deux dernières équations peuvent être évaluées sans connaissance du modèle du système.

Le problème de l'AR revient donc à apprendre les Q-valeurs et V-valeurs, autrement dit à les estimer en l'absence de connaissance *a priori* sur le modèle.

III.3.3 Les méthodes d'apprentissage par renforcement

Plusieurs algorithmes ont été élaborés pour trouver la politique optimale, en utilisant ou non le modèle du système. Dans ce qui suit, nous présentons les principales méthodes : programmation dynamique, Différences temporelles (TD-Learning), apprentissage qualitatif (Q-Learning).

III.3.3.1 Programmation dynamique

La programmation dynamique est un ensemble de méthodes permettant de trouver les politiques optimales dans le cas où on dispose d'une connaissance parfaite du modèle de l'environnement [KAE96][SUT97], comme dans les Processus de Décision Markoviens. Elle permet de trouver la fonction de valeur issue d'une politique donnée ou bien, la politique optimale et la fonction de valeur correspondante.

Les algorithmes d'AR basés sur la programmation dynamique sont appelés généralement les algorithmes de programmation dynamique incrémentale. Ces algorithmes sont une

approximation de l'équation $V^* = \max_{a \in A_x} \left[Q^*(x,a) \right]$ (équation d'optimalité de Bellman) à partir d'essais. La programmation dynamique est basée sur deux méthodes : l'itération de valeur ou l'itération de politique [ZHA97b][SUT97]:

III.3.3.1.1 Itération de valeur

C'est une méthode qui permet de trouver une fonction optimale de valeur sans connaissance *a priori* de la politique optimale [HAR96] [KAE96]. Cette méthode a pour but de calculer $V^*(x)$ qui s'écrit :

$$V^*(x) = \lim_{k \to \infty} V_k^*(x) \qquad \text{(III.28)}$$

$V_k^*(x)$ est l'estimation de la fonction optimale de valeur pour k fini, i.e., que $V_k^*(x)$ est l'espérance maximum de la somme des récompenses si la tâche de décision est terminée en un ensemble k fini.

Pour $k = 1$, $V_k^*(x)$ est l'espérance de la récompense à un instant donné :

$$V_1^*(x) = \max_{a \in A_x} E\left[r_{xy}^a \right] \qquad \forall x \qquad \text{(III.29)}$$

Par récurrence, on peut écrire :

$$V_{k+1}^*(x) = \max_{a \in A_x} E\left[r_{xy}^a + \gamma \sum_y P_{xy}^a V_k^*(y) \middle| x_t = x, a_{t=a} \right] \qquad \forall x \qquad \text{(III.30)}$$

On notera que la politique n'est pas prise en compte dans cette équation.

III.3.3.1.2 Itération de politique

Cette méthode prend en compte la représentation d'une politique ainsi qu'une fonction de valeur. Elle met en œuvre une politique améliorée en utilisant cette représentation et la fonction de valeur. L'algorithme de la méthode d'itération de politique est formulé comme suit :

Soit $\pi := $, une politique initiale arbitraire, calculer V^{π}.

Répéter :

Calculer Q^{π} en utilisant :

$$Q^{\pi}(x,a) = \sum_{y \in X}\left[P_{xy}^{a}\, r_{xy}^{a}\right] + \gamma \sum_{y \in X}\left[P_{xy}^{a}\, V^{\pi}(y)\right]$$

Trouver μ (une politique donnée) en utilisant :

$$\mu(x) = \arg\max_{a \in A_x} Q^{\pi}(x,a) \qquad \forall x$$

et calculer V^{μ}.

Mettre: $\pi := \mu$ *et* $V^{\pi} := V^{\mu}$ et Si $V^{\pi} \neq V^{\mu}$, aller à 2.

III.3.3.2 Différences temporelles (TD-Learning)

La méthode TD-Learning [KAE96][SUT97][SUT88] est utilisée pour le traitement des problèmes à long terme lors de la prédiction. L'objectif est de prédire un événement à l'instant T à partir des instants $t_0, t_1, \ldots (T-1)$.

Notons $z(T) = z(m+1)$ cet événement et $s(t_0), s(t_1), \ldots, s(m)$, les prédictions calculées à partir des observations $u(t_0), u(t_1), \ldots, u(m)$.

Une première approche consiste à effectuer un apprentissage supervisé, pour déterminer la valeur $z(T)$. La fonction coût associée est :

$$j(t) = \frac{1}{2}\left[s(t) - z(T)\right]^2 \qquad \text{(III.31)}$$

avec :

$$s(t) = f(W,t) \qquad \text{(III.32)}$$

Ainsi, la méthode du gradient est utilisée pour minimiser la fonction coût $j(t)$:

$$\frac{\partial j(t)}{\partial W(t)} = [s(t) - z(T)]\frac{\partial s(t)}{\partial W(t)} \tag{III.33}$$

Pour trouver la meilleure prédiction, il suffit de trouver les meilleurs W . L'apprentissage, dans le cas supervisé, s'appuie sur l'algorithme de descente de Windrow-Hoff [MEL94][ABD94] :

$$W \leftarrow W + \alpha\Delta W \tag{III.34}$$

avec :

$$\Delta W_t = [z(T) - s(t)]\frac{\partial s(t)}{\partial W(t)} \tag{III.35}$$

Pour une séquence allant de $t = 1 \quad à \quad T$, on a :

$$W \leftarrow W + \alpha\sum_{t=1}^{m}\Delta W_t \tag{III.36}$$

Cette méthode présente un inconvénient puisqu'il faut attendre l'achèvement de la séquence pour connaître la cible $z(T)$ et par la suite corriger les paramètres.

Afin de comprendre cette méthode, nous allons considérer l'exemple suivant: Soit la chaîne de Markov illustrée Figure II.6 [HAR96], les deux états lundi et dimanche sont respectivement l'état initial et l'état final

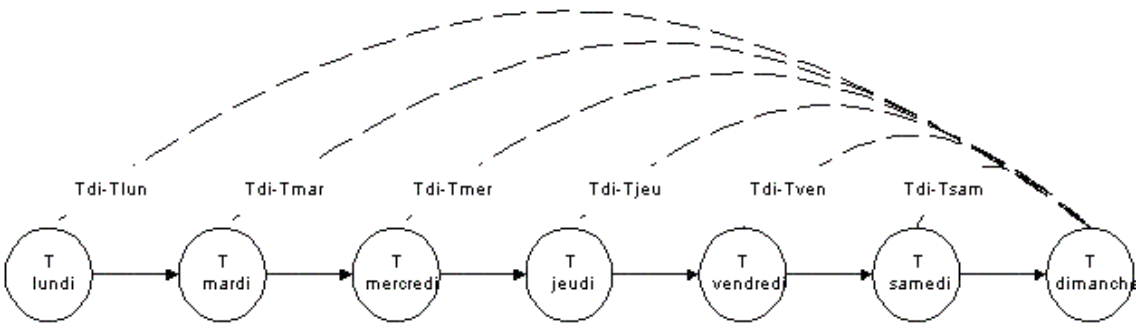


Figure III.7 : Prédiction par les méthodes conventionnelles

Cependant, pour les différences temporelles, l'erreur $s(t)-z(T)$ peut être écrite comme une suite de différences successives :

$$z(T)-s(t)=\sum_{k=t}^{m}\left[s(k+1)-s(k)\right] \tag{III.37}$$

Finalement, nous obtenons un modèle de ce type (Figure III.8)

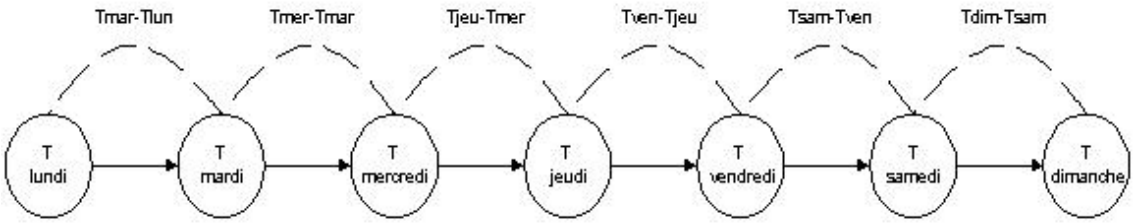


Figure III.8 : Prédiction par la méthode des différences temporelles

En combinant l'équation (39) et l'équation (37), on obtient :

$$\begin{aligned}\sum_{t=1}^{m}\Delta W_t &= \sum_{k=1}^{m}\sum_{t=t}^{m}\left[z(k+1)-s(k)\right]\frac{\partial s(t)}{\partial W(t)}\\ &= \sum_{t=1}^{m}\left[z(t+1)-s(t)\right]\sum_{k=1}^{t}\frac{\partial s(k)}{\partial W(k)}\\ &= \sum_{t=1}^{m}\Delta W_t^{'}\end{aligned} \tag{III.38}$$

Avec :

$$\Delta W_t^{'}=\left[z(t+1)-s(t)\right]\sum_{k=1}^{t}\frac{\partial s(k)}{\partial W(k)} \tag{III.39}$$

Ainsi, on peut à chaque instant t calculer les paramètres W selon la relation suivante :

$$W \leftarrow W+\alpha\,\Delta W_t^{'} \tag{III.40}$$

Dans ce cas, il est possible de modifier les paramètres après chaque pas de temps [SUT97][SUT88][ABD94].

On peut ajouter un terme de pondération exponentielle, $\lambda(0 \leq \lambda \leq 1)$ dans $\Delta W_t'$, tel que :

$$\Delta W_t' = [z(t+1) - s(t)] \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial s(k)}{\partial W(k)} \qquad \text{(III.41)}$$

Un choix correct des paramètres α et λ garantit la convergence de l'algorithme de différences temporelles, la convergence de la méthode TD est assurée pour α=0 [SUT88].

III.3.3.3 Q-Learning

Les deux méthodes présentées section III.3.3.1 posent deux contraintes majeures, à savoir, le besoin de disposer du modèle de l'environnement et la recherche exhaustive dans l'espace des états. De telles méthodes ne sont pas bien adaptées pour des problèmes complexes. La méthode du Q-learning [HAR96] [WAT89] [STE97a] [ZHA97b] [SUT97] est quant à elle une méthode d'apprentissage ne nécessitant pas de modèle, et son objectif est de trouver une approximation de la Q-fonction, notée Q, qui approche Q^*.

Cela passe par la résolution de l'équation d'optimalité de Bellman [MEL94][ABD94], telle que :

$$V^* = \max_{a \in A_x} \left[Q^*(x,a) \right] \qquad \text{(III.42)}$$

L'algorithme du Q-Learning peut se résumer ainsi :

Soit $t = 0 \quad et \quad x =$ un état initial.

Répéter :

1. Effectuer une action $a \in A_x$ qui conduit à l'état suivant y.
2. Ajuster Q, telle que :

$$Q(x,a) = Q(x,a) + \alpha\left[r_{xy}^{a} + \gamma \max_{b \in A_y} Q(y,b) - Q(x,a) \right]$$

3. Mettre $x = y$.

Watkins [WAT89] a montré que les valeurs de Q(x,a) tendent vers les valeurs optimales Q^* (x; a) avec une probabilité de 1.

La méthode du Q-learning possède des avantages évidents. Tout d'abord, elle ne nécessite aucune connaissance a priori du système. Ensuite, le point de vue de l'agent est très local, ce qui correspond bien à la problématique du routage dynamique où chaque routeur doit décider du chemin par lequel il envoi ses paquets sans connaître l'état de tous les routeurs. De plus, la mise en œuvre ne nécessite de disposer que d'une table de Q-valeurs comparable à une table de routage, et d'ajuster cette dernière après chaque action effectuée. Cependant, cet algorithme possède un inconvénient majeur que l'on retrouve d'une manière générale dans tous les problèmes d'apprentissage par renforcement : il s'agit de la garantie de la convergence de Q vers Q^*, pour laquelle des hypothèses fortes doivent être admises: le processus doit être markovien et stationnaire, et l'ensemble des couples états-actions (x,a) doivent être visités un nombre infini de fois. Cette dernière hypothèse n'est pas cruciale dans tous les sujets d'apprentissage par renforcement. En effet, dans des problèmes de robotique où le robot apprend à se déplacer dans un environnement donné ou dans des problèmes d'apprentissage de type apprentissage de jeux (backgammon, échec), on se donne une période d'apprentissage où le comportement de l'agent importe peu, et où on peut se permettre un grand nombre d'explorations. Ce n'est généralement qu'à la fin de cette période d'apprentissage que l'on poursuit les explorations ayant conduit aux meilleurs résultats. Dans

notre problème, l'environnement n'est pas stationnaire, mais dynamique et par conséquent il n'y a pas de séparation entre période d'apprentissage et période d'exploitation. Le nombre de visites des couples (x; a) devra donc être non seulement fini, mais suffisamment grand pour assurer la convergence vers Q^* sans tomber dans un minimum local, et suffisamment petit pour ne pas saturer le système. Pour les approches de routage adaptatif que nous proposons, l'étude d'une solution adaptée au problème de l'exploration est développée.

III.4 Conclusion

Dans ce chapitre, nous avons montré l'intérêt que représente l'apprentissage par renforcement s'appuyant sur la méthode du Q-learning pour répondre à la problématique du routage adaptatif sous contraintes de QoS, où chaque routeur doit décider du chemin par lequel il envoi ses paquets à partir d'une perception locale de l'état de ses voisins. Cependant, la mise en œuvre de cette technique d'apprentissage nécessite une solution appropriée au problème de l'exploration compte tenu de la nature dynamique et non stationnaire d'un réseau à trafic irrégulier.

Chapitre IV

L'algorithme Q-Neural Routing

IV.1 Introduction

Dans ce chapitre, nous présentons le premier algorithme de routage adaptatif appelé Q-Neural Routing inspiré de l'algorithme Q-Routing. Nous décrivons en particulier le mécanisme d'exploration utilisé pour la mise à jour des paramètres de routage et le modèle neuronal utilisé pour l'estimation des Q-valeurs. Ensuite, nous présentons l'outil de simulation OPNET utilisé ainsi que l'implémentation de l'algorithme Q-Routing sur lequel s'appuient nos deux approches algorithmiques. Nous analysons ensuite les performances de l'algorithme proposé en terme de délai d'acheminement moyen des paquets, pour différentes conditions de trafic ainsi que son comportement face à un changement de topologie du réseau.

IV.2 Position du problème

Nous avons montré dans le chapitre II que les algorithmes de routage adaptatif basés sur le principe du Q-Routing permettent de prendre en compte les variations de charge du réseau. Ces algorithmes utilisent une table pour sauvegarder les Q-valeurs de toutes les destinations, ce qui a pour conséquence la réservation d'un espace mémoire conséquent. De plus, pour le calcul du chemin optimal, tous ces algorithmes sont basés sur une métrique simple qui est le temps de bout en bout. L'ajout d'autres métriques s'avère complexe aussi bien du point de vue de la formulation que de celui de la mise en œuvre car l'utilisation d'une fonction de renforcement combinant plusieurs métriques est très complexe à mettre en oeuvre. De plus, l'efficacité de ces algorithmes dépend fortement des informations utilisées par chaque routeur. Ces informations doivent être suffisantes et pertinentes pour refléter la charge réelle du réseau au moment de la prise de décision de routage.

La technique de mise à jour utilisée dans l'algorithme Q-Routing est basée sur le principe de l'exploration avancée (forward exploration) [BOY94]. Elle consiste à utiliser les informations de l'état du réseau à chaque fois qu'un routeur envoi un paquet à l'un de ces voisins (Figure IV.1).

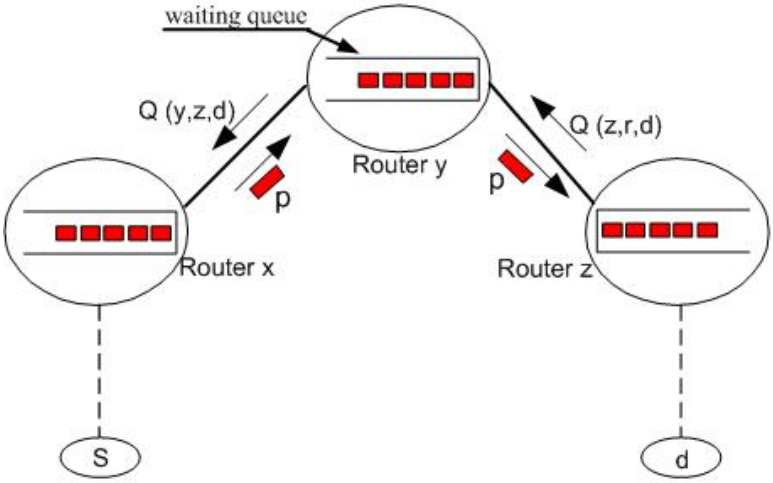


Figure IV.1 : L'exploration avancée (forward exploration).

Boyan et Littman [BOY94] ont montré que cette technique permet une bonne prise en compte de la montée en charge du réseau. Cependant, elle n'offre pas une adaptation suffisante en cas de baisse de charge, et cela pour plusieurs raisons :

4. Le changement de route ne se fait que si le coût associé à la route initialement choisie augmente. Dans le cas où cette dernière redevient optimale, le routeur n'en tient pas compte.

5. Elle ne permet pas l'exploration des autres chemins, ce qui laisse penser que les routeurs ne calculent pas forcément le chemin optimal.

6. Certaines Q-valeurs ne sont mises à jour que rarement, ce qui les rend peu fiables lorsqu'il s'agit de les utiliser pour une décision de routage.

Pour illustrer ce problème, analysons ce qui se passe dans un réseau à trois routeurs (Figure IV.2)

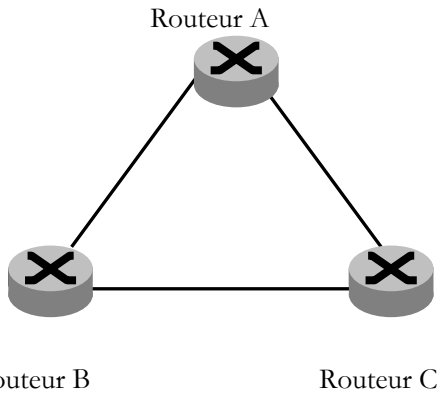


Figure IV.2 : Réseau utilisé pour illustrer le problème d'exploration

Lorsqu'un paquet doit aller du routeur *A* au routeur C, il est mis dans une file d'attente de traitement. Une fois arrivé dans l'unité de traitement, une estimation du temps nécessaire pour acheminer le paquet vers sa destination est calculée. Supposons que la ligne ayant la plus faible estimation soit AB car la ligne AC est saturée. Dans ce cas, le paquet est envoyé vers le routeur B qui l'enverra à son tour vers le routeur C. Le routeur A recevra donc un paquet de renforcement qui lui permettra de mettre à jour sa Q-table. Supposons maintenant que la ligne AC ne soit plus saturée et qu'un paquet en A doit aller vers C. La Q-valeur correspondant à la ligne AC n'est pas mise à jour, puisque aucun paquet n'a été envoyé sur cette ligne. Par conséquent, le routeur A enverra le paquet vers le routeur B, augmentant ainsi le trafic sur la ligne AB. En fait, il continuera à envoyer les paquets sur cette ligne tant que le temps d'acheminement ne dépasse pas le temps d'acheminement associé à la ligne AC devenu obsolète. Ainsi, en cas de montée en charge du réseau, le routeur optimise sa politique de routage afin d'utiliser au mieux les lignes libres du réseau. Cependant, en cas de libération de place dans la file d'attente, le temps de convergence vers une politique de routage optimal est plus élevé.

Pour remédier aux inconvénients énumérés ci-dessus, nous proposons un nouvel algorithme basé sur un apprentissage par renforcement que nous appèlerons Q-Neural Routing. Contrairement aux algorithmes basés sur l'algorithme Q-Routing, l'estimation et la mise à jour de la fonction Q sont effectuées à l'aide d'un réseau de neurones. Compte tenu de ses capacités d'approximation et de prédiction, un réseau de neurones permet d'une part, la réduction de l'espace mémoire occupé par la table de routage et d'autre part, l'intégration de l'état des files d'attente des voisins pour anticiper les problèmes de congestion des routeurs.

De plus, pour résoudre le problème d'exploration, une solution intéressante consisterait à introduire un mécanisme permettant l'exploration régulière de tous les chemins. Plusieurs solutions existent telles que l'exploration par génération de bruit, l'exploration probabiliste, l'exploration par inondation [THR92].

- **L'exploration par génération de bruit :** Cette méthode est employée principalement quand l'implémentation du Q-Learning est faite par un réseau de neurones. Cependant, on peut l'adapter à notre problème en ajoutant un bruit aléatoire gaussien dans la mise à jour du signal de renforcement. Ceci permet de générer une partie aléatoire dans l'estimation de la fonction de renforcement, et ainsi d'explorer une fois de temps en temps et de manière aléatoire un chemin non optimal.

- **L'exploration par échange d'informations :** Ce type d'exploration est utilisé par exemple dans RIP où les routeurs échangent leurs tables de routage par période de 30 secondes. Dans notre cas, l'échange régulier des Q-tables entre routeurs voisins permet de déterminer à chaque fois le chemin optimal. Cependant, les informations retournées par les routeurs voisins ne reflètent pas forcement l'état du réseau surtout si le changement se produit en un point éloigné du réseau. Ce problème peut être résolu en répercutant tous les changements d'état et ce en inondant le réseau par ce type de paquet d'exploration. Une telle technique peut créer une surcharge du réseau même si on fait passer ces paquets sur des canaux dédiés, séparés des canaux de communication.

- **L'exploration probabiliste** : elle consiste à attribuer une probabilité à chaque route menant à une destination. Ainsi, le plus court délai aura une probabilité P_{max}, et l'ensemble des autres chemins, une probabilité (1 - P_{max}). Un nombre tiré au hasard déterminera alors quel chemin utiliser. De cette manière, les paquets atteignent leur destination avec un délai proche de l'optimum, tout en assurant une bonne exploration des chemins non optimaux.

Le choix de la valeur de P_{max} résulte des considérations suivantes:

- Si celle-ci est trop élevée, tous les paquets emprunteront le chemin présentant le plus court délai estimé, et par conséquent n'exploreront pas les solutions

alternatives. Le risque est alors que le routeur ne puisse être informé d'une baisse de trafic sur un de ces chemins, et qu'il continue à emprunter un chemin qui n'est plus optimal.

- Si P_{max} est trop faible, on notera une baisse de performances en termes de temps d'acheminement des paquets. En effet, un nombre relativement élevé de paquets de données seront destinés à l'exploration et feront augmenter le temps de transit moyen.

IV.3 Solution proposée : L'algorithme Q-Neural Routing

Cet algorithme est basé sur un réseau de neurones permettant une modélisation fiable de la dynamique du réseau. Cette méthode permet un apprentissage discriminant et une meilleure prise en compte du contexte dans lequel évolue le réseau, pour l'estimation des temps d'acheminement des paquets. L'utilisation du réseau de neurones pour la mise à jour de la fonction Q a pour objectif d'une part, de prendre en compte en plus du temps de bout en bout, d'autres paramètres tels que l'état des files d'attentes ou la nature des flux et d'autre part, de minimiser l'espace mémoire occupé par la Q-table. Ainsi, l'algorithme Q-Routing nécessite un espace mémoire proportionnel au produit du nombre d'adresses de destination par le nombre de voisins, alors que le Q-Neural Routing ne nécessite qu'un espace mémoire proportionnel à la taille du réseau de neurones et cela quel que soit le nombre de destinations. Dans ce mémoire, nous proposons un algorithme prenant en compte le temps de bout en bout et l'état des files d'attentes qui constituent deux paramètres critiques de la QoS. Ceci n'enlève rien à la généricité et la validité de l'approche.

IV.3.1 Formulation mathématique du routage avec apprentissage par renforcement

Soient :

- X : L'ensemble des états du réseau (vus par un routeur) et $x_t \in X$ l'état d'un routeur à l'instant t.

- A_x : L'ensemble des choix de routes possibles (choix d'un voisin y pour arriver à la destination d) en se plaçant dans l'état x_t.

- $A = \bigcup_{x \in X} A_x$: L'ensemble de tous les choix de routes possibles.

- $(y_t, d) \in A_x$: Le choix de la route à l'instant t pendant l'état x_t.

- R : L'ensemble des récompenses que le routeur peut recevoir de la part de l'environnement (ses voisins), tel que :

$$r(x_t, y_t, d) = q_y + \delta + Q(y_t, \tilde{z}, d) \tag{IV.1}$$

où $r(x_t, y_t, d)$ est la récompense retournée par le voisin y en appliquant le choix (y_t, d). q_y et δ représentent respectivement le temps d'attente dans le routeur y et le délai de transmission entre les nœuds x et y. $Q(y, \tilde{z}, d)$ représente le temps moyen optimal pour acheminer un paquet vers sa destination d, à partir du routeur y via le routeur $\tilde{z}$.

- π : La politique de routage adoptée. Il s'agit du choix (y_t, d) du routeur x : $(y_t, d) = \pi(x_t)$ telle que :

$$\begin{aligned} \pi \quad &: \quad X \to A \\ &\quad\; x_t \mapsto (y_t, d) \end{aligned} \tag{IV.2}$$

- Π : L'ensemble des politiques

- U : L'ensemble des couples état-choix de route.

- Q^π : La fonction de valeur du couple état-choix de route, telle que :

$$\begin{aligned} Q^\pi : &\ (U, \Pi) \to R \\ &\ (x_0, (y_0, d), \pi) \mapsto Q^\pi(x_0, (y_0, d)) \end{aligned} \tag{IV.3}$$

La politique optimale est celle qui conduit à la meilleure qualité (Q^*), définie par :

$$\pi^* = \min_{a \in A_x} Q^\pi(x, y, a) \tag{IV.4}$$

La politique optimale est celle qui minimise le critère Q^{π} . L'algorithme d'apprentissage par renforcement a pour objectif de minimiser de manière itérative la politique π.

En utilisant l'algorithme du Q -Learning, l'estimation de la fonction Q-valeur s'écrit:

$$Q(x,y,d) = Q(x,y,d) + \alpha\left[r(x,y,d) + \gamma \min_{(z,d)\in A_y} Q(y,z,d) - Q(x,y,a) \right] \qquad \text{(IV.5)}$$

IV.3.2 Mise à jour des poids du réseau de neurones

Dans le chapitre II, nous avons vu que l'algorithme Q-Routing sauvegarde le signal de renforcement reçu des voisins dans une Q-table. Dans l'algorithme Q-Neural Routing, nous utilisons ce signal de renforcement pour mettre à jour les poids du réseau de neurones, utilisé pour l'estimation des Q-valeurs.

Toutes les Q-valeurs reçues sont susceptibles de varier, sauf dans le cas où le routeur est directement connecté à une destination. Dans ce cas, et seulement dans ce cas, la Q-valeur est fixe et égale à zéro. Il s'agit de la seule information sûre dont nous disposons. Par conséquent, il est nécessaire de commencer l'exploration à partir de ce point précis. L'objectif est donc de faire remonter l'information depuis son point d'arrivée, puisque c'est ce dernier qui dispose du temps d'acheminement réel. Ceci a pour conséquence d'éviter tout d'abord des "discussions" inutiles entre routeurs qui ne s'échangeraient que des valeurs estimées et pas forcément actualisées. L'information doit remonter de l'arrivée jusqu'aux sources. Ceci est confirmé si on démarre l'algorithme Q-Neural Routing sans aucune méthode d'exploration, avec une valeur initiale identique pour toutes les fonctions de renforcement. On voit alors nettement les paquets transiter d'un routeur à un autre de manière aléatoire, puis à force d'exploration un paquet arrive à trouver sa destination. Une fois réceptionné par la destination, le routeur met à jour son réseau de neurones à partir d'une valeur de renforcement fiable. Ce qui signifie qu'à chaque fois qu'un paquet arrive sur ce routeur, il détermine la route optimale. Ce routeur envoi à son tour au routeur précédent une valeur de renforcement fiable, ainsi de suite.

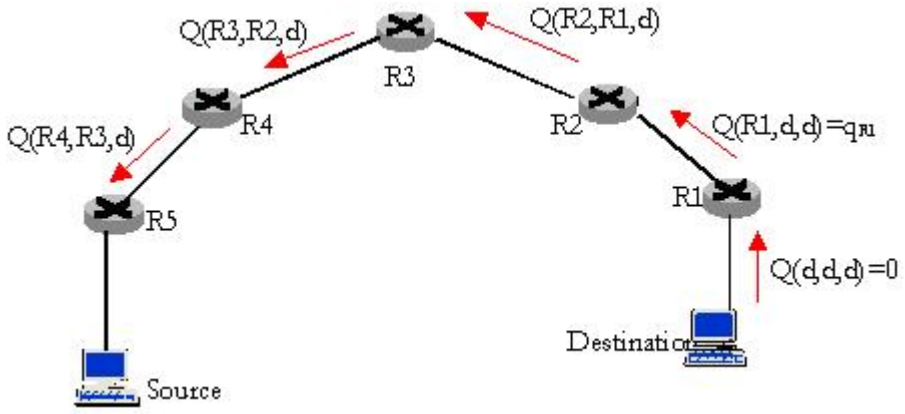


Figure IV.3 : Exploration à l'initialisation de l'algorithme de routage

La technique d'exploration que nous proposons s'appuie donc sur la technique de l'exploration avancée (forward exploration) utilisé dans le Q-Routing à laquelle nous avons adjoint un mécanisme inspiré du protocole RSVP qui consiste à remonter à partir des nœuds destinations, le temps d'acheminement de bout en bout. Ainsi, périodiquement, chaque routeur connecté à un sous-réseau envoi à tous ses voisins un paquet contenant l'adresse du sous-réseau sur lequel il est connecté ainsi que la valeur de renforcement représentant l'état de sa file d'attente ; la valeur de sa fonction de renforcement étant nulle (Figure IV.3). Chaque routeur recevant ce paquet doit alors calculer le temps minimum correspondant à la valeur de la fonction de renforcement pour atteindre le sous-réseau qui a émis l'information, et en informe tous ses voisins (Figure IV.4).

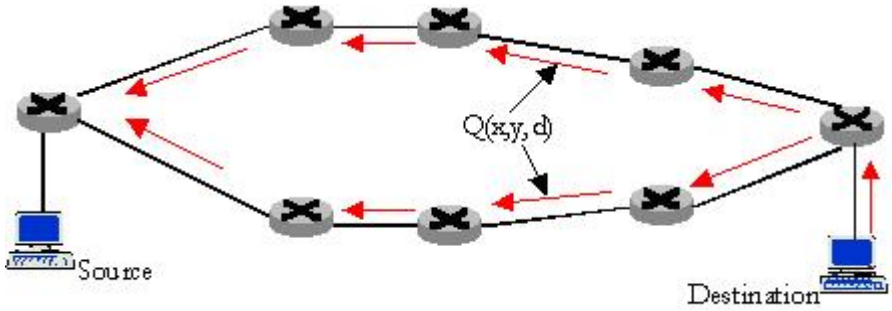


Figure IV.4 : Exploration du réseau par inondation

Dans le choix de la périodicité de mise à jour des poids du réseau de neurones, les considérations suivantes doivent être prises en compte:

- Si cette période est trop courte, le trafic lié à cette mise à jour devient plus important, ce qui conduit à surcharger un peu plus le réseau.

- Si cette période est trop grande, les Q-valeurs estimées deviennent obsolètes dès lors que l'état du réseau varie sensiblement entre deux instants de mise à jour.

Cependant, ce processus conduit à ce que ces paquets de renforcement restent tout le temps présents dans le réseau transportant ainsi une information obsolète et saturant le réseau. Donc, il est possible qu'un paquet fasse un tour et soit utilisé pour la mise à jour avec une mauvaise valeur. Pour résoudre ce problème, il est possible d'associer à chaque paquet une durée de vie TTL (Time To Live), dont la valeur dépend essentiellement de la taille du réseau ou d'utiliser une autre technique qui consiste à numéroter les paquets de renforcement et à reprendre le principe d'inondation utilisé dans OSPF.

La fréquence de mise à jour des poids du réseau de neurones, peut également être réalisée non plus selon une certaine fréquence mais plutôt en utilisant un compteur, le routeur d'arrivée n'envoyant une requête de mise à jour qu'après réception de N paquets. Une autre solution consiste à combiner les deux solutions précédentes. Ainsi, un timer est utilisé avec une grande période quand le routeur d'arrivée ne reçoit aucun paquet, et dans le cas contraire, il s'agit d'envoyer une requête après réception de N paquets sachant que comme le routeur est actif, le besoin de connaître les meilleurs chemins est important.

IV.3.2.1 Formulation mathématique

Après avoir vu dans le paragraphe précédant le principe de mise à jour des Q-Valeurs, nous présentons ici sa formulation mathématique. Le modèle neuronal de l'estimateur des Q-Valeurs comprend une couche d'entrée, une couche cachée et une couche de sortie. Soient:

- c : le vecteur d'entrée du réseau de neurones
- h : le vecteur représentant la réponse de la couche cachée
- o : le vecteur représentant la réponse des cellules de la couche de sortie. Il représente les Q-valeurs estimées des voisins.
- t : le vecteur représentant la réponse désirée des cellules de la couche de sortie. Il représente les Q-valeurs réelles des voisins.

- *W* : la matrice représentant les poids des connexions reliant les cellules de la couche d'entrée aux cellules de la couche cachée.

- *Z* : la matrice représentant les poids des connexions reliant les cellules de la couche cachée aux cellules de la couche de sortie.

Lorsqu'un routeur *y* transmet un paquet de renforcement (signal renforcement) à un routeur voisin *x,* ce paquet contient l'estimation optimale $Q(y,\tilde{z},d)$ du temps restant pour acheminer les paquets à la destination *d*. Cette estimation est calculée comme suit :

$$Q(y,\tilde{z},d) = \min_{z \in \text{voisins de } y} \{q_z + \delta + Q(y,z,d)\} \quad \text{(IV.6)}$$

Cette estimation est alors utilisée par le routeur *x* comme valeur de sortie désirée de son réseau de neurones. La mise à jour des poids synaptiques est alors effectuée en utilisant la méthode de rétro propagation du gradient. Cette méthode peut être décrite comme suit :

A l'itération *t*+1, la mise à jour de la matrice de connexions *Z*, s'écrit :

$$Z_{[t+1]} = Z_{[t]} + \alpha\delta_{cachée}h^T \quad \text{(IV.7)}$$

où α est un nombre réel positif définissant le pas d'apprentissage et $\delta_{cachée}$ le signal d'erreur des cellules de la couche cachée qui est définie comme suit :

$$\delta_{cachée} = o \otimes (I - o) \otimes e \quad \text{(IV.8)}$$

où $\otimes$ représente le produit terme à terme des vecteurs et *I* un vecteur unité. e qui représente le signal d'erreur des cellules de la couche de sortie s'écrit:

$$e = (t - o) \quad \text{(IV.9)}$$

L'apprentissage pour la couche cachée se déroule ensuite de façon similaire à celui de la couche de sortie. La mise à jour de la matrice de connexions W s'écrit:

$$W_{[t+1]} = W_{[t]} + \alpha\delta_{entrée}c^T \quad \text{(IV.10)}$$

où $\delta_{entrée}$ représente le signal d'erreur des cellules de la couche d'entrée. Ce signal d'erreur ne peut-être évalué par comparaison avec une valeur idéale. Il est estimé comme une fonction du signal d'erreur provenant de la couche de sortie et de l'activation des cellules de la couche cachée. Précisément, le vecteur d'erreur pour les cellules de la couche cachée s'obtient ainsi :

$$\delta_{entrée} = h \otimes (1-h) \otimes (Z_{[t]}^{T} \delta_{cachée}) \qquad \text{(IV.11)}$$

Comme on peut le constater, le signal d'erreur s'obtient en propageant l'erreur de la couche de sortie à travers la couche cachée.

IV.4 Mise en œuvre du Q-Neural Routing

Pour la mise en œuvre de l'algorithme proposé, nous avons utilisé des routeurs à quatre interfaces d'entrée/sortie, c'est à dire que chaque routeur ne peut avoir plus de quatre voisins. Ceci n'enlève rien à la validité de l'algorithme proposé étant donné que l'on s'intéresse ici essentiellement aux performance de l'algorithme de routage dans des conditions de trafic variées qui peuvent être établies quel que soit le nombre d'interfaces du routeur.

IV.4.1 Réseau de neurones utilisé

Le réseau de neurones utilisé est un perceptron multicouches (MLP)[DAV90] avec 5 entrées et 4 sorties (Figure IV.5). Les 5 entrées correspondent à l'état des files d'attentes des voisins et à l'adresse de la destination *d* à atteindre. Les 4 sorties représentent le temps estimé pour acheminer le paquet jusqu'à sa destination en passant par l'un des quatre voisins.

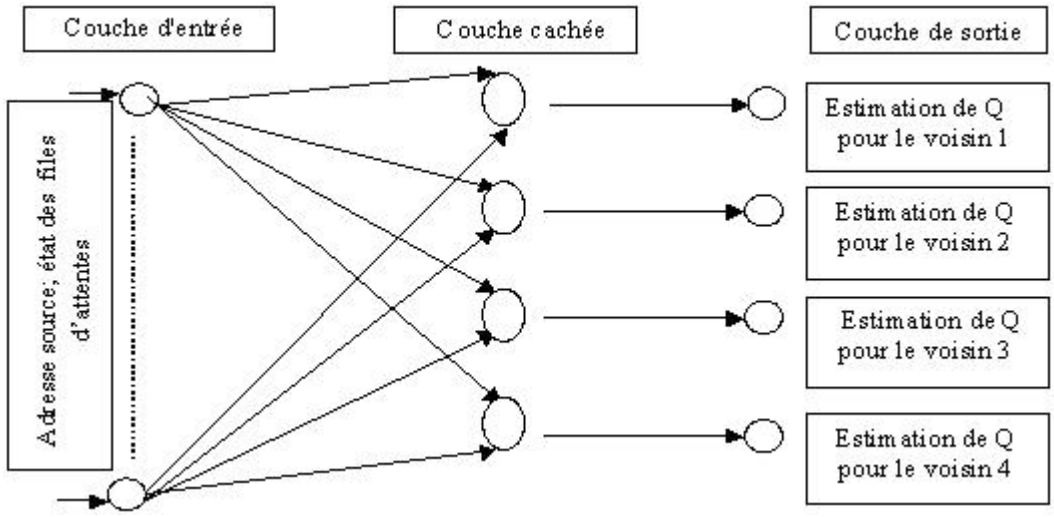


Figure IV.5 : Réseau de neurone estimateur

L'algorithme général du Q-Neural Routing implémenté peut être résumé comme suit:

```
Début
Tant que (paquet non reçu)
    Faire
    |
    Fait

Si (paquet= "paquet de renforcement") alors
|   Faire
|   |   Mettre à jour les poids du réseau de neurones (méthode de rétro propagation)
|   |   Détruire le paquet de renforcement
|   Fait
|
|  Sinon
|   Faire
|   |   Calculer  la Q-Valeur optimale en utilisant le modèle neuronal.
|   |   Envoyer  le paquet au routeur correspondant à cette dernière.
|   |   Envoyer un paquet de renforcement contenant la Q-valeur optimale
|   Fait
|
Fin de Si

Goto Début
```

IV.4.2 Dimensionnement du réseau de neurones

La topologie du réseau de neurones doit être en générale fixée avant l'apprentissage. Les seules variables pouvant être modifiées sont les valeurs des poids des connexions. La spécification de cette architecture, du nombre de cellules de chaque couche et des connexions demeure un problème crucial. Si ce nombre est insuffisant, le modèle ne pourra pas prendre en compte l'ensemble des données. A contrario, s'il est trop important, l'apprentissage sera parfait mais la reconnaissance sera médiocre. Ce problème est connu sous le nom de "sur-apprentissage" ou "overfitting". Le modèle apprend par coeur les données de l'apprentissage. Des progrès ont été faits dans ce sens, en appliquant par exemple une méthodologie bayésienne à la sélection du modèle [MCK91], des contraintes sur les poids [NOW91] [GAL99], ou on se sert souvent du nombre d'exemples disponibles dans une base d'apprentissage. Or, il s'agit ici d'un apprentissage en ligne, pour lequel le nombre d'exemples n'est pas défini a priori. Pour cela, nous proposons une étude empirique pour trouver un compromis entre une estimation satisfaisante de la fonction Q et un temps de calcul acceptable.

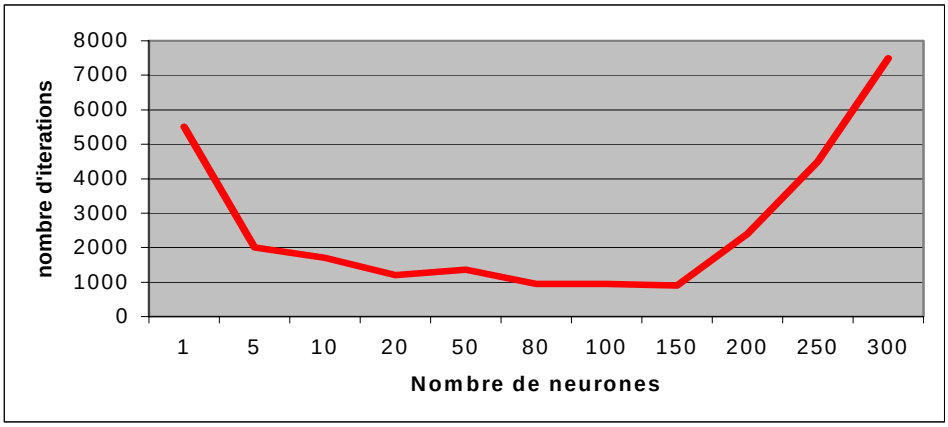


Figure IV.6 : Influence du nombre de neurones de la couche cachée sur le temps de convergence

Les résultats illustrés Figure IV.6 montrent que 150 neurones pour la couche cachée donnent les meilleurs résultats.

IV.5 Outil de simulation

La validation et l'évaluation des approches algorithmiques que nous proposons nécessitent leurs implémentations sur un simulateur. Une partie relativement importante de

ces travaux de thèse a été consacrée à cette tâche, notamment à la prise en main de l'outil de simulation OPNET et à la méthodologie de création, d'implémentation et de validation de modèles de routage. Dans ce paragraphe, nous passerons en revue quelques outils de simulation couramment utilisés. Nous nous focaliserons ensuite plus particulièrement sur l'outil OPNET en développant l'implémentation de l'algorithme Q-Routing sur lequel s'appuient nos deux approches algorithmiques.

IV.5.1 Simulateurs de réseau

Il existe de nombreux simulateurs de réseau permettant l'implémentation et l'évaluation des performances de protocoles. Certains d'entre eux sont des prototypes issus de la recherche universitaire et d'autres des produits commercialisés. Dans ce qui suit, nous présentons les simulateurs NS, MaRS et OPNET.

IV.5.1.1 NS

NS (pour *Network Simulator* http://www.isi.edu/nsnam/ns/) est distribué par l'université de Berkeley (USA). Il s'agit d'un simulateur à évènements discrets conçu pour la recherche dans le domaine des réseaux. Il fournit un appui substantiel pour les simulations utilisant TCP, ainsi que les protocoles de routage. Le simulateur est écrit en langage C++ et utilise le langage OTcl comme interface de commande et de configuration.

NS a l'avantage de proposer de nombreuses extensions pour améliorer ses caractéristiques de base. L'utilisateur spécifie la topologie du réseau en inscrivant la liste des nœuds et des liens dans un fichier de topologie. Des générateurs de trafic peuvent être attachés à n'importe quel nœud pour décrire son comportement (par exemple, un serveur FTP ou un client Telnet).

Il existe différentes manières pour collecter les informations produites pendant une simulation. Les données peuvent être affichées directement pendant la simulation ou, être enregistrées dans un fichier pour un traitement et une analyse, ultérieurs.

Les deux modes de surveillance sous NS sont d'une part, l'enregistrement de chaque paquet lorsqu'il est émis, reçu ou rejeté et d'autre part, la surveillance de paramètres bien précis, telles que la date d'arrivée des paquets, leur taille, etc. Les données produites pendant

la simulation sont enregistrées dans des fichiers et peuvent ensuite être traitées. L'animateur de Réseau (*Network Animator*) est un outil permettant d'analyser ces fichiers. Son interface graphique permet d'afficher la topologie configurée et de visualiser les échanges de paquets.

Le principal avantage de NS est sa gratuité (son code source est disponible sur Internet).

NS est largement utilisé pour les recherches dans le domaine des réseaux, et reconnu comme un outil permettant d'expérimenter de nouvelles idées et de nouveaux protocoles.

IV.5.1.2 MaRS

MaRS (*Maryland Routing Simulator* www.cs.umd.edu/projects/netcalliper/software.html), de l'université du Maryland, est un simulateur à évènements discrets qui fournit une plate-forme flexible pour l'évaluation et la comparaison d'algorithmes de routage. Il est implémenté en langage C sous UNIX, avec deux interfaces graphiques : Xlib et Motif.

MaRS permet à l'utilisateur de définir une configuration réseau constituée d'un réseau physique, d'algorithmes de routage et de générateurs de trafic. L'utilisateur peut contrôler sa simulation, enregistrer les valeurs de certains paramètres, et sauvegarder, charger et modifier des configurations réseau.

MaRS est structuré en deux parties : un moteur de simulation, qui contrôle la liste des événements et l'interface utilisateur ; et une collection de composants pour modéliser la configuration réseau et manipuler certaines fonctions de la simulation.

La configuration de la simulation (réseau, algorithmes, liaisons, trafic) peut se faire via des fichiers de configuration ou via une interface graphique X-Windows.

IV.5.1.3 OPNET

Le projet OPNET (*Optimum Network Performance* http://www.mil3.com/) a été lancé en 1987. Actuellement, ce logiciel est commercialisé par MIL3 et constitue le premier outil commercial de simulation de réseaux de communication.

C'est un outil de simulation à base d'événements discrets, écrit en langage C. Il possède une interface graphique utilisée dans divers modes, qui facilite le développement de nouveaux modèles et programmes de simulation.

De nombreux modèles de protocoles sont fournis avec la distribution standard d'OPNET, ce qui constitue un avantage certain par rapport aux autres simulateurs, qui ne proposent bien souvent qu'une implémentation de TCP/IP et de quelques protocoles classiques de routage.

OPNET est un outil très souple, qui permet de simuler un réseau à n'importe quel niveau de granularité. Il est utilisé de manière intensive dans de nombreux projets de développement.

IV.5.2 Logiciel de simulation utilisé : OPNET

Pour la validation des performances de l'algorithme proposé, nous avons opté pour le simulateur OPNET pour plusieurs raisons. Il permet de visualiser la topologie physique d'un réseau local, métropolitain, distant ou embarqué. Le langage de spécification de protocoles s'appuie sur la description formelle à l'aide d'automates à états finis. Il permet aussi d'appréhender plus facilement les problèmes de délais de transit et de topologie du réseau étudié.

OPNET dispose de trois niveaux hiérarchiques imbriqués : le *network domain*, le *node domain* et le *process domain.*

IV.5.2.1 Network Domain

C'est le niveau le plus élevé de la hiérarchie d'OPNET. Il permet de définir la topologie du réseau en y déployant des hôtes, des liens ainsi que des équipements actifs tels que des switchs ou des routeurs, (Figure IV.7).

Chaque entité de communication (appelée nœud), définie par son modèle, est entièrement configurable. L'interface graphique associée au network domain tient compte du déploiement spatial du réseau (distance entre un routeur situé à Paris et un autre situé à Boston) .

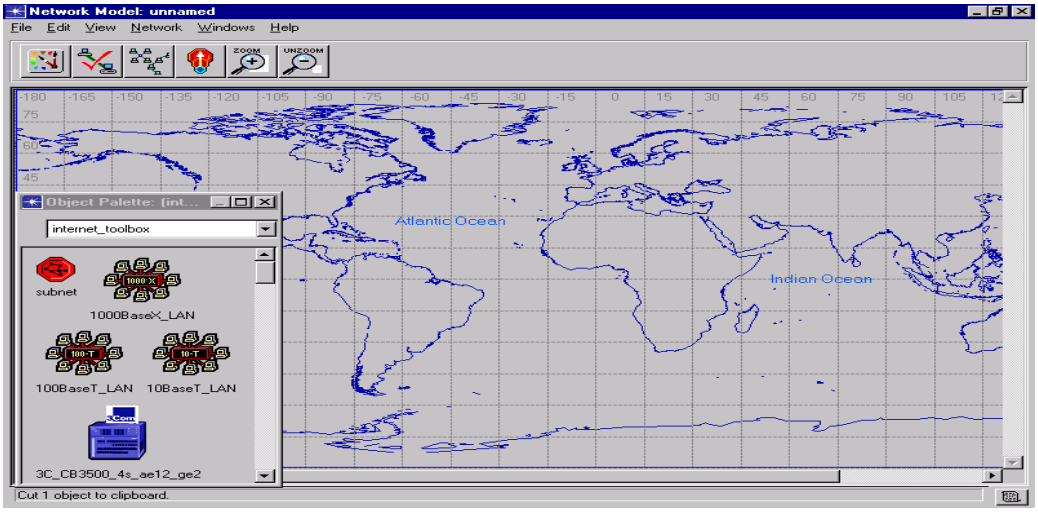


Figure IV.7 : Fenêtre du Network Editor

IV.5.2.2 Node Domain

Le *Node domain* permet de définir la constitution des nœuds (routeurs, stations de travail, hub, etc). Le modèle est défini à l'aide de blocs appelés modules (Figure IV.8).

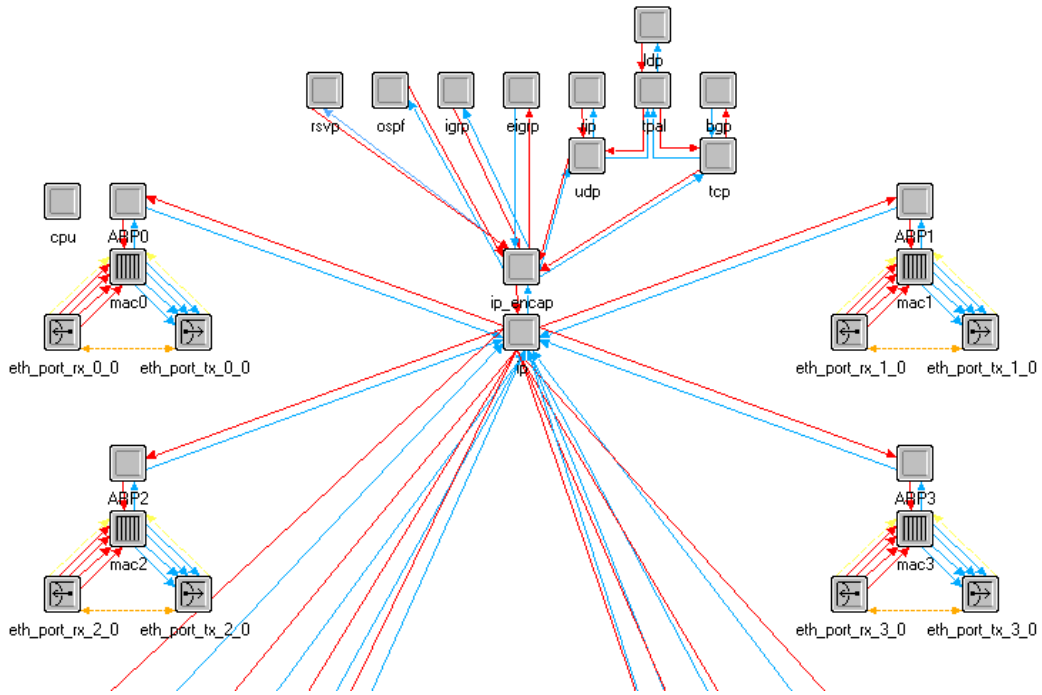


Figure IV.8: Le Node Domain sous OPNET

Certains modules sont non programmables : il s'agit principalement des émetteurs (*transmitters)* et des récepteurs (*receivers)*, dont la seule fonction est de s'interfacer entre le nœud et les liens auxquels il est connecté. Par contre les autres modules sont entièrement programmables, il s'agit des processus (*processors)* et des fils d'attente (*queues).*

- Les ***processors*** sont des modules qui remplissent une tâche bien précise du nœud

7. Ils symbolisent en fait les différentes parties du système d'exploitation d'une machine, et plus principalement les différentes couches réseau implémentées dans le nœud (Ethernet, IP…).

8. Ils peuvent communiquer entre eux via des flux de paquets (*packets streams*), qui permettent de faire transiter un paquet d'une couche à une autre à l'intérieur d'une même machine.

Cette organisation permet d'avoir une vision claire de la pile de protocoles implémentée dans un nœud, et de connaître rapidement leurs interactions.

Par exemple, le module IP est relié, via des *streams*, aux modules de la couche 4 tels que TCP, UDP, et à ceux de la couche 2 (Ethernet).

Les *statistic wires* constituent le second type de liens permettant une communication entre modules. Ils permettent de remonter les statistiques d'un module à un autre, comme par exemple la taille et le délai des files d'attente (*queues)* des *transmitters.*

IV.5.2.3 Process Domain

C'est à ce niveau que l'on définit le rôle de chaque module programmable.

Un module possède par défaut un processus principal, auquel peuvent s'ajouter des processus fils accomplissant une sous-tâche précise. OPNET fournit des mécanismes permettant à tous les processus créés à l'intérieur d'un *process domain* (Figure IV.9) de communiquer entre eux, via un bloc de mémoire partagée ou, l'ordonnancement d'interruptions logicielles.

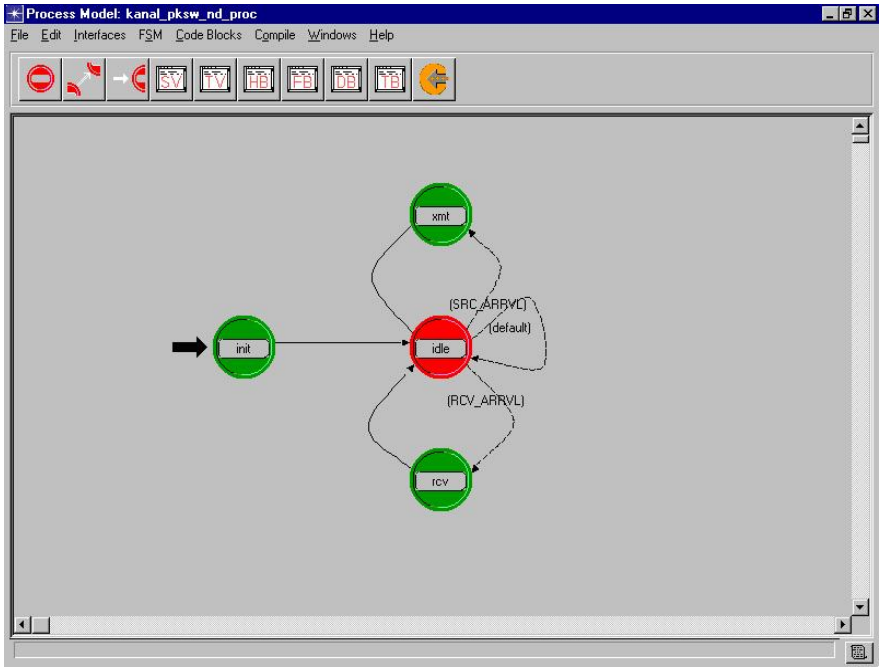


Figure IV.9: Fenêtre du Process Editor

Le rôle d'un module est déterminé par son *process model*, que l'on décrit sous forme d'une machine à états finis (*finite state machine*).

Chaque bloc représente un état différent, dans lequel la machine exécute un code déterminé.

Les transitions sont symbolisées par des liens entre blocs et déterminées par des conditions (interruptions, variable ayant une certaine valeur, etc.).

Les actions à effectuer sont écrites en langage C, et OPNET fournit une bibliothèque de plus de 400 fonctions propriétaires spécifiques à l'usage des réseaux (création, envoi et réception de paquets, extraction de valeurs contenues dans les différents champs d'une entête, etc.).

IV.5.2.4 Autres paramètres

OPNET permet de gérer deux autres types d'objets relatifs aux réseaux : les liens et les formats de paquets. Des dizaines de modèles de liens sont fournis (par exemple un modèle de lien Ethernet 100Mpbs, un modèle de lien ATM, etc.) mais il est tout à fait possible de

créer son propre modèle pour lequel il est nécessaire de définir des paramètres comme par exemple, le débit du lien, le type de paquet supporté, le type d'erreurs générées, etc.

Le dernier aspect important dans la construction d'un modèle est le type de trame circulant sur le réseau. Là encore, OPNET fournit en standard le modèle de la plupart des entêtes connues, utilisables immédiatement dans le *Process Domain*

Cependant, pour la mise en œuvre d'un nouveau protocole, la création d'un nouveau format de paquet est nécessaire. On peut alors utiliser l'éditeur de paquets, et définir la taille, la position et le type de données contenues dans chaque champ.

IV.5.2.5 Simulation sous OPNET

OPNET fournit en standard une liste conséquente d'implémentations d'équipements réseau: routeurs, stations de travail, switchs. On peut donc construire une simulation de réseaux en utilisant principalement deux méthodes :

1. En utilisant les nœuds pré-programmés fournis par la librairie de OPNET.
2. En définissant ses propres modèles d'équipements

Dans la mesure où OPNET fournit les codes source des nœuds proposés dans la librairie, la seconde méthode paraît plus aisée. Cependant, elle est bien évidemment plus complexe que la première, et nécessite de bonnes connaissances en matière de programmation et de réseaux. Néanmoins elle est indispensable dans le cas où l'on désire valider un nouvel algorithme.

IV.5.3 Implémentation des algorithmes

IV.5.3.1 Implémentation dans le Node Domain

Tous les routeurs sont basés sur un même modèle de nœud, comportant quatre entrées et quatre sorties (Figure IV.10). Les quatre entrées sont notées *rcv* et les quatre sorties *xmt*. Les émetteurs et les récepteurs sont en réalité connectés deux à deux (rcv_i avec xmt_i) pour former une seule et même liaison bidirectionnelle un seul port physique). La Figure V.4 illustre le modèle utilisé pour les routeurs.

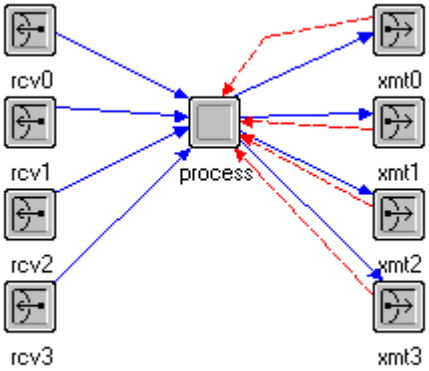


Figure IV.10: Le Node model du routeur

Les flèches bleues représentent les *packet streams*, c'est à dire les canaux par lesquels les modules peuvent échanger des paquets au sein d'un même nœud.

Les paquets sont reçus par les entrées *rcv* et directement envoyés au *process*. Ce dernier lance l'algorithme de routage et décide sur quelle sortie *xmt* renvoyer le paquet. Les temps d'attente des queues des émetteurs sont communiqués au *process* via des *statistic wires* (représentés par les flèches rouges en pointillés).

Il faut noter l'absence de modules implémentant des queues de paquets entre les récepteurs et le processeur, et entre le processeur et les émetteurs car ces derniers sont intégrés dans les *rcv* et les *xmt*.

Par conséquent, ce modèle de nœud est un modèle parfait dans lequel les queues ont une taille infinie : leur taille n'étant pas limitée, elle peut accueillir sans exception tous les paquets reçus et émis.

Dans la réalité, les queues ont une taille fixe et relativement petite pour économiser la mémoire. Par conséquent, si une queue se remplit trop vite (à cause d'une émission ou d'une réception trop brusque), les paquets suivants ne peuvent pas être ajoutés et sont simplement détruits. Ils sont alors définitivement perdus.

IV.5.3.2 Format de paquet

Pour simplifier le traitement des paquets et ne traiter que les paramètres utilisés dans les algorithmes de routage mis en œuvre, un nouveau format de paquet (Figure IV.11) a été créé.

type
source
destination
coût
renforcement (bits 0-31) …
… renforcement (bits 32-63)

Figure IV.11: Format de paquet utilisé

- Le champ *type* permet de définir le type de paquet : données, informations de routage, renforcement.

- Les champs *source* et *destinataire* indiquent respectivement les adresses logiques de l'émetteur et du destinataire final. Toujours dans un soucis de simplicité, les adresses sont des entiers positifs assignés de manière arbitraire : il n'y a pas, comme pour le protocole IP, de notion de sous-réseau et de masque. Il faut aussi remarquer que ces adresses sont des adresses logiques appartenant aux deux hôtes situés aux extrémités de la communication. Dans la mesure où les liaisons entre nœuds sont comparables à des liaisons point à point, il n'est pas nécessaire de mettre en place un mécanisme d'adressage physique (comme le fait Ethernet avec les adresses MAC) : sur une liaison point à point, une machine ne peut avoir qu'un seul interlocuteur direct.

- Enfin, le champ *renforcement* est un champ de 64 bits destiné à contenir la valeur du signal de renforcement.

IV.5.4 Implémentation dans le process domain: application au Q-Routing

La Figure IV.12 montre le processus du routeur sous la forme d'une machine à états finis.

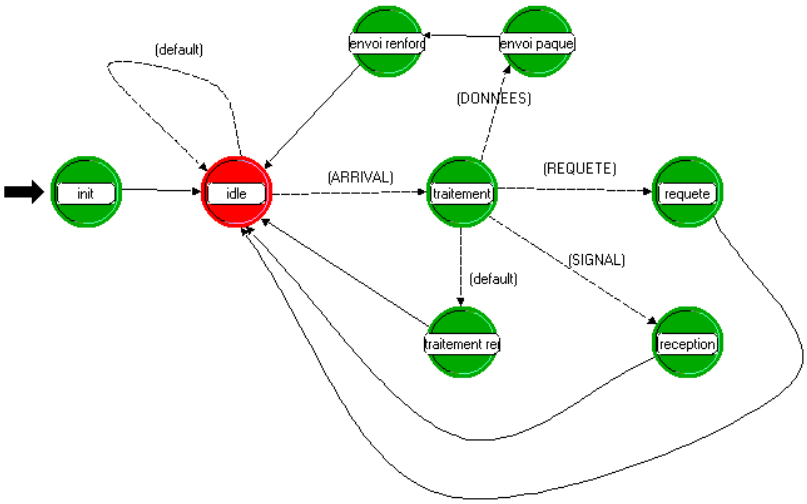


Figure IV.12: Le Process Model du Q-Routing

Après initialisation, le routeur attend l'arrivée de paquets. A l'arrivée d'un paquet (indiquée par l'évènement ARRIVAL), le routeur détermine son type en lisant la valeur du champ *type* (initialement, le type du paquet était contenu dans le champs *TOS* de l'entête IPv6).

Un traitement différent est effectué en fonction du type de paquet :

- Si le paquet est un paquet de données (DONNEES), le routeur trouve l'interface indiquant le délai le plus court, place le paquet dans la file d'attente et envoie un paquet de renforcement.

- Le paquet est une requête (REQUETE) : il s'agit d'une mise à jour de la fonction de renforcement des routeurs pour une destination particulière.

- Le paquet est un signal (SIGNAL) : c'est un paquet spécial circulant au moment de l'initialisation et permettant au routeur de connaître les sous réseaux auxquels il est directement connecté. La fonction de renforcement pour ce sous réseau est généralement égale à zéro.

- Si le champ *type* ne correspond à aucun des types précédents, il s'agit d'un paquet de renforcement : le routeur procède alors à la mise à jour de la fonction de renforcement pour la destination spécifiée par le paquet.

La table de routage comporte une entrée pour chaque destination du réseau. Chaque entrée contient les délais pour chaque interface du routeur.

L'automate à états finis (process model) du Q-Routing comporte 5 états principaux :

INITIALISATION : cette phase permet principalement de détecter les interfaces actives.

DONNEES : lorsque le routeur reçoit un paquet de données, il active l'algorithme de routage.

- Il extrait du paquet l'adresse du destinataire, et sauvegarde l'index de l'interface sur laquelle le paquet a été reçu, pour envoyer plus tard un paquet de renforcement.
- Il récupère les temps d'attente de chaque file.
- Il interroge ensuite sa table de routage, en sélectionnant la ligne correspondant au destinataire, puis détermine la ligne présentant le plus court délai, c'est à dire l'interface dont la valeur de T est minimale.
- Il place le paquet dans la file d'attente déterminée.
- Il envoie un paquet de renforcement sur l'interface de réception, avec T comme valeur de renforcement.

REQUETE : pour connaître le délai total pour une destination à partir de toutes les interfaces, une exploration est nécessaire. Un paquet de requête est envoyé régulièrement par les routeurs connectés directement à un sous réseau. Ce paquet ressemble à un paquet de renforcement, avec un signal égal à zéro. Les routeurs voisins reçoivent ce paquet, et si ce dernier contient un meilleur temps d'acheminement, ils mettent à jour leur table. Une recherche du plus court délai est alors effectuée, prenant en compte le temps passé dans les files d'attente. Le paquet est construit avec cette nouvelle valeur, puis envoyé sur toutes les interfaces autres que l'interface de réception.

SIGNAL : le routeur se trouve généralement dans cet état lors de la phase d'initialisation. Le paquet indique l'adresse d'un réseau auquel le routeur est directement connecté. Une

nouvelle entrée est créée dans la table de routage, avec un délai égal à 0 pour l'interface de réception.

RENFORCEMENT : le routeur extrait la valeur du signal de renforcement T, ainsi que l'adresse de destination d pour laquelle la fonction de renforcement doit être mise à jour.

La fonction de renforcement pour l'interface de réception i est alors modifiée de la manière suivante :

$$Q(i,d) = Q(i,d)(1 - \eta) + \eta(T + s)$$

s étant le temps de transmission pour atteindre le routeur suivant.

IV.6 Simulation

L'objectif de ces simulations est de comparer les performances de l'algorithme Q-Neural routing à celles obtenues avec l'algorithme Q-Routing en termes de délai d'acheminement moyen des paquets. Pour cela, quatre scénarios ont été étudiés. Dans les trois premiers, nous analysons l'algorithme proposé dans trois niveaux de charge du réseau (trafic faible, trafic fort, pics de trafic). Dans le quatrième scénario, il s'agit d'étudier la capacité d'adaptation de l'algorithme Q-Neural routing face à un changement de topologie du réseau. Pour parvenir à une analyse comparative non biaisée des performances de l'algorithme proposé par rapport à celle du Q-Routing, nous avons adjoint à ce dernier, un mécanisme d'exploration par inondation.

IV.6.1 Réseau utilisé

Pour les simulations, nous avons utilisé deux réseaux irréguliers : comportant respectivement 32 nœuds (Figure IV.13) et 28 nœuds (Figure IV.14).

Dans le réseau illustré Figure IV.13, il existe deux chemins possibles pour acheminer les paquets entre la partie gauche et la partie droite du réseau: l'itinéraire comprenant les routeurs 21 et 22 (R 1) et l'itinéraire comprenant les routeurs 29 et 30 (R 2). Dans le réseau illustré Figure IV.14, on distingue quatre itinéraires possibles : passant respectivement par les liens R1, R2, R3, et R4.

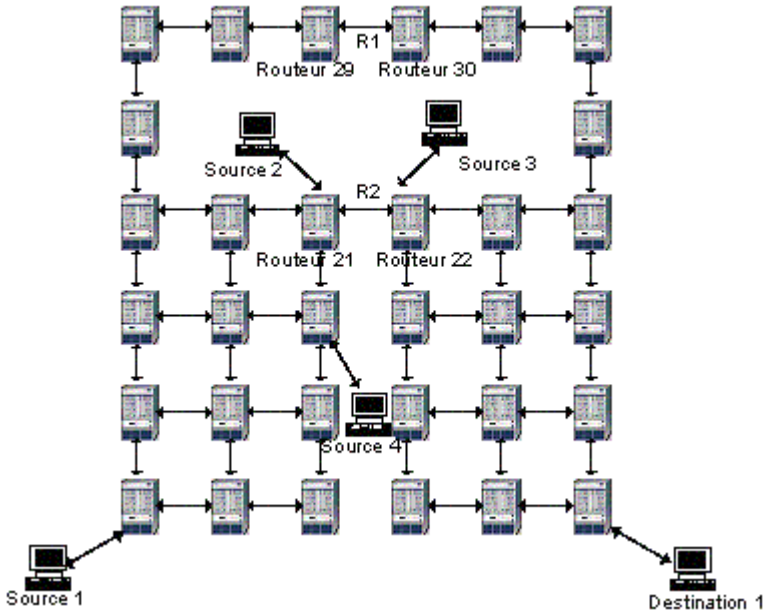


Figure IV.13 : Réseau de 32 nœuds à topologie irrégulière utilisée pour la simulation.

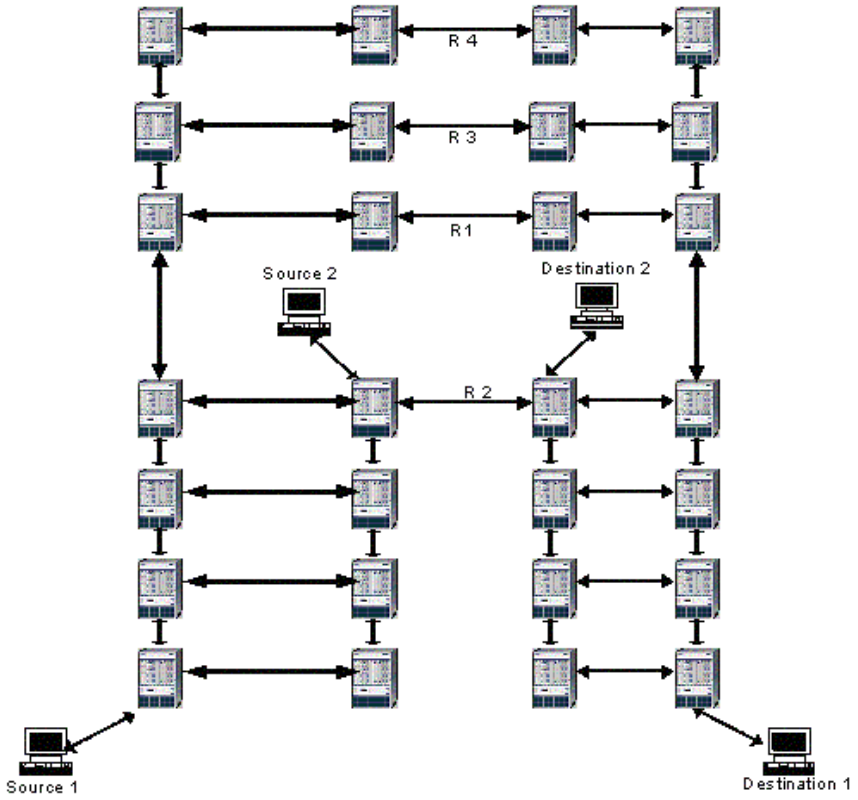


Figure IV.14 : Réseau de 28 nœuds à topologie irrégulière utilisée pour la simulation

IV.6.2 Conditions de simulation

Dans les paragraphes précédents, nous avons vu que les algorithmes Q-Routing et Q-Neural Routing sont basés sur plusieurs paramètres tels que le pas d'apprentissage ou le nombre de neurones. La valeur de ces paramètres, pour les simulations réalisées, sont résumés dans le tableau suivant :

Algorithme	Pas d'apprentissage	Nombre de neurones de la couche cachée	Temps entre 2 explorations
Q-Routing	0.85	---------	50s
Q-Neural Routing	0.85	150	50s

Les évaluations des performances ont porté sur l'évolution du temps d'acheminement (*end-to-end delay*) des paquets échangés entre deux hôtes : le nœud "source 1" émet régulièrement des paquets à destination du nœud "destination 1".

Dans ces simulations, nous avons étudié trois conditions de trafic:

- **Trafic faible** où seul le nœud « source 1 » génère du trafic à destination du nœud "destination 1". Dans ce cas, les paquets empruntent le chemin incluant le lien R2. L'objectif de ce scénario est d'étudier le comportement de l'algorithme Q-Neural Routing en cas de non congestion de la route optimale.

- **Trafic élevé** où en plus du trafic entre les nœuds "source 1" et "destination 1", un trafic soutenu est ajouté afin de saturer la route R2, provoquant ainsi la congestion des routeurs faisant partie du chemin optimal.

- **Plusieurs Pics de trafic** où un trafic élevé entre les nœuds "source 2" et "destination 2", est généré à des instants aléatoires suivis d'un arrêt complet de l'échange. Cette opération est réalisée plusieurs fois au cours de la simulation. Il s'agit ici d'étudier la robustesse de l'approche proposée et la convergence de l'algorithme Q-Neural Routing.

IV.6.3 Résultats de simulation

IV.6.3.1 Q-Routing amélioré

Les résultats illustrés *Figure IV.15* sont ceux obtenus sur le réseau à 28 nœuds (Figure IV.14) où le nœud "source 1" envoi régulièrement des paquets au nœud "destination 1". Une heure après le début de simulation et pendant 30 minutes, le nœud "source 2" envoi des paquets au nœud "destination 1" pour saturer le chemin optimal incluant le lien R2.

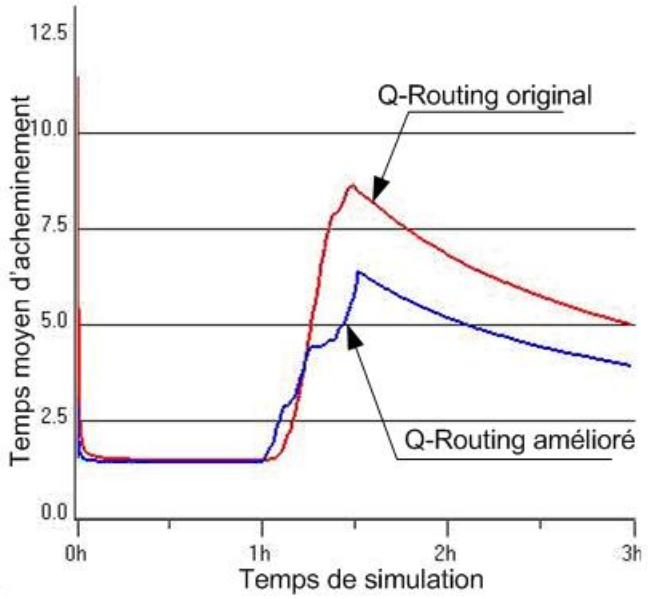


Figure IV.15 : Comparaison du temps moyen d'acheminement du Q-Routing original et du Q-Routing amélioré

La Figure IV.15 montre clairement que lorsque la charge du réseau est faible, les temps moyens d'acheminement des paquets sont approximativement les mêmes. Ce résultat est justifié puisque les deux algorithmes choisissent le même chemin optimal, passant par le lien R2. Tant que ce dernier n'est pas congestionné, il reste toujours optimal. Par contre, dès que la charge du réseau augmente ce qui conduit à la saturation du chemin optimal, le Q-Routing "amélioré" améliore le temps moyen d'acheminement comparé au Q-Routing original. En effet, on constate par exemple qu'au bout d'une heure et demi de simulation, le Q-Routing "amélioré" réduit le temps moyen d'acheminement de prés de 38% par rapport au Q-Routing original. Ce résultat est justifié par la bonne connaissance de l'état réel du réseau puisque, l'exploration par inondation permet une mise à jour régulière des Q-Valeurs des routeurs appartenant aux chemins non utilisés.

IV.6.3.2 Trafic faible

La Figure IV.16 et la Figure IV.17 illustrent le temps moyen d'acheminement (end-to-end delay) mesuré par le nœud "destination 1", des paquets envoyés par le nœud "source 1".

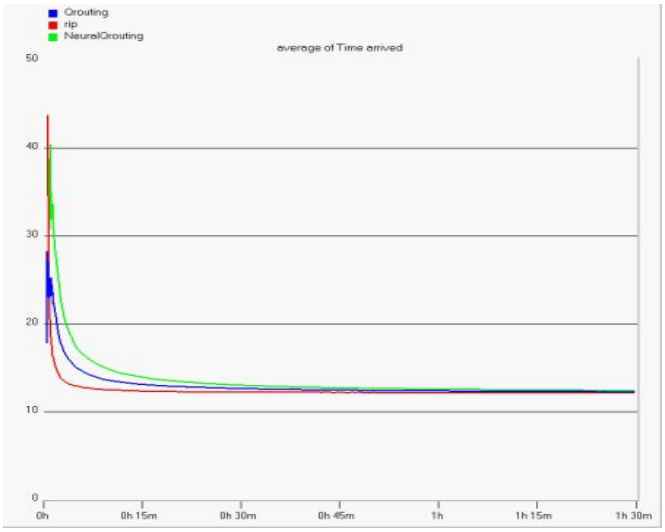


Figure IV.16 : Temps moyen d'acheminement pour un trafic faible sur le réseau à 32 nœuds

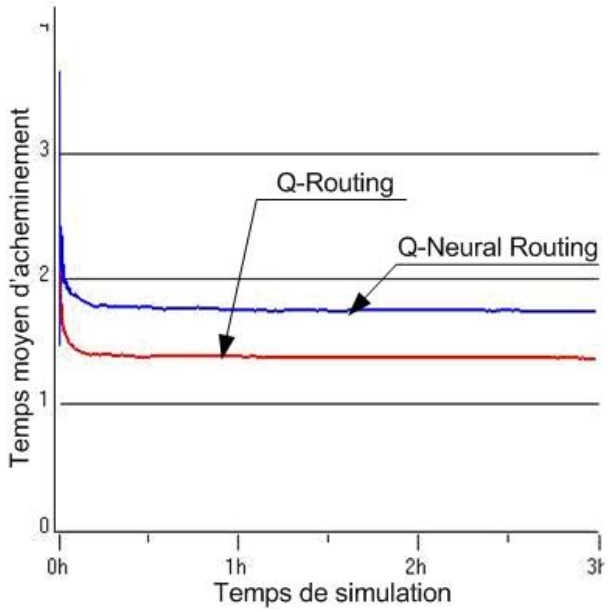


Figure IV.17 : Temps moyen d'acheminement pour un trafic faible sur le réseau à 28 nœuds

L'interprétation des figures ci-dessus met en avant deux aspects :

- Les temps moyens d'acheminement des paquets pour l'algorithme Q-Neural Routing ne se stabilisent qu'au bout de 20 minutes de simulation, ce qui démontre

que le temps de convergence de cet algorithme est supérieur aux deux autres. En effet, l'initialisation des paramètres du réseau de neurones au début de la simulation est aléatoire puisqu'on ne dispose d'aucune information sur l'état du réseau. Par conséquent, pour mettre à jour les poids du réseau de neurones, un temps de calcul supplémentaire est nécessaire. Ce n'est qu'après un certain temps que les paramètres du réseau de neurones se stabilisent.

- Les temps moyens d'acheminement sont plus favorables à l'algorithme basé sur la Q-table que celui utilisant le modèle connexionniste (sur l'expérience illustrée Figure IV.17, le temps d'acheminement total est augmenté de 10%). En effet, il apparaît clairement qu'en cas de faible charge, la variation des paramètres du réseau de communication issue du changement de trafic est négligeable. Une méthode classique de routage suffit largement à intégrer cette variation dans un routeur. L'utilisation d'un réseau de neurones est surdimensionné par rapport à la complexité du problème. En effet, tous les calculs dus à la mise à jour des poids synaptiques, et nécessitant un certain temps, ne sont pas adaptés à la nature du problème posé.

IV.6.3.3 Trafic élevé

Dans ce scénario, les résultats illustrés Figure IV.18 ont été obtenus sur le réseau à 32 nœuds (Figure IV.13) où les nœuds "source 2", "source 3" et "source 4" envoient des paquets au nœud "destination 1" durant 10 minutes et cela 5 minutes après le début de simulation. Les résultats de la Figure IV.19 correspondent à ceux obtenus sur le réseau à 28 nœuds (Figure IV.14) où, une heure après le début de simulation et durant 30 minutes le nœud "source 2", envoi des paquets au nœud "destination 1".

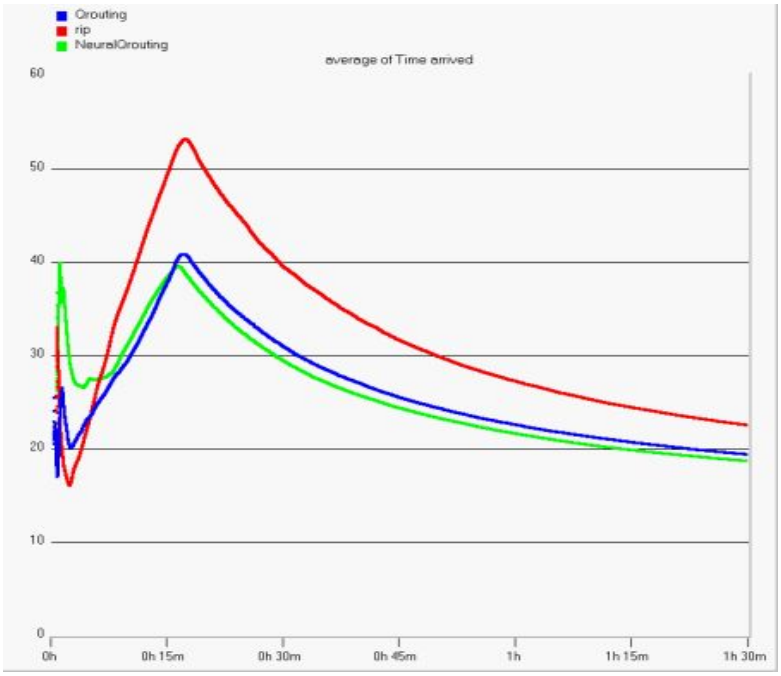


Figure IV.18 : Temps moyen d'acheminement pour un trafic élevé sur le réseau à 32 nœuds

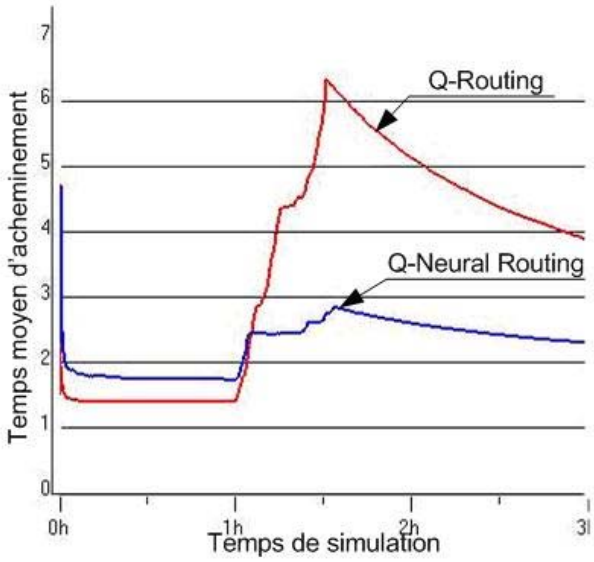


Figure IV.19 : Temps moyen d'acheminement pour un trafic élevé sur le réseau à 28 nœuds

Dans le cas où la charge du réseau est élevée, les résultats obtenus (Figure IV.18) montrent clairement, qu'après une période d'apprentissage, les algorithmes Q-Routing et Q-Neural routing, permettent de minimiser sensiblement le temps d'acheminement des paquets comparativement à l'algorithme RIP. Ainsi, au bout de 15 minutes de simulation, ces deux algorithmes réduisent ce temps respectivement de 23,6% et de 27,3%. En effet, lorsque la charge du réseau augmente sur les routeurs 21 et 22, la taille des files d'attente augmente très

vite. Or ce facteur n'est pas pris en compte dans la politique de routage utilisé dans l'algorithme RIP contrairement aux politiques Q-Routing et Q-Neural Routing. Celles-ci établissent de nouvelles routes à chaque fois qu'une route en cours d'utilisation s'avère encombrée.

Par ailleurs, dans le cas où le trafic sur le réseau devient très important et durable, on constate que le Q-Neural Routing réduit substantiellement le délai moyen d'acheminement. Par exemple, au bout d'une heure et demi de simulation, le Q-Neural routing réduit ce temps de 52% par rapport au Q-Routing. Une telle amélioration est due d'une part, à l'intégration de l'état des files d'attentes pour anticiper la congestion des routeurs et d'autre part, à la qualité d'apprentissage du réseau de neurones.

IV.6.3.4 Plusieurs Pics de trafic élevé

Ces expériences ont pour objectif d'étudier la robustesse des algorithmes Q-Neural Routing et Q-Routing face à différents pics de trafic successifs, se produisant à des instants aléatoires nous permettant de juger de la robustesse des approches proposées. Les résultats de la Figure IV.20 résultent de l'injection de deux pics de trafic séparés d'une durée d'une heure : le premier pic se produit trente minutes après le début de la simulation et s'arrête trente minutes plus tard, le second pic de même durée se produit trente minutes après.

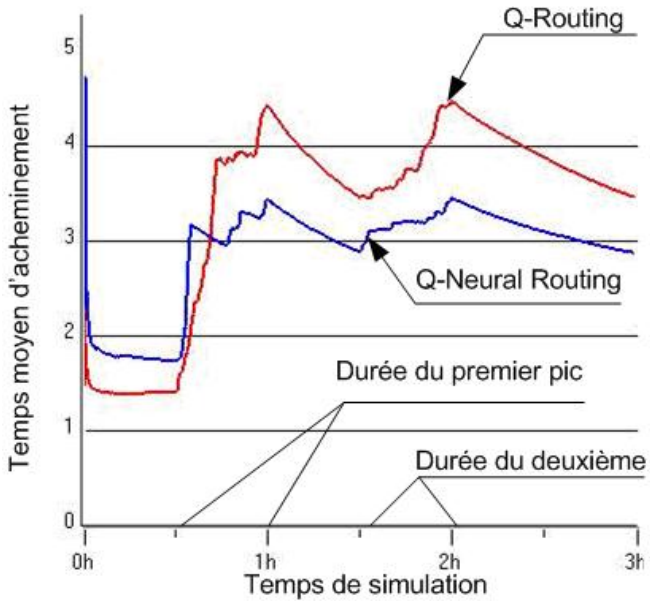


Figure IV.20 : Temps moyen avec deux pics de trafic successifs sur un réseau à 28 nœuds

Les résultats de ce scénario viennent valider ceux obtenus dans le paragraphe précédent confirmant ainsi le comportement déjà observé de l'algorithme proposé. Ce dernier garantit des temps moyens d'acheminement inférieurs de prés de 30% par rapport à l'algorithme Q-Routing.

IV.6.3.5 Synthèse des résultats d'évaluation

Le tableau 1 illustre le temps moyen global d'acheminement des paquets envoyés de la "source 1" vers la "destination 1". Dès que la charge du réseau augmente fortement et provoque la congestion du chemin optimal en cours d'utilisation, l'algorithme proposé améliore le temps d'acheminement total moyen de l'algorithme Q-Routing de près de 38% dans le cas d'une forte charge et de 30% dans le cas de deux pics successifs de trafic. Cependant, dans le cas d'une faible charge du réseau, l'algorithme Q-Neural Routing augmente ce temps de prés de 10% par rapport à l'algorithme Q-Routing.

	Trafic faible	Trafic élevé	Deux pics de trafic
Q-Routing	1.32 s	3.47 s	4.08 s
Q-Neural Routing	1.48 s	2.14 s	2.86 s

Tableau 1: Moyenne globale des temps d'acheminement

IV.6.3.6 Tolérance aux pannes

Plusieurs expériences ont été effectuées dans ce but. Elles consistent à introduire des ruptures de lignes changeant ainsi la topologie du réseau, l'objectif étant d'étudier la capacité d'adaptation de l'algorithme Q-Neural Routing face à un changement de topologie du réseau par rapport à l'algorithme Q-Routing. La Figure IV.21 illustre les résultats obtenus à partir d'une expérience effectuée sur le réseau de la Figure IV.14 où le nœud "source 1" envoie régulièrement des paquets à la "destination 1". Au bout de dix minutes, une rupture simultanée des liens R1 et R2 est simulée.

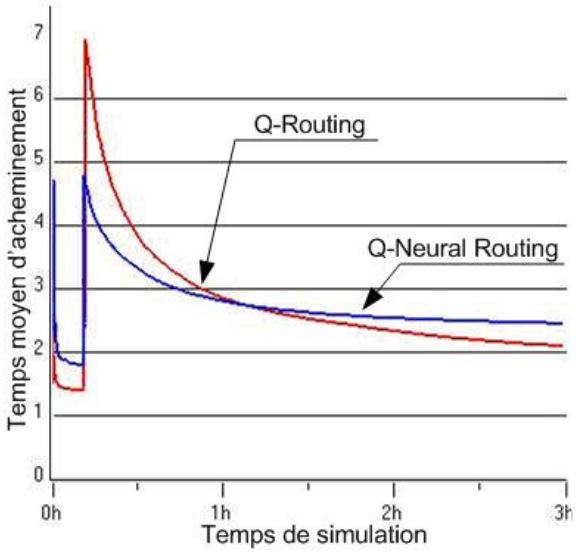


Figure IV.21 : Temps moyen d'acheminement dans le cas de changement de topologie

Comme on pouvait s'y attendre, les temps moyens d'acheminement des paquets ont augmenté par rapport à ceux constatés dans une simulation à topologie constante. Il n'en demeure pas moins que les conclusions auxquelles nous sommes arrivés concernant le comportement de l'algorithme proposé restent toujours valables. En effet, sur l'ensemble des expériences effectuées (Cf. Figure IV.21), lors d'un changement de topologie (dans notre cas, rupture de ligne), introduit une forte augmentation du délai d'acheminement du Q-Routing, contrairement à l'algorithme Q-Neural Routing. Ce dernier, après un court instant d'instabilité, retrouve assez rapidement le chemin optimal. La prise en compte des informations issues du contexte dans le modèle neuronal rend l'algorithme Q-Neural Routing plus réactif aux changements dans le réseau.

IV.7 Conclusion

Dans ce chapitre, nous avons présenté l'algorithme Q-Neural Routing qui constitue une amélioration de l'algorithme Q-Routing. L'algorithme proposé est basé sur un apprentissage par renforcement où l'estimation et la mise à jour des Q-Valeurs sont effectuées à l'aide d'un réseau de neurones. L'utilisation de ce dernier permet d'une part, l'intégration de l'état des files d'attente des voisins pour anticiper d'éventuelles congestions des routeurs et d'autre part, l'utilisation d'un espace mémoire représentant la Q-table, indépendant du nombre de destinations. Le mécanisme d'exploration utilisé pour la mise à jour des Q-Valeurs s'appuie sur la technique de l'exploration avancée (forward exploration) à laquelle nous avons adjoint

un mécanisme d'exploration par inondation. Ceci permet une mise à jour régulière des Q-Valeurs des routeurs appartenant aux chemins non utilisés.

Les simulations réalisées sur différentes topologies de réseau à trafic irrégulier, montrent que l'algorithme proposé est bien adapté pour des réseaux à fortes charges ou des réseaux soumis à des pics de trafic. En effet, dans ces conditions, comparativement à l'algorithme Q-Routing, l'amélioration des temps moyens d'acheminement des paquets atteint 38%. De plus, les expériences menées sur des réseaux à topologie changeante montrent clairement la capacité d'adaptation du Q-Neural Routing comparativement au Q-Routing.

Chapitre V

L'algorithme K-Shortest path Q-Routing

V.1 Introduction

Dans ce chapitre, nous présentons le deuxième algorithme de routage adaptatif appelé K-Shortest path Q-Routing. Ce dernier est basé sur la technique du routage multi-chemin combiné avec l'algorithme Q-Routing. Pour ce faire, nous nous appuyons sur l'algorithme de Dijkstra généralisé pour la recherche des K plus courts chemins minimisant le nombre de sauts. A partir d'un mécanisme d'exploration hybride, nous formulons l'algorithme de mise à jour des paramètres de routage. Les performances de l'algorithme proposé sont ensuite évaluées et comparées à celles des algorithmes RIP, Q-Routing et K-Shortest path Routing.

V.2 L'algorithme K-Shortest path Q-Routing

En plus de la réduction de l'espace mémoire occupé par la table de routage, nous nous intéressons ici à la minimisation du temps de convergence de la politique de routage. Pour ce faire, nous proposons une approche algorithmique basée sur la technique du routage multi-chemin combiné avec l'algorithme Q-Routing où l'espace d'exploration est réduit aux k meilleurs chemins au sens d'un critère qui peut être lié par exemple à la bande passante, au nombre de sauts ou au taux de perte. Dans ce travail, nous nous sommes focalisés sur la recherche des chemins minimisant le nombre de sauts. Ceci n'enlève rien à la validité de l'approche proposée. Pour ce faire, nous proposons un algorithme basé sur l'algorithme de Dijkstra généralisé auquel on adjoint un mécanisme de suppression de boucles. Le chemin optimal parmi les k meilleurs est celui dont le temps d'acheminement de bout en bout est le plus court. L'algorithme de mise à jour des paramètres de routage repose sur une méthode hybride associant le principe de l'exploration avancée à chaque fois qu'un paquet de données est échangé entre routeurs, et celui de l'exploration probabiliste permettant d'explorer les (k-1) autres chemins sans surcharger le réseau.

V.2.1 Algorithme de recherche des k plus courts chemins

La résolution du problème de recherche des k plus courts chemins, fait appel à la théorie des graphes où de nombreux algorithmes ont été proposés [HSU94] [EPP99] [MAR98] [APO98]. Ces algorithmes peuvent être classés en deux types : Ceux qui déterminent une liste de chemins parmi lesquels figurent les k plus courts et ceux qui sont basés sur le principe d'optimalité. Ce principe stipule qu'il existe un plus court chemin constitué des plus courts

chemins élémentaires [MAR98]. La formulation de l'algorithme K-Shortest path Q-Routing proposé dans ce chapitre est basée sur ce principe.

V.2.1.1 Formulation de l'Algorithme

Pour résoudre le problème de recherche des k plus courts chemins, nous proposons une approche algorithmique basée sur un algorithme à étiquetage (labeling algorithms) [MAR98].

Les algorithmes à étiquetage ont en commun, l'utilisation d'une ou plusieurs étiquettes (*labels*) assignées à chaque nœud. Pour la recherche du plus court chemin, une seule étiquette est assignée à un nœud, tandis que dans la recherche des k plus courts chemins, il peut y avoir une ou plusieurs étiquettes qui peuvent être assignées à un seul nœud.

Une étiquette permet de reconstituer les chemins dont fait partie le nœud. Pour un nœud quelconque i, elle contient les informations suivantes:

9. La distance du chemin allant d'un nœud source s jusqu'au nœud intermédiaire i.

10. Le nœud précédant le nœud i sur ce chemin.

Ces informations suffisent pour parcourir le réseau et reconstituer les k plus courts chemins en partant de la destination et en remontant jusqu'à la source. Comme il peut y avoir plusieurs étiquettes associées à un même nœud, chacune comportant des informations différentes, il est nécessaire de définir une fonction, notée h, permettant de retrouver le nœud auquel est associée l'étiquette.

Selon la technique utilisée pour le choix de l'étiquette à explorer, on distingue deux classes d'algorithme à étiquetage généralisé:

- **Les algorithmes à correction d'étiquettes** (label correcting algorithms) : dans cette catégorie, l'étiquette à explorer est choisie arbitrairement (habituellement la première découverte). De plus, les étiquettes sont rendues permanentes simultanément à la fin de l'exploration du chemin. Un des algorithmes les plus connus, reposant sur ce principe, est celui de Bellman-Ford généralisé [MAR98].

- **Les algorithmes à création d'étiquettes** (*label setting algorithms*) : dans cette catégorie, l'étiquette dont la distance est la plus petite est choisie comme étiquette à explorer. Il est à noter que l'étiquette d'un nœud est rendue permanente à chaque itération. Cependant, pour utiliser ce type d'algorithmes, le réseau ne doit comporter aucun arc ayant un coût négatif.

Notre approche est basée sur l'algorithme de Dijkstra généralisé, reposant sur le principe de la création d'étiquettes [MAR98]. Ce choix est justifié étant donné que la recherche des K plus courts chemins est basée sur la minimisation du nombre de sauts, où le coût d'un arc est égal à un. De plus, le principe d'optimalité retenu stipule qu'un plus court chemin est constitué de plus courts chemins élémentaires. Par conséquent, la recherche du plus court chemin consiste à explorer à chaque itération le nœud ayant la plus petite distance.

L'algorithme de Dijkstra généralisé comprend trois étapes:

- **Etape d'initialisation :** A l'initialisation de l'algorithme, l'ensemble des étiquettes éligibles X ne contient que l'étiquette du nœud source. La distance cumulée de cette étiquette ainsi que le nombre de chemins P sont nuls.

- **Etape 1 :** Dans cette étape, on cherche dans l'ensemble X l'étiquette qui présente la plus courte distance cumulée. Celle-ci est alors retirée de X. Si le nœud associé à l'étiquette sélectionnée correspond au nœud destination alors ce nœud est ajouté au chemin sinon on passe à l'étape 2.

- **Etape 2 :** Dans cette étape, les arcs partant du nœud correspondant à l'étiquette sélectionnée dans l'étape 2 sont explorés. Les étiquettes des nœuds situés à l'autre extrémité sont ajoutées à l'ensemble des étiquettes éligibles X et la distance cumulée correspondant à chaque étiquette est alors calculée.

Remarque

La recherche d'une étiquette dépend du critère utilisé. Par exemple, si ce dernier concerne la recherche de la meilleure bande passante sur un chemin, l'étiquette sélectionnée correspond alors à celle ayant la plus grande distance cumulée. Par contre si, le critère utilisé concerne la minimisation du nombre de sauts alors l'étiquette sélectionnée est celle dont la

distance cumulée est la plus courte. Dans l'étape 2, pour pouvoir remonter tout le chemin jusqu'à la source, il est nécessaire de sauvegarder le nœud précédant chaque nœud étiqueté.

```
* N – Liste des noeuds du réseau
 * X – Liste des nœuds "éligibles"
 * Count_i – Nombre de chemins déjà calculés entre s et i
 * elm – Nombre d'étiquettes assignées
 * P – Liste des chemins de s à t
 * K – Nombre de chemins à calculer
/* Initialisation */
count_i  = 0 pour tout i appartenant à N
elem = 1    ( Il n'y a qu'une seule étiquette pour l'instant : celle associée au nœud source (s))
  (Associe l'étiquette et le nœud)
h(elem) = s
h-1(s) = {elem}
distance_elem = 0
X = {elem}
P^K = 0     /* Aucun chemin n'a encore été calculé */
While (count_t < K and X != 0) /* Tant qu'on n'a pas calculé les K chemins et qu'il existe encore des candidats */
    begin
/* ETAPE 1 */
          (Trouver l'étiquette éligible présentant la plus courte distance)
          k = element de X tel que distance_k <= distance_x pour tout x de X
          X = X – {k}                                          (Retirer l'étiquette de la liste des candidats)
          i = h(k)                                             (Trouver le nœud associé à l'étiquette)
          count_i = count_i + 1                                (Incrémenter le compteur de chemins passant par ce nœud)
          if (i == t)alors                                     (Si on est arrive à destination)
              begin                                            (Ajouter le nouveau chemin)
                  p = chemin de 1 à k
                  P^K = P^K U {h(p)}
              end
/*  ETAPE 2*/

        if (count_i <= K) alors                                (S'il n'y a pas trop de chemins passant par ce nœud)
          begin
                for each arc(i,j) ∈ i                          (Pour chaque lien appartenant à ce nœud)
                 begin
                    elem = elem + 1    (Ajouter le noeud situé à l'autre extremité de l'arc, dans la liste des
                                        candidats)
                    distance_elem = distance_k + c_ij    (Sauvegarde des informations de la nouvelle etiquette)
                    precedent_elem = k
                    h(elem) = j                  (Pour retrouver le nœud auquel appartient l'étiquette)
                    h-1(j) = h-1(j) U {elem}
                    X = X U {elem}                             (Ajouter l'étiquette à la liste des candidats)
                 end
          end
    end
```

Figure V.1 : Algorithme de Dijkstra généralisé

Cet algorithme peut conduire à un chemin optimal comportant une ou plusieurs boucles, c'est à dire passant plusieurs fois par le même nœud. En effet, lorsque la liste des arcs associés à un nœud est parcourue, il n'est pas vérifié que les nœuds suivants ne figurent pas déjà dans un chemin en cours d'exploration. Dans le cas du routage, un chemin comportant des boucles ne fait que retarder l'acheminent des paquets et doit par conséquent être éliminé. Pour ce faire, nous introduisons dans cet algorithme un mécanisme de suppression de boucles que nous développerons dans le paragraphe V.2.1.2.

Exemple d'application

Pour illustrer le fonctionnement de cet algorithme, considérons un réseau composé de cinq nœuds numérotés de 1 à 5 (Figure V.2). Le nœud 5 représente la source à partir de laquelle nous calculons les trois plus courts chemins vers le nœud 1. Dans cet exemple, les liaisons étant unidirectionnelles, la formation de boucles est ainsi évitée.

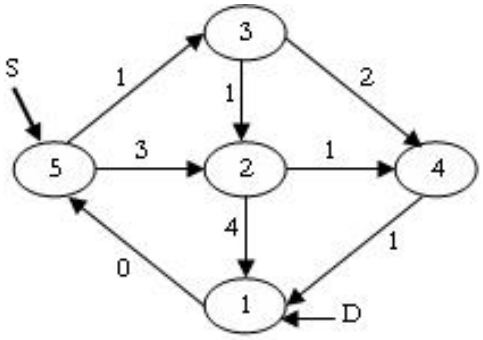
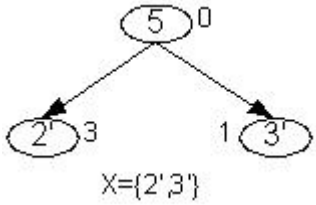


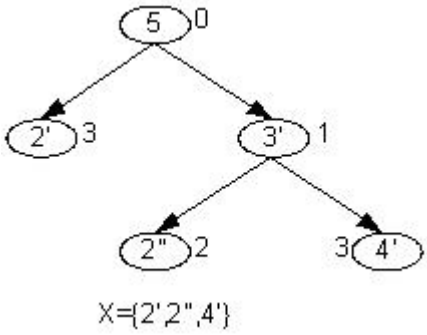
Figure V.2 : Réseau utilisé pour illustrer le calcul des k plus courts chemins

Étape 1: A l'initialisation, l'ensemble X des étiquettes candidates ne contient que l'étiquette associée au nœud source.

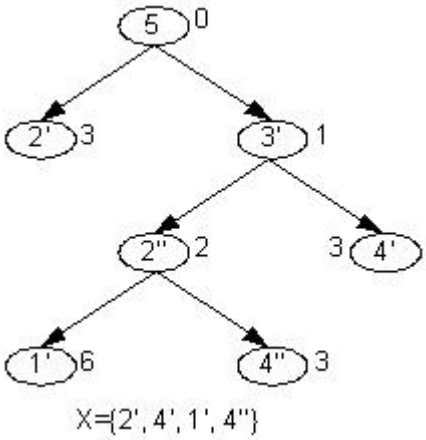
Étape 2: Tous les arcs allant du nœud 5 sont parcourus, et les nœuds situés à l'autre extrémité sont ajoutés à X. Le nœud 1 n'est pas ajouté puisque la liaison du nœud 1 vers le nœud 5 est unidirectionnelle.



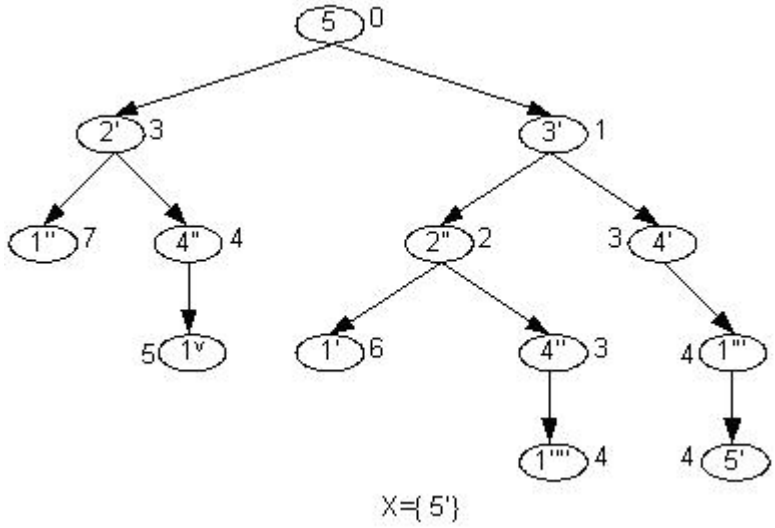
Étape 3: On cherche dans X l'étiquette dont la distance cumulée est la plus courte. Il s'agit de l'étiquette associée au nœud 3, notée 3'. Comme précédemment, on liste les arcs de ce dernier et on ajoute les nœuds auxquels il est directement relié.



Étapes 4 à n -1: L'opération précédente est répétée jusqu'à la détermination des trois plus courts chemins.



Étape n: Les 3 plus courts chemins sont classés par ordre croissant des distances de s à t.



Le principe d'optimalisé permet de déterminer les chemins dans l'ordre suivant : le chemin {5, 3', 2", 4", 1""} avec un coût de 4, le chemin {5, 2", 4"', 1^{V}} avec un coût de 5 et enfin le chemin {5, 3', 2", 1'} avec un coût de 6.

V.2.1.2 Amélioration de l'algorithme de Dijkstra généralisé

L'analyse de l'algorithme de Dijkstra généralisé montre que ce dernier peut conduire à un chemin pouvant comporter une ou plusieurs boucles. Pour y remédier, nous introduisons dans l'étape 2 de l'algorithme (Figure V.1) une condition à l'ajout d'un nœud. Pour chaque nœud v_j, on parcourt le chemin en sens inverse en partant de i, jusqu'à atteindre le nœud source. Si le nœud v_j se trouve déjà sur le chemin, il est éliminé de l'arbre d'exploration. La Figure V.3 illustre la nouvelle formulation de l'étape 2 de l'algorithme de Dijkstra généralisé.

```
/* ETAPE 2 modifiée */
if (counti <= K)   alors                          (S'il n'y a pas trop de chemins passant par ce nœud)
   begin
    for each arc(i,j) ∈ i                  (Pour chaque arc (c'est à dire chaque lien) appartenant à ce nœud)
        begin
                          (On vérifie que j n'est pas déjà dans le chemin calculé)
            v=k                           ( Sauvegarde de l'étiquette du nœud traité)
            while (h(v) != s)             (Remonter le chemin jusqu'à la source)
               begin
                  if (h(v) == j)          (Si le nœud correspond au nœud que l'ont souhaite ajouter)
                     begin
                        goto ne_pas_ajouter
                     end
                  v = precedentv          (Remonter au noeud précédent)
               end
             elem = elem + 1                    (Ajouter le noeud situé à l'autre extrémité de l'arc, dans la liste des
                                                 candidats)
            distanceelem = distancek + cij  (Sauvegarde des informations de la nouvelle étiquette)
            precedentelem = k
            h(elem) = j                   (Pour retrouver le nœud auquel appartient l'étiquette)
            h-1(j) = h-1(j) U {elem}
            X = X U {elem}                (Ajouter l'étiquette à la liste des candidats)
           ne_pas_ajouter:
        end
   end
```

Figure V.3 : Modification de l'algorithme de Dijkstra généralisé

Exemple d'application

Pour illustrer le fonctionnement du nouvel algorithme, considérons un réseau composé de cinq nœuds numérotés de 1 à 5 (Figure V.4). Le nœud 1 représente la source et le nœud 5 la destination. Dans cet exemple, les liaisons sont bidirectionnelles.

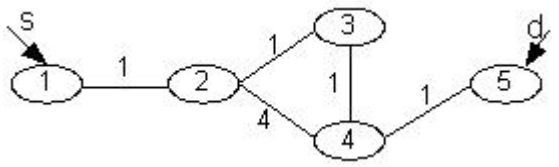
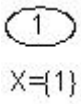
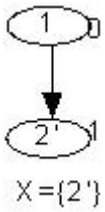


Figure V.4 : Réseau utilisé pour illustrer le calcul des k chemins avec l'algorithme modifié

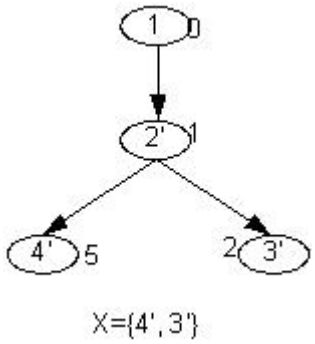
Étape 1: L'ensemble des étiquettes candidates X contient l'étiquette associée au nœud source.



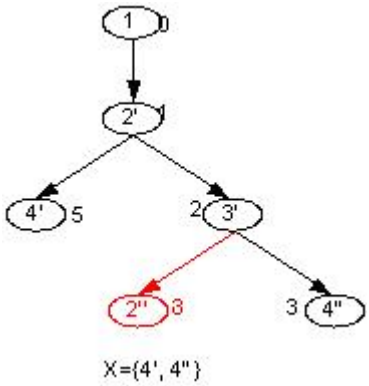
Étape 2 : Tous les arcs partant du nœud 1 sont alors parcourus, et les nœuds situés à l'autre extrémité sont ajoutés à X.



Étape 3 : La recherche de l'étiquette ayant la plus courte distance cumulée conduit à l'étiquette associée au nœud 2, notée 2'. A partir de ce nœud, on dénombre les nœuds voisins 3 et 4 à qui on associe respectivement les étiquettes 3' et 4'.

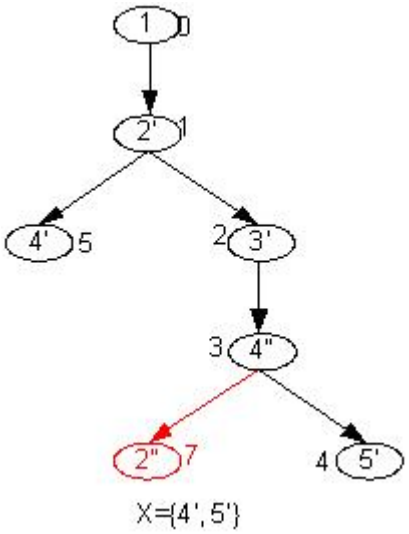


Étape 4 : Le processus précédent est réitéré pour l'étiquette associée au nœud 3, notée 3'. Comme le nœud 2 appartient déjà au chemin parcouru, son étiquette 2" est retirée de l'ensemble X. Ce dernier ne contient alors que les étiquettes 4' et 4" du nœud 4.

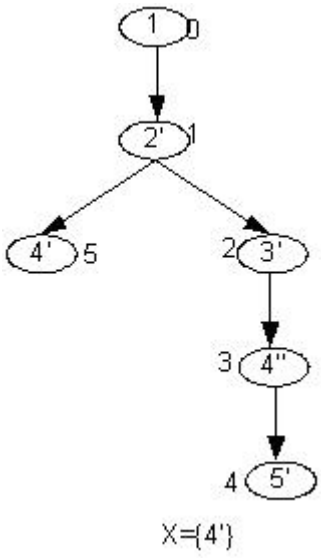


Étape 5 : Dans cette étape, l'étiquette associée au nœud 4, notée 4" est celle dont la distance cumulée est la plus courte. A l'ensemble X sont ajoutés les étiquettes des nœuds auxquels le nœud 4 est directement relié. L'étiquette 2" associée au nœud 2 qui fait partie déjà

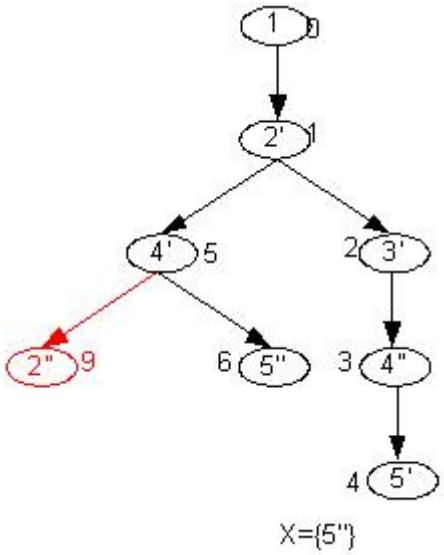
du chemin parcouru est supprimée. L'ensemble X est alors constitué des étiquettes 4' et 5' correspondants respectivement aux nœuds 4 et 5.



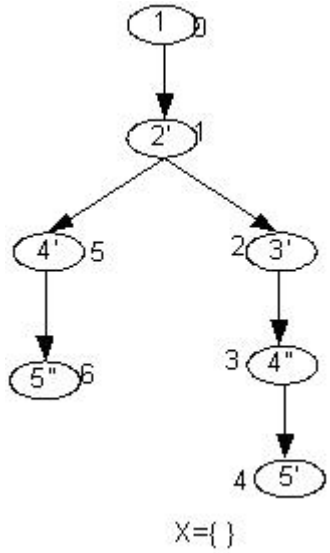
Étape 6 : Dans cette étape, comme le nœud 5 correspond à la fois à l'étiquette ayant la plus courte distance cumulée et au nœud destination, le premier plus court chemin ainsi trouvé a un coût de 4.



Étape 7 : Le processus précédent conduit à l'ensemble X ne comportant que l'étiquette 5" associée au nœud 5.



Étape 8: De la même manière qu'à l'étape 6, cette étape conduit au deuxième plus court chemin avec un coût de 6.



V.2.2 Paramètre de routage : signal de renforcement

Dans la mise en œuvre de l'algorithme généralisé de Dijkstra modifié, le routeur n'a besoin de conserver dans sa table de routage que les informations suivantes :

- Le coût de chaque chemin (en termes de nombre de sauts, par exemple).
- La ligne de sortie associée à chaque chemin (nom du routeur voisin).

La décision d'envoyer un paquet sur l'un ou l'autre de ces chemins se fait a l'aide de l'algorithme Q-Routing qui utilise un signal de renforcement pour la mise à jour des paramètres de routage.

Dans notre approche, chaque nœud x maintient en mémoire une Q-table. Cependant, à la différence de Q-Routing où la Q-table contient les Q-Valeurs de tous les voisins, nous utilisons une Q-table ne comportant que les Q-valeurs des voisins appartenant à l'un des K plus courts chemins. Une telle approche permet de réduire le temps de convergence vers une politique de routage optimal et les ressources mémoires nécessaires. Cette dernière est proportionnelle au produit du nombre d'adresses de destination par le nombre K des plus courts chemins, alors que l'algorithme Q-Routing nécessite un espace mémoire proportionnel au produit du nombre d'adresses de destination par le nombre de voisins.

Dans l'état stationnaire, quand les Q-valeurs dans tous les nœuds représentent l'état réel du réseau, les Q-valeurs de deux nœuds voisins x et y doivent satisfaire les relations suivantes:

L'inégalité générale: si un paquet destiné au nœud d, se trouve au niveau du routeur x pour être routé vers le routeur y, alors :

$$Q(x,y,d) \leq q_y + \delta + Q(y,z,d) \qquad \forall y \in N(x) \;\; \forall z \in N(y) \tag{V.1}$$

L'égalité optimale:

$$Q(x,y,d) = q_y + \delta + Q(y,\tilde{z},d) \tag{V.2}$$

Cette équation est un cas particulier de l'inégalité générale qui montre que le temps optimal pour acheminer un paquet au nœud d, à partir du routeur x via le routeur y voisin de x, est la somme de trois composants: (1) le temps d'attente q_y dans la file d'attente correspondant au routeur y, (2) le délai de transmission entre les nœuds x et y, et (3) le temps optimal $Q(y,\tilde{z},d)$ nécessaire pour délivrer un paquet au nœud d, à partir du routeur y via le routeur $\tilde{z}$.

V.2.3 Mise à jour des Q-valeurs

A partir de l'analyse de l'algorithme Q-Routing présenté dans le chapitre II, nous proposons une nouvelle méthode pour l'exploration du réseau. Celle-ci s'appuie sur la technique d'exploration avancée (forward exploration) (Figure V.5) [BOY94], à laquelle on adjoint un mécanisme d'exploration probabiliste où des probabilités sont associées aux voisins appartenants aux k plus courts chemins. Ce mécanisme nous permet d'une part, d'explorer de temps en temps les chemins non optimaux et d'autre part, d'éviter de surcharger encore plus le réseau puisque les paquets de données sont aussi utilisés pour l'exploration. Cependant, cette méthode n'est pas très adaptée au cas où la charge du réseau est faible puisque certains paquets n'empruntent pas le chemin optimal, retardant ainsi leur acheminement.

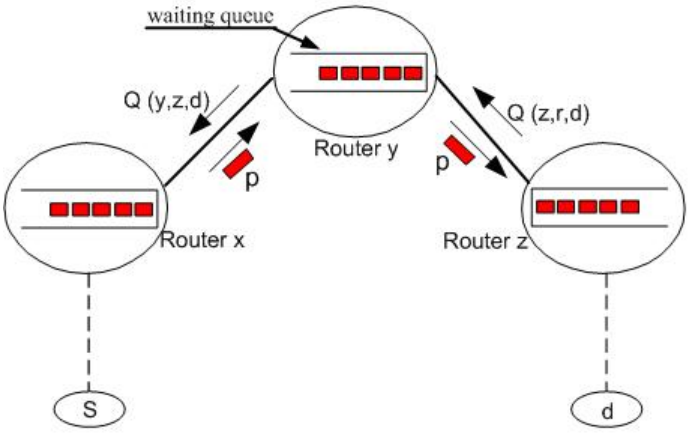


Figure V.5 : Mise à jour dans l'algorithme Q-Routing

Le mécanisme de mise à jour proposé peut être formulé comme suit :

11. Dès qu'un routeur x envoie un paquet *P* destiné au nœud *d* via l'un des routeurs voisins *y*, ce dernier envoie un paquet de renforcement (signal renforcement) au routeur x. Ce paquet inclut l'estimation optimale $Q(y,\tilde{z},d)$ du temps restant pour arriver à la destination *d*.

$$Q(y,\tilde{z},d) = \min_{z \in \text{voisins de } y} \{q_z + \delta + Q(y,z,d)\} \qquad \text{(V.3)}$$

2. Quand un routeur x reçoit le signal de renforcement, il calcule la nouvelle estimation $Q(x,y,d)$ comme suit:

$$Q(x,y,d)^{est} = q_y + \delta + Q(y,\tilde{z},d) \qquad \text{(V.4)}$$

Cette estimation est calculée à partir de l'équation (V.2)

La mise à jour de $Q(x,y,d)$ à partir de la valeur estimée $Q(x,y,d)^{est}$ s'écrit:

$$Q(x,y,d)^{nouv} = Q(x,y,d)^{anc} + \eta(Q(x,y,d)^{est} - Q(x,y,d)^{anc}) \qquad \text{(V.5)}$$

où η représente le pas d'apprentissage (valeur comprise entre 0 et 1).

Le calcul peut être représenté plus simplement :

$$\underbrace{Q(x,y,d)}_{nouvelle Valeur} = \underbrace{Q(x,y,d)}_{ancienne\ valeur} + \eta(\underbrace{(\underbrace{q_y + \delta + Q(y,\tilde{z},d)}_{nouvelle\ estimation} - \underbrace{Q(x,y,d)}_{ancienne\ valeur})}) \qquad \text{(V.6)}$$

Ce calcul correspond en réalité à une moyenne, pondérée par le pas d'apprentissage η, de la nouvelle estimation $Q(x,y,d)^{est}$ et de l'ancienne valeur $Q(x,y,d)^{anc}$.

Remarque

La nouvelle estimation $Q(x,y,d)$ est admissible que si elle satisfait l'équation d'inégalité générale (équation(V.1)) . Or, si l'ancienne valeur $Q(x,y,d)$ satisfait cette inégalité alors la nouvelle estimation la satisfait aussi.

Démonstration

Soit $Q(x,y,d)^{nouv}$ la nouvelle estimation obtenue à partir de l'équation V.6.

Si l'ancienne estimation $Q(x,y,d)$ satisfait l'inégalité générale alors pour tout nœud z voisin de y, appartenant aux k plus courts chemins $Q(x,y,d)$ s'écrit:

$$Q(x,y,d) = q_y + \delta + Q(y,z,d) - \omega(z) \qquad \text{(V.7)}$$

où $\omega(z)$ est une valeur positive.

Comme $Q(y,\tilde{z},d)$ est le minimum de toutes les estimations $Q(y,z,d)$, quelque soit z voisin de y, $Q(y,\tilde{z},d)$ peut s'écrire :

$$Q(y,\tilde{z},d) = Q(y,z,d) - \omega(z,\tilde{z}) \qquad \text{(V.8)}$$

où $\omega(z,\tilde{z})$ est une valeur positive.

A partir des équations (V.6), (V.7) et (V.8), $Q(x,y,d)^{nouv}$ peut s'écrire:

$$Q(x,y,d)^{nouv} = q_y + \delta + Q(y,z,d) - [(1-\eta)\omega(z) + \eta\omega(z,\tilde{z})] \qquad \text{(V.9)}$$

Comme $[(1-\eta)\omega(z) + \eta\omega(z,\tilde{z})]$ est une valeur positive alors:

$$Q(x,y,d)^{nouv} \leq q_y + \delta + Q(y,z,d) \quad \forall y \in N(x) \ \ \forall z \in N(y) \qquad \text{(V.10)}$$

Formulation générale de l'algorithme K-Shortest path Q-Routing

L'algorithme général du K-Shortest path Q-Routing peut être résumé comme suit (Figure V.6):

Au lancement de l'algorithme, on recherche les k plus courts chemins entre une source est une destination. Trois traitements sont possibles en fonction du type de paquet reçu (paquet de donnée, paquet de changement de topologie, paquet de renforcement).

- Si le paquet est un paquet de donnée, un nombre au hasard est tiré pour choisir le routeur qui recevra ce paquet. Un paquet de renforcement est ensuite envoyé au routeur source. Il est à noter que le routeur dont la Q-valeur est minimale a une probabilité P_{max} d'être choisie.

- Si le paquet est un paquet de changement de topologie, on recalcule les k plus courts chemins.

- Si le paquet est un paquet de renforcement, on effectue la mise à jour de la Q-valeur correspondante.

```
Début
    Etiq1 Recherche des k plus courts chemins
    Etiq2 Tant que (paquet non reçu)
                Faire
                  |
                Fait

    Si (paquet= "paquet de renforcement") alors
      Faire
          Mettre à jour la Q-valeur correspondante
          Détruire le paquet de renforcement
      Fait

      Sinon Si (paquet= "paquet de données") alors
                  Faire
                    Calculer  la Q-Valeur optimale.
                    Envoyer  le paquet au routeur correspondant à cette dernière.
                    Envoyer un paquet de renforcement contenant la Q-valeur optimale
                  Fait

                 Sinon ("paquet de changement de topologie")
                  Faire
                    Aller à Etiq1
                  Fait

              Fin de Si

    Fin de Si

    Aller à Etiq2

Fin
```

Figure V.6: algorithme général du K-Shortest path Q-Routing

V.3 Simulations

L'objectif de ces simulations est de comparer les performances de l'algorithme K-Shortest path Q-Routing à ceux des l'algorithmes RIP, Q-Routing et K-Shortest path Routing en terme de temps d'acheminement moyen des paquets. Comme dans le chapitre V, nous proposons d'étudier quatre scénarii : trafic faible, trafic élevé, plusieurs pics de trafic, et tolérance aux pannes.

V.3.1 Réseau utilisé

Pour valider et mettre en évidence les performances de l'algorithme K-Shortest path Q-Routing, nous avons réalisé une simulation sur un réseau irrégulier contenant 32 nœuds (routeur) (Figure V.7), identique à celui utilisé pour la validation de l'algorithme Q-Neural Routing.

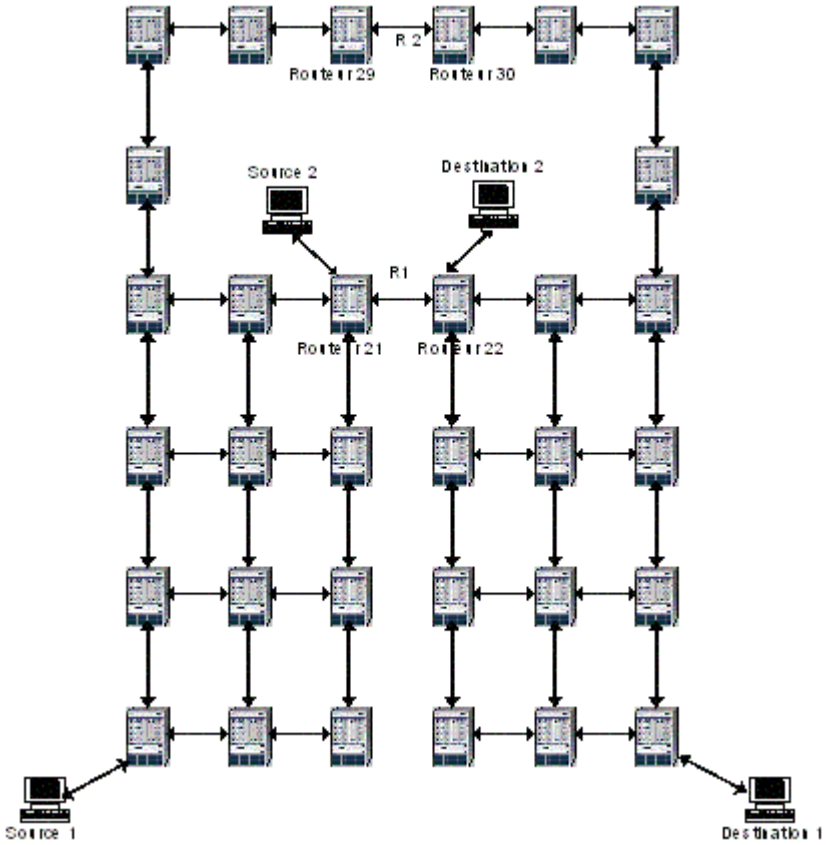


Figure V.7 : Réseau utilisé pour la simulation

V.3.2 Conditions de simulation

Les paramètres utilisés par les algorithmes de routage sont résumés dans le tableau suivant :

Algorithme	Pas d'apprentissage	Nombre k de chemins	Probabilité pour l'exploration	
			Meilleur chemin	Le deuxième chemin
Q-Routing	0.85	---------	----------	----------
K-Shortest path Q-Routing	0.85	2	80%	20%

Les simulations portent sur l'évolution du temps de transit (*end-to-end delay*) des paquets échangés entre deux hôtes : le nœud "source 1" jouera le rôle d'un client et émettra régulièrement des paquets à destination d'un serveur, le nœud "destination 1".

V.3.3 Résultats

V.3.3.1 Trafic faible

La Figure V.8 représente l'évolution du temps moyen d'acheminement (*end-to-end delay*) mesuré par le nœud "destination 1".

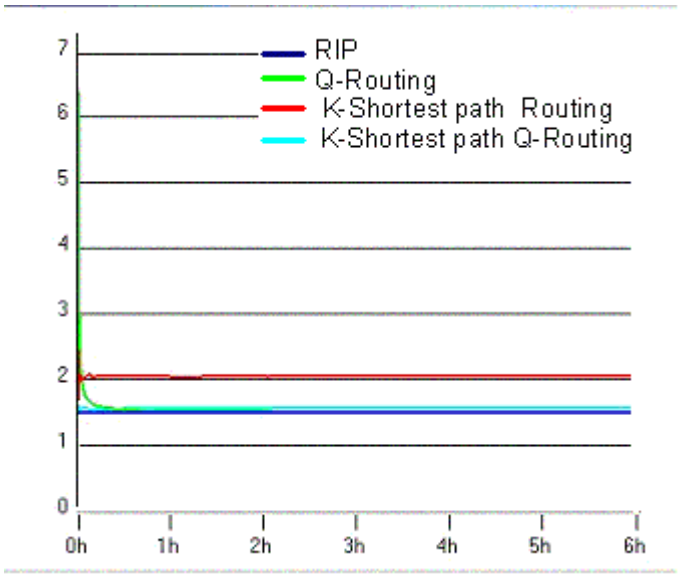


Figure V.8 : Temps d'acheminement moyen pour un trafic faible

A l'exception du K-Shortest path Routing (courbe en rouge), les temps moyens d'acheminement sont pratiquement identiques et constants, ce qui indique que la plupart des paquets empruntent le chemin optimal incluant les routeurs 21 et 22 (Figure V.7). Cependant, au début de la simulation, pour l'algorithme Q-Routing, on constate que le temps

d'acheminement moyen ne se stabilise qu'au bout de 20 minutes puisque le temps de convergence de cet algorithme est largement supérieur aux autres. Enfin, signalons les piètres performances du K-Shortest path Routing, dont les temps moyens d'acheminement sont 30% plus élevés. Les nombreux paquets explorant les chemins alternatifs et sous optimaux expliquent ce résultat.

V.3.3.2 Trafic élevé

Dans ce scénario, le chemin passant par les routeurs 21 et 22 est soumis à un pic de trafic élevé, une heure après le début de la simulation, puis interrompu deux heures plus tard (échange de paquets entre les nœuds "source 2" et "destination 2").

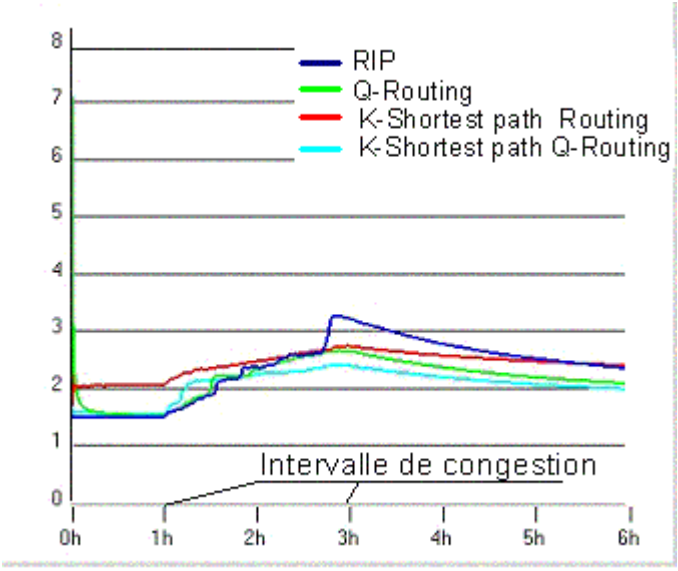


Figure V.9 : Temps moyen pour un trafic élevé

Sous ces conditions, de grandes disparités apparaissent : Les résultats obtenus (Figure V.9) montrent que les routeurs implémentant l'algorithme RIP ignorent complètement la montée en charge sur les routeurs 21 et 22. Les paquets continuent à emprunter la route passant par le lien R1 (Figure V.7), ce qui se traduit par une augmentation du temps moyen d'acheminement entre les nœuds "source1" et "destination2".

Malgré les nombreux paquets empruntant le chemin secondaire (au Nord du réseau), le K-Shortest path Routing ne présente pas de meilleures performances parce qu'il repose sur une méthode probabiliste pour répartir la charge du réseau, est non sur la dégradation des délais d'acheminement. En revanche, le Q-Routing parvient à détecter le pic de trafic. Malgré

un temps de convergence toujours élevé, les routeurs remettent les paquets plus rapidement que les algorithmes précédents. Le K-Shortest path Q-Routing a un meilleur comportement par rapport au Q-Routing.

Dès le début de la quatrième heure, tous les algorithmes réagissent à la diminution du trafic. Les temps moyens d'acheminement du Q-Routing et du K-Shortest path Q-Routing restent inférieurs à ceux des deux autres algorithmes et tendent à converger l'un vers l'autre.

Il faut toutefois noter que les temps de transit utilisés pour construire ces graphes sont des temps moyens depuis le début de la simulation : ceci explique pourquoi, qu'après la quatrième heure, les temps diminuent mais restent néanmoins supérieurs aux valeurs enregistrées pendant la première heure.

Pour caractériser la réactivité du K-Shortest path Q-Routing face à des pics de trafic, nous avons réalisé une seconde simulation d'une durée d'une heure avec un pic de trafic de dix minutes. La Figure V.10 illustre les résultats de cette simulation.

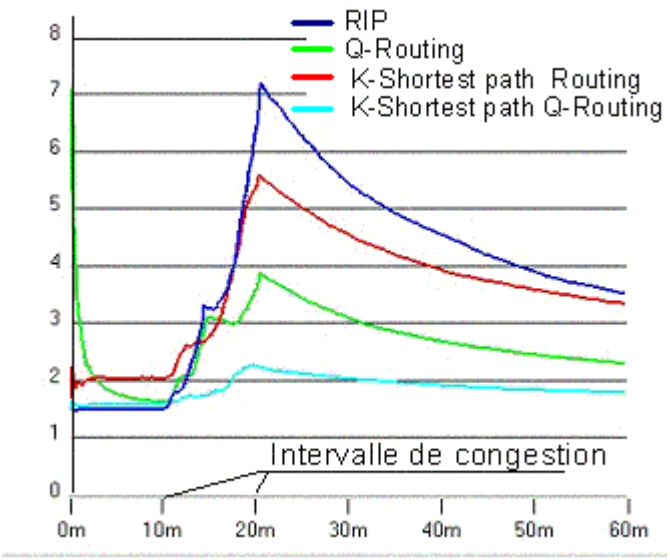


Figure V.10 : Temps de transit avec un pic de trafic de 10 minutes

On voit clairement l'écroulement des performances du RIP et du K-Shortest path Routing quand la charge du réseau devient importante (le pic commence dix secondes après le début de la simulation, et s'arrête au bout de dix minutes).

Par rapport aux deux algorithmes précédents, le Q-Routing présente des temps moyens d'acheminement inférieurs. En effet, cet algorithme réagit rapidement à la montée en charge et route les paquets vers un chemin alternatif. Pendant le pic de trafic, la plupart des paquets émis par le nœud "source 1" empruntent la partie Nord du réseau pour atteindre le nœud "destination 1" et ainsi éviter les liens congestionnés. De plus, l'algorithme Q-Routing converge plus rapidement que l'algorithme RIP lorsque la charge redevient normale.

Enfin, le K-Shortest path Q-Routing présente clairement les meilleures performances. En effet, après une période d'adaptation, les délais sont largement inférieurs aux délais induits par les autres protocoles. L'algorithme sélectionne bien le chemin optimal en conditions de charge élevée, mais aussi lorsque le pic disparaît. C'est l'algorithme qui converge le plus rapidement vers la valeur optimale. Cela est dû au nombre moins élevé de routes explorées par rapport au Q-Routing.

V.3.3.3 Plusieurs pics de trafic élevé

Comme dans le cas du Q-Neural routing, les expériences menées dans cette partie ont pour objectif d'étudier la robustesse et la convergence de l'algorithme proposé. Les courbes de la Figure V.11 montre le comportement des algorithmes face à deux pics de trafic successifs, apparaissant à des instants aléatoires. Pendant une heure, on introduit deux montées en charge du réseau : le premier pic commence dix secondes après le début de la simulation et s'arrête au bout de dix minutes. Trois minutes après, un second pic d'une durée de dix minutes fait son apparition.

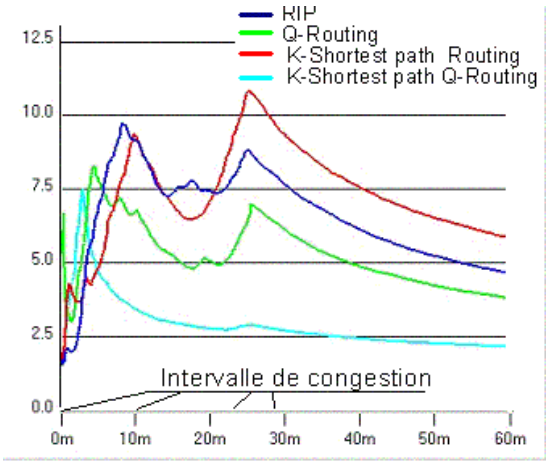


Figure V.11 : Temps de transit avec deux pics de trafic successifs

Ici encore, on constate l'efficacité du Q-Routing face aux algorithmes classiques RIP et K-Shortest path Routing. Cependant, l'algorithme K-Shortest path Q-Routing présente une fois de plus les meilleures performances : son adaptation est extrêmement rapide, et l'apparition du second pic est à peine perceptible.

V.3.3.4 Synthèse des résultats

La Figure V.12 illustre le temps moyen total d'acheminement des paquets envoyés de la "source 1" vers la "destination 1". Dès que la charge du réseau augmente fortement et provoque la congestion du chemin optimal calculé par RIP, l'algorithme K-Shortest path Q-Routing améliore le temps d'acheminement total moyen de près de 57% par rapport à algorithme K-Shortest path Routing, et de près de 45% par rapport à l'algorithme Q-Routing.

	Trafic faible	Trafic élevé sur 6h	Trafic élevé sur 1h	Deux pics de trafic
RIP	1.4567	2.3245	3.9314	6.5005
K-Shortest path Routing	2.0143	2.3848	3.5275	7.3075
Q-Routing	1.4891	2.1260	2.6156	5.2249
K-Shortest path Q-Routing	1.5203	2.0182	1.8257	2.8369

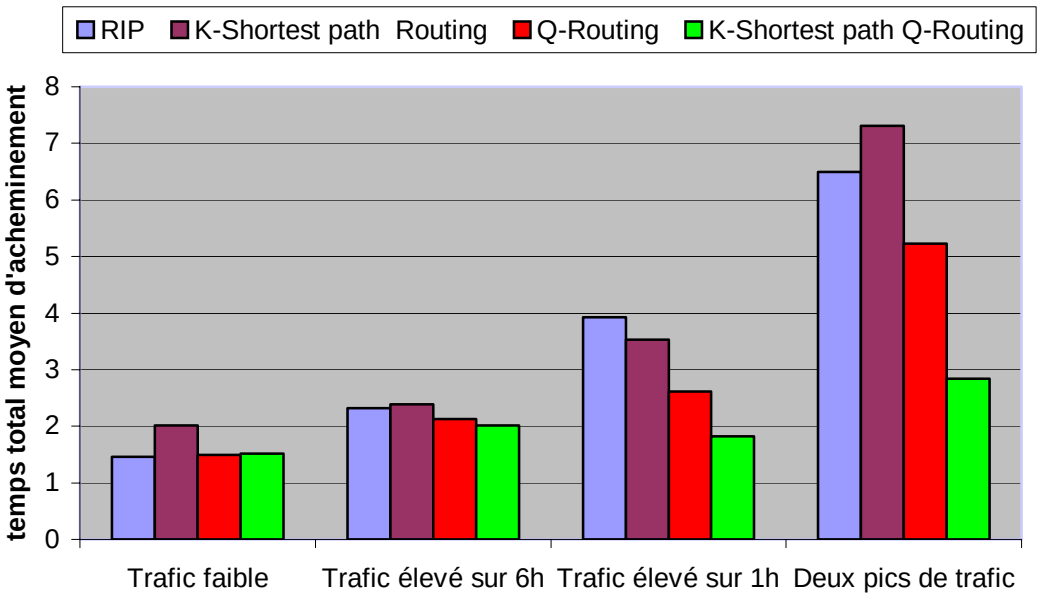


Figure V.12 : Tableau comparatif des temps moyens d'acheminement

V.3.3.5 Tolérance aux pannes

Dans ce scénario, nous étudions la capacité d'adaptation à un changement de topologie du réseau (Figure V.13) de l'algorithme K-Shortest path Q-Routing par rapport à l'algorithme Q-Routing. Chaque nœud source (source 1, source 2, source 3, source 4) choisit au hasard une destination (destination 1, destination 2, destination 3, destination 4) et lui envoie régulièrement des paquets. Au bout d'une heure de simulation, il y a rupture des liens R1 et R2.

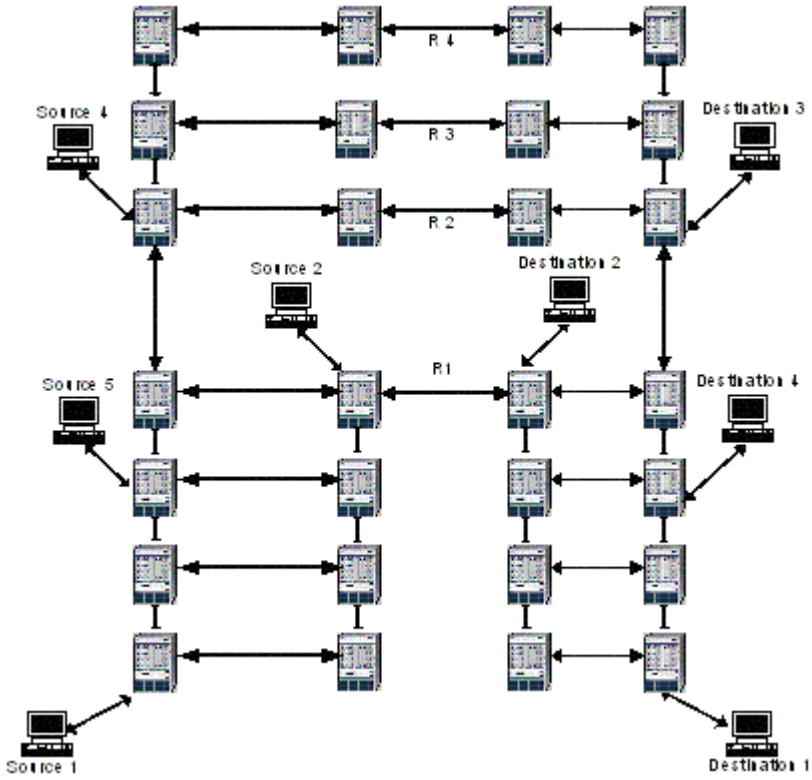


Figure V.13 : Réseau utilisé pour la simulation du scénario de changement de topologie

La Figure V.14 montre que dès que ces liens tombent, provoquant ainsi un changement de topologie du réseau, la moyenne des temps moyens d'acheminement des paquets augmente. Cependant, l'algorithme K-Shortest path Q-Routing présente de meilleures performances que l'algorithme Q-Routing. Ainsi, au bout d'une heure et demi de simulation, le K-Shortest path Q-Routing réduit le temps moyen d'acheminement de près de 38% comparé au Q-Routing. Cela s'explique par le temps relativement élevé que met le Q-Routing pour explorer tout le réseau comparativement au K-Shortest path Q-Routing. Ce dernier

prend en compte plus rapidement le changement de topologie du réseau grâce à un envoi de messages à tous les routeurs.

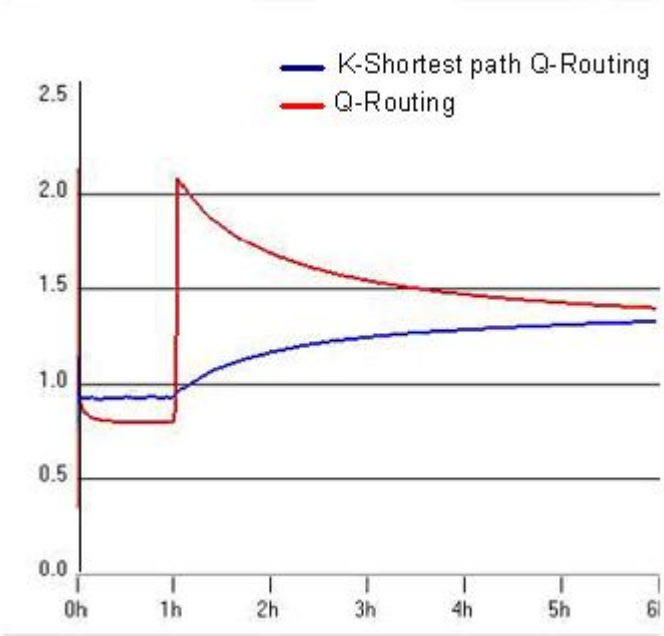


Figure V.14 : Moyenne des temps moyens d'acheminement dans le cas d'un changement de topologie

V.4 Conclusion

Dans ce chapitre, nous avons présenté l'algorithme K-Shortest path Q-Routing qui est basé sur la technique du routage multi-chemin combiné avec l'algorithme Q-Routing où l'espace d'exploration est réduit aux k meilleurs chemins, minimisant le nombre de sauts. Comparé au Q-Routing, l'algorithme proposé ne nécessite qu'un espace mémoire proportionnel au produit du nombre d'adresses de destination par le nombre K des plus courts chemins. L'algorithme de recherche des K plus courts chemins est basé sur l'algorithme de Dijkstra généralisé auquel on adjoint un mécanisme de suppression de boucles. Le chemin optimal correspond à celui dont le temps d'acheminement moyen est le plus court. Le mécanisme d'exploration utilisé pour la mise à jour des Q-Valeurs repose sur une méthode hybride associant la technique de l'exploration avancée à chaque fois qu'un paquet de donnée est échangé entre routeurs, et celle de l'exploration probabiliste pour l'exploration des k-1 chemins restants.

Les simulations réalisées sur différentes topologies de réseau montrent que dans le cas d'une forte charge de ce dernier, l'algorithme K-Shortest path Q-Routing améliore le temps d'acheminement total moyen de près de 57% par rapport à l'algorithme K-Shortest path Routing, et de près de 45% par rapport à l'algorithme Q-Routing. Cependant, dans le cas d'une faible charge du réseau, l'algorithme proposé augmente le temps d'acheminement total moyen de près de 5% par rapport à l'algorithme Q-Routing du à l'utilisation de certains paquets de données pour l'exploration.

Conclusion Générale

Au terme de ce manuscrit, nous nous proposons de faire un récapitulatif de notre travail, d'analyser globalement les résultats obtenus et enfin de dresser des perspectives qui permettront d'envisager de nouveaux axes de recherche comme une suite logique de ce travail.

Nous avons présenté dans cette thèse deux nouvelles approches algorithmiques permettant une meilleure prise en compte de la QoS dans les décisions de routage, dans un réseau à trafic irrégulier. Pour appuyer notre démarche, nous avons effectué une analyse critique sur les différents travaux de recherche qui ont eu pour objectifs de développer et d'améliorer les performances des méthodes de routage adaptatif avec QoS. Cette analyse nous a conduit à retenir comme base de travail, l'algorithme Q-Routing qui s'appuie sur la technique d'apprentissage par renforcement. En effet, cet algorithme prend en compte les fluctuations de la charge du réseau et le type d'apprentissage utilisé est bien adapté à la problématique du routage puisque le modèle représentant l'environnement (le réseau) dans lequel se situe le routeur est a priori inconnu. En revanche, cet algorithme présente des limitations dues à la technique d'exploration utilisée et à la métrique employée, en l'occurrence, le temps de bout en bout, pour le calcul du chemin optimal. En effet, il ne permet pas l'exploration de chemins devenus optimaux, et par conséquent, ne met à jour que rarement, certaines informations de routage, ce qui rend ces dernières peu fiables lors d'une décision de routage.

Pour remédier aux inconvénients des techniques utilisant le Q-Routing des points de vue de la rapidité de convergence vers la solution optimale et de la réduction de l'espace mémoire occupé par la table de routage, nous avons proposé deux algorithmes de routage adaptatif : l'algorithme Q-Neural Routing et l'algorithme K-Shortest path Q-Routing.

L'algorithme Q-Neural Routing, est basé sur un apprentissage par renforcement où l'estimation et la mise à jour des Q-Valeurs sont effectuées à l'aide d'un réseau de neurones. L'utilisation de ce dernier permet d'une part, l'intégration d'informations contextuelles comme l'état des files d'attente des voisins pour anticiper d'éventuelles congestions des routeurs et d'autre part, l'utilisation d'un espace mémoire représentant la Q-table, indépendant du nombre de destinations. Le mécanisme d'exploration utilisé pour la mise à jour des Q-Valeurs s'appuie sur la technique de l'exploration avancée (forward exploration). Celle-ci s'avérant insuffisante, nous avons proposé de lui adjoindre un mécanisme

d'exploration par inondation permettant une mise à jour régulière des Q-Valeurs des routeurs appartenant aux chemins non utilisés.

L'algorithme K-Shortest path Q-Routing est quant à lui basé sur la technique du routage multi chemin combiné avec l'algorithme Q-Routing où l'espace d'exploration est réduit aux k meilleurs chemins, minimisant le nombre de sauts. Comparé au Q-Routing, l'algorithme proposé ne nécessite qu'un espace mémoire proportionnel au produit du nombre d'adresses de destination par le nombre k des plus courts chemins. L'algorithme de recherche des k plus courts chemins est basé sur l'algorithme de Dijkstra généralisé. Afin de lever le problème de l'apparition des boucles inhérent à ce type d'algorithme, nous avons proposé une heuristique permettant de supprimer ces boucles. Le chemin optimal est celui qui présente le temps d'acheminement moyen le plus court. Le mécanisme d'exploration utilisé pour la mise à jour des Q-Valeurs repose sur une méthode hybride associant la technique de l'exploration avancée, à chaque fois qu'un paquet de donnée est échangé entre routeurs, et celle de l'exploration probabiliste, pour l'exploration des k-1 chemins restants.

La validation et l'évaluation des performances des algorithmes proposés ont été réalisées en simulation à l'aide du logiciel OPNET, sur différentes topologies de réseaux à trafic irrégulier. Une analyse comparative des performances de ces deux algorithmes par rapport à celles de l'algorithme Q-Routing a été réalisée. Elle montre clairement l'efficacité des algorithmiques proposées et leur intérêt des points de vue du temps moyen d'acheminement pour des réseaux à fortes charges ou soumis à des pics de trafic et de la tolérance aux pannes. En effet, dans ces conditions, les algorithmes Q-Neural Routing et K-Shortest path Q-Routing améliorent les temps moyens d'acheminement, obtenus avec l'algorithme Q-Routing, de facteurs pouvant atteindre respectivement 38% et 45%.

Dans le cadre de ces travaux de thèse, notre objectif a été d'assurer une adaptabilité de la décision du routage vis-à-vis de trafics irréguliers pouvant inclure différents types de flux, le marché actuel demandant de plus en plus à concentrer l'ensemble de ces flux sur une même infrastructure réseau. Les approches actuelles sont basées sur l'intégration d'une connaissance a priori. Notre approche, de par l'utilisation des réseaux de neurones, permet une meilleure prise en compte du contexte sans pour autant trop charger le réseau. Ce travail de thèse a ainsi permis de montrer :

- la pertinence de l'apprentissage par renforcement dans un environnement inconnu a priori.

- le prise en compte de la QoS par l'intégration de paramètres dynamiques dans la prise de décision concernant le choix des routes (« *laisser le réseau s'auto-organiser* »).

Les perspectives d'évolution de ce travail s'inscrivent dans la continuité du travail déjà accompli. Il s'agit en premier lieu d'étudier le comportement des algorithmes proposés en intégrant plusieurs paramètres de QoS (bande passante, nature des flux, taux de perte, etc.).

La seconde perspective s'inscrit dans l'optique d'une extension de ces approches pour les réseaux mobiles. En effet, l'intégration voix/vidéo/données et la diversité des services exigés par le marché actuel sont orientées de plus en plus vers la prise en compte de l'aspect nomade et mobile du terminal usager. La recherche de multiples routes intégrant la QoS apparaît alors comme indispensable.

Références Bibliographiques

[ABD94] H. Abdi, "Les Réseaux de neurones, Sciences et technologies de la connaissance", PUG, Grenoble, 1994.

[ALM92] P. Almquist, "Type of Service in the Internet Protocol Suite", RFC 1349 July 1992.

[APO99] G.Apostolopoulos, S. Kamat, R. Guérin, A. Orda, T. Przygienda, and D. Williams, "QoS Routing Mechanisms and OSPF extensions", RFC 2676 August 1999.

[ART93] T. Artieres, P. Gallinari, "Neural models for extracting speaker characteristics in speech modelization systems" In EuroSpeech, pp 2263-2266, 1993.

[BAK98] F. Baker, S. Brim, T. Li, F. Kastenholz, S.Jagannath, J. K. Renwick, "IP Precedence in Differentiated Services Using the Assured Service", IETF Internet Draft, 1998.

[BEN98] Y. Bennani, P. Gallinari, "Connectionist approaches for automatic speaker recognition", ESCA workshop on speaker recognition, Martini, Suisse 1998

[BER98] Y. Bernet, "Requirements of Diff-serv Boundary Routers", IETF Internet Draft, 1998.

[BOY94] J. A. Boyan and M. L. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach." In Cowan, Tesauro and Alspector (eds), Advances in Neural Information Processing Systems 6, 1994.

[BRA97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification". RFC 2205, IETF, Sep 1997.

[CAL90] R. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments", RFC 1195 December 1990.

[CAS93] F. Castel, "Les télécommunications, dirigé par", éditions Berger-Levrault International, 1993.

[CIS] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ibmsna.htm#xtocid8

[CRA98] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC2386, IETF, August 1998.

[CUL43] M. Culloch, W.S. and Pitts, "A logical calculus of the ideas immanent innervous activity". Bull. Math. Biophy. 5, PP 115-133.

[CYB89] G. Cybenko, "Approximation by superposition of sigmoïdal functions" In Math. of Control, Signals and Systems, 2, n° 4, 1989.

[DAR81] "Transmission Control Protocol", Darpa Internet Program, Protocol Specification, September 1981.

[DEM89] A. Demera, S. Keshav, S. Shenker, "Design and Analysis of a Fair Queueing Algorithm", ACM SIGCOMM'89, Austin, 1989.

[EPP99] D. Eppstein, "Finding the K Shortest Paths", SLAM J. Computing 28:0(1999) pp. 652-673.

[FEN97] W. Feng, D. Kandlur, D. Saha, K. Shin, "Understanding TCP Dynamics in an Integrated Services Internet", NOSSDAV '97, 1997.

[FER97] P. Ferguson, "Simple Differential Services: IP TOS and Precedence, Delay Indication and Drop Preference", IETF Internet Draft, 1997.

[FLO93] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", ACM/IEEE Transactions on Networking, 397-413, August 1993.

[GAL99] P. Gallinari, T. Cibas, "Practical complexity control in multilayer perceptrons". Signal Processing, vol.74, 29-46, 1999.

[GEL00] E. Gelenbe, R. Lent, Z. Xu "Towards networks with cognitive packets", Proc. IEEE MASCOTS Conference, ISBN 0-7695-0728-X, pp. 3-12, San Francisco, CA, Aug. 29-Sep. 1, 2000.

[GEL04] E. Gelenbe, R. Lent et Z. Xu, "Measurement and performance of cognitive packet networks" In Jour. Comp. Networks, Vol. 37, 691-701, 2001

[GOE96] P. Goetz, S. Kumar, R. Miikkulainen, "On-Line Adaptation of a Signal Predistorter through Dual Reinforcement Learning", Proc Machine Learning, Proceedings of the 13th Annual Conference Bari, Italy 1996.

[GUE96] R. Guerin, A. Orda, D. Williams, "Quality of Service Routing Mechanisms and OSPF Extensions", IETF Internet Draft, 1996.

[HAR96] M.E. Harmon, "Reinforcement Learning: A Tutorial". WL/AACF , 17 Décembre 1996.

[HAW96] J. Hawkinson T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", RFC 1930 March 1996.

[HED88] C. Hedrick, "Routing Information Protocol", RFC 1058 1988.

[HEI99] J. Heinanen, F. Bakern, W. Weiss, "Wroclawski, Assured Forwarding PHB Group", RFC 2597 June 1999.

[HOP82] John J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences, 79:2554-2558, 1982.

[HOR91] K. Hornik, "Approximation capabilities of multilayer feed-forward networks" In Neural Networks, Vol. 4, n° 2, pp 251-258, 1991.

[HSU94] C. Hsu and J. Y. Hui, "Load-Balanced K-Shortest Path Routing for Circuit-Switched Networks", In Proceedings of IEEE NY/NJ Regional Control Conference, August 1994.

[HUI95] C. Huitema, "Le Routage dans Internet", Edition Eyrolles 1995.

[INT95] "RSVP Showcase", Interop & Networld September, 1995.

[ISO] http://www.ISO.org.

[ISO90] K. Iso, T. Watanabe, "Speaker-independent word recognition using a neural prediction model" In ICASSP, pp 441-444, 1990.

[JAC88] V. Jacobson, "Congestion Avoidance of Network Traffic", Computer Communication Review, vol. 18, no. 4, pp. 314-329, 1988.

[JAC99] V. Jacobson, K. Nichols, "An Expedited Forwarding PHB", RFC 2598, June 1999.

[KAE96] L. P. Kaelbling, M.L. Littman. A.W. Moore, "Reinforcement learning : A survey", Journal of Artificial Intelligence Research, Vol 4 :237-285, 1996.

[KUM97] S. Kumar, R. Miikkulainen, "Dual Reinforcement Q-Routing: An on-line adaptive network routing algorithm", Artificial Neural Networks in Engineering, Nov. 1997.

[KUM98a]S. Kumar and R. Miikkualainen, "Confidence-based Q-routing: an on-queue adaptive routing algorithm", In Proceedings of Neural Networks in Engineering, 1998.

[KUM98b]S. Kumar, "Confidence based Dual Reinforcement Q-Routing: An On-line Adaptive Network Routing Algorithm", Master's thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX-78712, USA Tech. Report AI98-267, 1998.

[KUM99] S. Kumar and R. Miikkulainen, "Confidence based Dual Reinforcement Q-Routing: An adaptive online network routing algorithm", International Joint Conference on Artificial Intelligence, Stockholm, Sweden, July 1999.

[LIT'94] M.L. Littman, "Markov games as a framework for multi-agent reinforcement learning". In proceedings of the eleventh international conference on machine learning, san francisco 1994 pp.157-163.

[LIP87] R.P. Lippmann, "An introduction to computing with neural nets" In IEEE ASSP Magazine, Vol. 4, n° 2, pp 4-22, 1987.

[LON90] J. Loncell, "Contribution des réseaux connexionnistes au traitement d'images bas niveau" Thèse de doctorat, Université de Paris-Sud, Novembre, 1990.

[LOT'92] M. Lottor, "Internet Growth (1981-1991)", RFC 1296 January 1992.

[LOU91] K. Lougheed Y. Rekhter, "A Border Gateway Protocol 3 (BGP-3)", RFC 1267 October 1991.

[MAL93] G. Malkin, "RIP version2: Carrying Additional Information", RFC 1388 RFC 1993.

[MAR98] E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Santos, "The K Shortest Paths Problem", Research Report, CISUC, June 1998.

[MCC43] W.S. McCullogh et W.H. Pitts, "A Logical Calculus for the ideas imminent in nervous activity", Bulletin of Math. Biophysics, Vol.5, pp.115-133, 1943.

[MCK91] J.C McKay, "A practical bayesian framework for backprop networks" In Neural Computation, 1991.

[MEL90] A. MELLOUK, "Quelques approches pour la reconnaissance automatique de la parole continue; développement d'un système basé sur des réseaux prédictifs non-linéaires" Rapport LRI no 715- Janvier 92- Universite Paris-Sud, 1992.

[MEL94] A. MELLOUK, "Etude des Systèmes modulaires Neuro-Prédictif : application à la Reconnaissance Automatique de la Palire Continue", Thèse de Doctorat Université Paris-Sud, Orsay, 1994.

[MIL84] D.L. Mills, "Exterior Gateway Protocol Formal Specification", RFC 904 April 1984.

[MIN68] Minski et S.Papert, "Perceptrons", the M.I.T. Press, 1969.

[MOY89] J. Moy, "OSPF specification", RFC 1131 1989.

[MOY98] J. Moy, "OSPF Version 2", RFC 2328 1998.

[NIC98] K. Nichols et S. Blake, "Differentiated Services Operational Model and Definitions", IETF Internet Draft, 1998.

[NOW91] S.J. Nowlan., G.E. Hinton, "Soft weight-sharing" In Advancesin Neural Inf. Proc. Sys., Vol. 4, 1991.

[OSI94] Technologies de l'information -- Interconnexion de systèmes ouverts (OSI) -- Modèle de référence de base: Le modèle de base ISO/IEC 7498-1:1994.

[PAR92] C. Partridge, "A proposed flow specification", RFC1363, IETF, Septembre 1992.

[PAR94] A. Parekh, "A Generalized Processor Sharing Approach to Flow Control – The Multiple Node Case", IEEE/ACM Transactions on Networking, pp. 137-150, 1994.

[PER90] L. Personnaz, O. Nerrand et G. Dreyfus, Journées Internationales des sciences de l'informatique, Tunis, 1990.

[PIE00] S. Pierre, H. Said et W.G. Probst, "Routing in computer networks using artificial neural networks", Artificial Intelligence in Engineering 14 (4) (2000) pp. 295-305.

[PIE01] S. Pierre, H. Said et W.G. Probst, "An Artificial Neural Network Approach for Routing in Distributed Computer Networks", Engineering Applications of Artificial Intelligence, Vol. 14 (2001), Elsevier Science, pp. 51-60.

[PLU82] David C. Plummer, "An Ethernet Address Resolution Protocol", RFC826 November 1982.

[POS81] J. Postel, "Internet Protocol", Darpa Internet Program, Protocol Specification, RFC 791 DARPA September 1981.

[PUJ02] G. Pujolle, "Les Réseaux", Eyrolles 3eme édition, 2002.

[RON01] E. Rondeau, "Conception d'architectures de réseaux locaux par analyse du trafic". Habilitation à Diriger des Recherches, Université Henri Poincaré - Nancy 1, 30 novembre 2001

[ROS99] E. Rosen, A. Viswanathan et R. Callon, "Multiprotocol Label Switching Architecture", Internet Draft draft-ietf-mpls-arch-06.txt, August 1999.

[RUM86] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "in Parallel Distributed Processing". M.I.T. Press, vol.1 1986.

[SHE97] S. Shenker, C. Partridge et R Guerin, "Specification of guaranteed quality of service", RFC2212, septembre 1997.

[SHR95] M. Shreedhar et G. Varghese, "Efficient Fair Queuing using Deficit Round Robin", In Proceedings of SIGCOMM'95, 1995.

[STA01] W. Stallings, "MPLS", Internet Protocol Journal, Vol. 4, nr. 3, Septembre 2001.

[STE97a] K. Steenhaut, M. Fakir, A. Nowé et E. Dirkx, "Reinforcement learning for revenue optimisation in ATM networks: a case study based on simulation". Proceedings of the IASTED International Conference, Modelling and Simulation, Pittsburgh Pennsylvanian USA, May 15-17. pp 52-56.

[STE97b] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001 1997.

[SUB97] D. Subramanian, P. Druschel et J. Chen, "Ants and reinforcement learning: a case study in routing in dynamic networks", in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI_97), Morgan Kaufmann, San Francisco, CA, 1997, pp. 832–839.

[SUT88] R. S. Sutton, "Learning to Predict by the Method of Temporal Differences", Machine Learning,Vol.3, p 9-44, September 1988.

[SUT97] R. S. Sutton, A. G. Barto, "Reinforcement Learning", MIT Press, 1997.

[TCC90] H.Thomas, Cormen, Charles E. Leiserson, et Ronald L. Rivest, "Introduction to Algorithms", MIT Press, 1990 page 527.

[TEB90] J. Tebelskis, A. Waibel, "Large vocabulary recognition using linked predictive neural networks" In ICASSP, pp 437-440, 1990.

[THO90] H. Thomas, Cormen, Charles E. Leiserson, et Ronald L. Rivest, "Introduction to Algorithms", MIT Press, 1990 page 532.

[THR92] S. Thrun, "The role of exploration in learning control", In Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches, Van Nostrand Reinhold, D. A. White and D. A. Sofge (eds), 1992.

[TUR86] J. Turner, "New directions in communications (or which way to the information age) ", IEEE Communications Magazine, 24(10), 1986.

[VIL99a] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)", Internet Draft draft-ietfospf-omp-03, August 1999.

[VIL99b] C. Villamizar, "MPLS Optimized Multipath (MPLS--OMP)", Internet Draft draftietf-mpls-omp-00, August 1999.

[WAN95] Z. Wang et J. Crowcroft, "Bandwidth-Delay Based Routing Algorithms", IEEE GlobeCom'95, Singapore, Nov 1995.

[WAT89] D. Watkins, "Q-learning", Machine Learning 8 279-292 1992.

[WER74] P.J. werbos, " Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciencs ", Ph.D Thesis, Harvard University, 1974.

[ZAP98] D. Zappala, J. Kann, "RSRR: A Routing Interface for RSVP". Internet Draft draft-ietf-rsvp-routing-02, IETF, Jun 1998. Work in Progress.

[ZHA93] L. Zhang, S. Deering, D. Estrin et D. Zappala, "RSVP: A New Resource ReSerVation Protocol". *IEEE Network*, vol. 7, No 5, p. 8–18, Sep 1993.

[ZHA97a] Zhang, Sanchez, Salkewicz et Crawley, "Quality of Service Extensions to OSPF or Quality Of Service Path First Routing (QOSPF)", Internet Draft draft-zhang-qosospf-01.txt, September 1997.

[ZHA97b] P. Zhang, "Etude de Différents Aspects de l'Apprentissage par Renforcement", Thèse Université de Technologie de Compiègne, 1997.

Annexe I

Datagramme IP

Dans la couche réseau, le protocole IP (*Internet Protocol*)[POS81] est au cœur du fonctionnement des réseaux IP. Les informations devant être transmises d'un point à un autre du réseau sont fragmentées en paquets (datagramme IP). IP est chargé d'acheminer ces paquets jusqu'à leur destination. Pour cela, avant de les envoyer, il joint à chaque paquet un ensemble d'informations, appelé en-tête du paquet IP (Figure I.15).

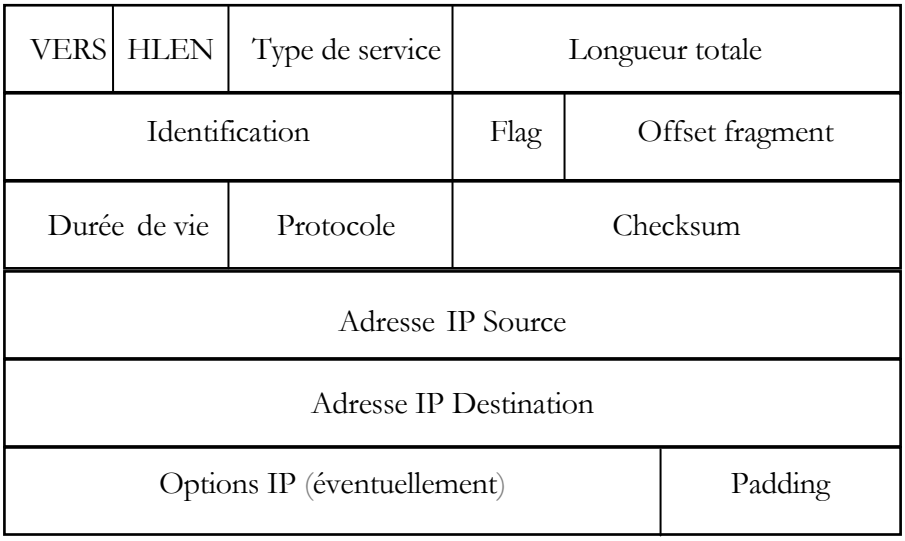


Figure I.15*:* En-tete IP

- **VERS** : numéro de version du protocole IP, actuellement version 4.
- **HLEN** : longueur de l'en-tête en mots de 32 bits, généralement égal à 5 (pas d'option).
- **Type de service** : indique les contraintes à prendre en compte dans le routage des paquets [ALM92].
- **Longueur totale** : longueur totale du datagramme (en-tête + données). Dans le cas d'un paquet fragmenté, il s'agit de la taille de ce fragment et non pas celle du datagramme initial.
- **IDENTIFICATION** : entier qui identifie le datagramme initial.

- **FLAGS :** C'est le champ qui permet au destinataire final de reconstituer le datagramme initial en identifiant les différents fragments (milieu ou fin du datagramme initial). Il contient un bit appelé "*do not fragment*" (01X) et un autre bit appelé "*More fragments*" (FLAGS = 001 signifie d'autres fragments à suivre)

- **Durée de vie** : Ce champ indique en secondes, la durée maximale de transit du datagramme sur l'internet. La machine qui émet le datagramme définit sa durée de vie.

- **Protocole** : Ce champ identifie le protocole de niveau supérieur dont le message est véhiculé dans le champ de données du datagramme (6: TCP, 17: UDP, 1: ICMP).

- **Checksum :** Ce champ permet de détecter les erreurs survenant dans l'entête du datagramme, et par conséquent son intégrité.

- **Offset fragment** : indique le déplacement (position) des données contenues dans le fragment par rapport au datagramme initial.

Annexe II

Implémentation sur OPNET de l'algorithme K-Shortest paths Q-Routing

Le Process Model (Figure V.7) du K-Shortest paths Q-Routing apparaît comme une combinaison du K-Shortest paths Routing et du Q-Routing. A la réception d'un paquet de données, il faut le faire suivre : le routeur extrait l'adresse de destination et enregistre le numéro de l'interface d'arrivée. La décision de routage peut alors se faire de deux manières :

- Si les tables de Q-Routing n'ont pas encore convergé, le routeur utilise la table calculée par l'algorithme des k plus courts chemins pour choisir le plus court chemin en termes de nombres de sauts.
- Dans le cas contraire, la Q-table est interrogée et la route présentant le plus court délai est choisie. Un paquet de renforcement est alors envoyé sur l'interface d'arrivée.

La table de routage doit donc être modifiée pour enregistrer à la fois les données calculées par l'algorithme des k plus courts chemins et les signaux de renforcement du Q-Routing :

```
struct table_routage
  {
      int destination;
      int nbre_chemins;
      /* Index de l'interface de sortie */
      int sortie[MAX_PATH];
      /* Cout du chemin (Calculé par l'algo du plus court chemin) */
      int cout[MAX_PATH];
      /* Delai calculé par le Q-Routing */
      double delay[MAX_PATH];
  };
```

Le tableau *delay[ ]* a pour rôle de sauvegarder les valeurs des signaux de renforcement calculés à partir des paquets de renforcement envoyés par les routeurs voisins.

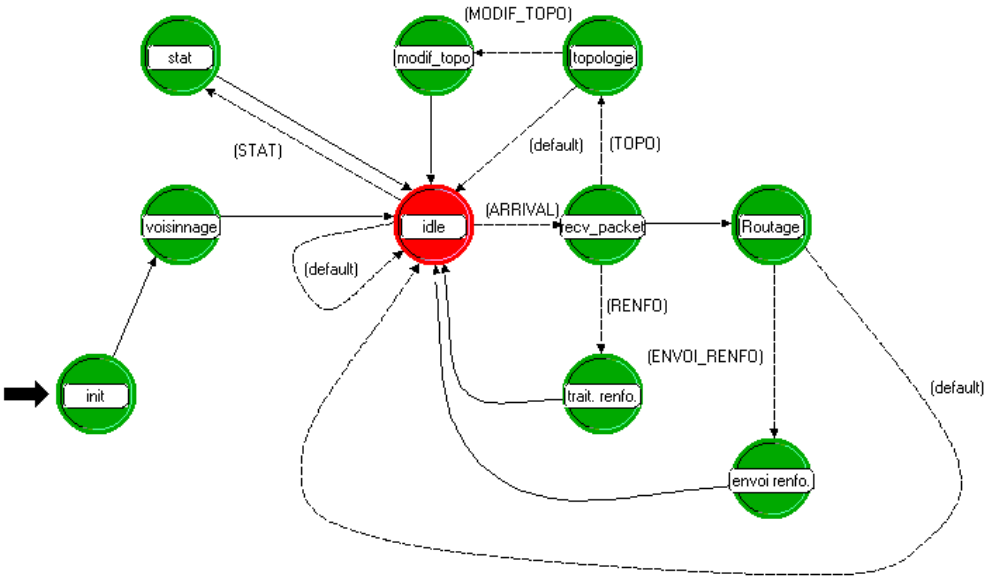


Figure V.7 : Le Process Model du K-Shortest paths Q-Routing

Le fonctionnement de l'automate à états finis K-Shortest paths du Q-Routing se fait comme suit:

La phase d'***initialisation*** (*init*) est exécutée une seule fois au début de la simulation. Elle permet au routeur de :

- Initialiser ses variables globales
- Découvrir ses paramètres de simulation (adresse locale…)
- Allouer des ressources pour collecter les statistiques
- Découvrir les interfaces actives et éventuellement le coût associé

L'étape de ***voisinage*** permet à chaque routeur de signaler sa présence auprès de ses voisins. En effet, la première phase leur permet de découvrir les réseaux auxquels ils sont connectés, mais pas toujours l'adresse des routeurs situés à proximité. L'émission d'un paquet de voisinage sur toutes les interfaces permet de résoudre ce problème. La seule information importante transportée par ce paquet est l'adresse locale du routeur. La phase d'initialisation terminée, les routeurs se mettent en état ***d'attente*** (*idle*).

En recevant un paquet de topologie (*TOPO*), un routeur extrait l'adresse de l'émetteur, et met à jour sa base de données, puis construit un paquet de topologie qu'il envoie sur toutes ses interfaces actives. Ce paquet indique l'existence d'un lien entre deux routeurs du réseau, les adresses de ces routeurs étant contenues dans les champs ***source* et *destination***. La réception d'un tel paquet provoque la même action qu'un paquet de voisinage : mise à jour de la base de données et propagation du paquet sur les autres interfaces actives. Cependant, un mécanisme est prévu pour éviter que les paquets de topologie soient détruits dès qu'ils sont inutiles (c'est à dire dès que tous les routeurs ont pris connaissance de l'information) : le paquet n'est pas réémis s'il contient des informations déjà présentes dans la base de données du récepteur.

Si la condition ***MODIF_TOPO*** est réalisée (si la base de données a été modifiée), l'automate passe dans l'état ***modif_topo***, recalcule les tables de routage, envoie l'information sur toutes ses interfaces et retourne dans l'état ***idle*** pour attendre l'arrivée d'un nouveau paquet. Si la condition n'est pas vérifiée, la propagation du paquet est inutile et l'automate passe directement à l'état ***idle***.

Le calcul des k plus courts chemins est réalisé en respectant les contraintes évoquées précédemment. Lorsqu'un nouveau chemin est découvert, on crée une entrée dans la table de routage. L'enregistrement du chemin complet étant inutile, seuls l'interface locale et le coût du chemin sont sauvegardés.

La dernière partie de l'automate concerne le routage proprement dit : par défaut, la réception d'un paquet fait passer le routeur dans l'état *routage*.

Il extrait alors l'adresse de destination du paquet, puis effectue une recherche linéaire dans la table de routage.

Une entrée dans la table est définie comme suit :

```
#define MAX_INTERFACE 4
struct table_routage
  {
      /* Adresse de destination */
      int destination;
      /* Nombre de chemins calculés */
      int nbre_chemins;
      /* Index de l'interface de sortie */
      int sortie[MAX_INTERFACE];
      /* Cout du chemin (Calculé par l'algo du plus court chemin) */
      int cout[MAX_INTERFACE];
  };
```

La recherche prend fin lorsque le routeur a trouvé une entrée correspondante au sous-réseau de destination. Il reste encore à choisir un chemin parmi les *nbre_chemins* calculés. Pour des raisons de commodité, les chemins sont classés par ordre croissant dans les tableaux *sortie[]* et *cout[]*.

Le choix se fait de la manière suivante :

On assigne la probabilité P au chemin le plus court et (1 - P) à l'ensemble des autres chemins, puis on tire un chiffre au hasard, compris entre 0 et 1. Cette valeur permet alors de sélectionner un chemin dans la liste.

Ainsi, tous les chemins calculés sont explorés de manière probabiliste. En donnant à P une valeur suffisamment grande, on s'assure que la plupart des paquets emprunteront le chemin considéré comme optimal.

Résumé

L'objectif de ce travail de thèse est de proposer des approches algorithmiques permettant de traiter la problématique du routage adaptatif (RA) dans un réseau de communication à trafic irrégulier. L'analyse des algorithmes existants nous a conduit à retenir comme base de travail l'algorithme Q-Routing (QR); celui-ci s'appuie sur la technique d'apprentissage par renforcement basée sur les modèles de Markov. L'efficacité de ce type de routage dépend fortement des informations sur la charge et la nature du trafic sur le réseau. Ces dernières doivent être à la fois, suffisantes, pertinentes et reflétant la charge réelle du réseau lors de la phase de prise de décision. Pour remédier aux inconvénients des techniques utilisant le QR, nous avons proposé deux algorithmes de RA. Le premier, appelé Q-Neural Routing, s'appuie sur un modèle neuronal stochastique pour estimer et mettre à jour les paramètres nécessaires au RA. Afin d'accélérer le temps de convergence, une deuxième approche est proposée : K-Shortest path Q-Routing. Elle est basée sur la technique de routage multi chemin combiné avec l'algorithme QR, l'espace d'exploration étant réduit aux k meilleurs chemins. Les deux algorithmes proposés sont validés et comparés aux approches traditionnelles en utilisant la plateforme de simulation OPNET, leur efficacité au niveau du RA est mise particulièrement en évidence. En effet, ceux-ci permettent une meilleure prise en compte de l'état du réseau contrairement aux approches classiques.

Mots clefs: Routage adaptatif, Qualité de service, Apprentissage par renforcement, Q-Learning, Q-Routing, Réseaux de neurones, Q-Neural Routing, K-Shortest path Q-Routing, k plus courts chemin.

Abstract

The aim of this thesis is to propose an algorithmic approach, which allows to treat the problems of adaptive routing (AR) in telecommunication networks with irregular traffic. The analysis of the existing approaches has lead us to base our work on the Q-Routing (QR) algorithm. This algorithm uses a reinforcement learning technique which is based on Markov models. The efficiency of these routing approaches depends on information about the network load and the nature of data flows. This information must be sufficient and relevant and it has to reflect the real network load during the decision making phase. To overcome drawbacks of techniques using QR, we have proposed two AR algorithms. The first one, which is called Q-Neural Routing, is based on a stochastic neural model, used for parameter estimation and updating required for routing. In order to reduce the convergence time, a second approach is proposed: k-Shortest path Q-Routing. It is based on a multi-paths routing technique combined with the QR algorithm. In this case, the exploration space is limited to k-Best paths. The proposed algorithms are validated and compared to traditional approaches using the OPNET Simulator. Their efficiency, with respect to AR, is illustrated. In fact, these algorithms allow taking into account the network state in a better way than the classical approaches do.

Keywords: Adaptive routing, Quality of service, Reinforcement Learning, Q-Learning, Q-Routing, Neural networks, Q-Neural Routing, K-Shortest path Q-Routing, K-Shortest paths.