Université Louis Pasteur de Strasbourg
Laboratoire des sciences de l'image, de l'informatique et de la télédétection

Routage multichemins par interface d'entrée

par

Pascal Mérindol

Thèse réalisée dans le Laboratoire des sciences de l'image, de l'informatique et de la télédétection
en vue de l'obtention du grade de Docteur
en Informatique

Université Louis Pasteur de Strasbourg
Laboratoire des sciences de l'image, de l'informatique et de la télédétection

Cette thèse intitulée

Routage multichemins par interface d'entrée

présentée par

Pascal Mérindol

a été évaluée par un jury composé des personnes suivantes :

Directeur de thèse	**Jean-Jacques Pansiot**
	Université Louis Pasteur - Strasbourg
Co-encadrant	**Stéphane Cateloin**
	Université Louis Pasteur - Strasbourg
Rapporteur Interne	**Thomas Noël**
	Université Louis Pasteur - Strasbourg
Rapporteurs Externes	**Olivier Bonaventure**
	Université catholique de Louvain
	Abdelmadjid Bouabdallah
	Université de Technologie de Compiègne
Examinateur	**Annie Gravey**
	TELECOM Bretagne - Brest

Remerciements

Je tiens avant tout à remercier toute personne qui lit ces lignes, et pour ceux qui ont plus de courage, l'intégralité de ce manuscrit. Parmis les lecteurs les plus méritants, je souhaite en premier lieu remercier mon directeur de thèse, Jean-Jacques Pansiot, et mon co-encadrant de thèse Stéphane Cateloin, dont les remarques et nombreuses contributions ont transformé ce modeste ouvrage depuis l'état de larve à celui de papillon. Leur implication dans mon travail de recherche ne se limite évidemment pas à la correction ortographique de mes publications mais s'est avéré très précieuse en bien des domaines.
La patience, la disponibilité et la faculté innée de Jean-Jacques Pansiot à supporter mes allées et venues inopinées dans son bureau ne seraient rien sans son écoute attentive et ses réponses avisées. De son coté, Stéphane Cateloin n'a pas non plus démérité, en tant qu'investigateur du projet multi-chemins, sa persévérance, sa motivation et ses qualités humaines m'ont été d'un grand secours au cours de ces quatres années de labeur. Un troisième relecteur d'un genre particulier, Julie Bocéréan, m'a également apporté moulte réconfort et su me guider sur la piste d'une orthographe plus saine...merci pour ton soutien au quotidien.
Bien entendu, je remercie mes parents, ma soeur et ma famille en général pour avoir cru en moi depuis l'époque bénie de la marche à quatre pattes.
Je profite de cet espace pour témoigner le plus grand respect à l'ensemble de mon équipe de recherche : Alex, Antoine, Benoit, Dominique, Guillaume, Julien, Romain, Sebastien, Vincent, Yana, et aussi à ceux, aujourd'hui disparus, qui m'ont permis d'apprécier l'informatique à sa juste valeur : Mickael Hoerdt et Emil Ivov.
Pour finir sur une note plus formelle, je remercie les membres de mon jury de thèse pour l'intérêt qu'ils ont porté à mon travail et simplement pour avoir accepté d'être rapporteurs, Messieurs Olivier Bonaventure, Abdelmadjid Bouabdallah et Thomas Noël, ou examinatrice, Madame Annie Gravey.

Résumé

La fiabilité d'un réseau IP face aux pannes et aux congestions dépend du temps de réaction associé au protocole de routage sous-jacent. Actuellement, les protocoles de routage à états des liens tels que OSPF ou IS-IS n'utilisent que les meilleures routes de coût égal pour commuter les paquets IP à l'échelle d'un domaine. La propriété de sous-optimalité des meilleures routes garantit la cohérence du routage au saut par saut bien que les chemins calculés via l'algorithme de Dijkstra soient composés de proche en proche. Selon la métrique employée, la diversité des chemins existant peut être largement sous exploitée avec une condition telle que la sous-optimalité. Or la diversité des alternatives de routage est l'un des éléments clés pour assurer un temps de réaction limité. La difficulté inhérente aux protocoles de routage multichemins saut par saut est la vérification de l'absence de boucles de routage. Chaque nœud doit garantir que le trafic qu'il achemine ne soit pas commuté sur un circuit dont il fait partie.
Dans ce rapport de thèse, après avoir mis en avant l'état de l'art existant dans la littérature, nous exposons deux contributions dont la combinaison assure cette propriété. La première proposition est basée sur l'algorithme de Dijkstra, il s'agit d'un algorithme de recherche opératoire nommé Dijkstra-Transverse qui calcule un ensemble de chemins transverses entre un nœud racine et chaque autre nœud du graphe modélisant le réseau. La seconde contribution est une procédure de validation distribuée dont le but est d'élaguer les circuits potentiellement générés par le routage saut par saut. Pour accroître la diversité des chemins validés, la procédure de commutation est spécifique à chaque interface entrante.
Par ailleurs, nous avons évalué l'impact de la diversité des chemins pour mettre en œuvre une couverture efficace en cas de panne de liens. La notion de couverture se décline en deux versions, locale ou globale, selon le type de protection envisagé, en d'autres termes, s'il est possible ou non de notifier les routeurs en amont de l'occurence d'une panne.
Nous nous sommes également intéressés aux aspects ingénierie de trafic liés à l'équilibrage de la charge en cas de congestion. Afin d'estimer l'importance de la diversité des chemins pour mettre en œuvre un routage proportionnel efficace, notre travail s'est focalisé sur la définition d'un module réactif de partage de charge. Celui-ci est simplement basé sur une analyse locale de la bande passante résiduelle et permet de mettre en relief les performances de nos propositions de routage par comparaison avec l'existant.
De manière générale, dans un souci de crédibilité, nos évaluations par simulation sont basés sur des topologies et une génération de trafic réalistes. Les résultats obtenus mettent en avant l'efficacité de nos algorithmes pour déployer un routage multichemins générant une diversité accrue par rapport à l'existant. Celle-ci est en effet nécessaire pour obtenir une capacité de commutation suffisante pour contourner les pannes et les congestions comme l'indiquent nos résultats liés aux deux types d'applications évalués.

Abstract

The reliability of IP networks in terms of failures and congestions depends on the reaction time associated with the underlying routing protocol. Currently, link state routing protocols such as OSPF or IS-IS use only the best paths to forward the IP packets at a domain scale. The sub-optimality property of best paths ensures consistency of hop by hop routing although the paths calculated using Dijkstra's algorithm are composed of close in close. According to the metric, the diversity of existing paths may be largely under estimated with a condition such as sub-optimality. Yet the diversity of alternatives paths is one of the key elements to ensure a limited reaction time. The main difficulty related to hop by hop multipath routing protocols is to ensure the absence of routing loops. Each node must verify that the traffic it carries is not switched on circuit where they belong.

In this PhD report, we present two contributions whose the combination ensures that property. The first proposition, based on Dijkstra's algorithm, is a multipath search algorithm called Dijkstra-Transverse (DT) which calculates a set of multiple paths between a root node and each other node in the graph modeling the network. The second contribution is a distributed validation procedure DT(p) whose the aim is to prune circuits potentially generated by hop by hop routing composition. To increase the diversity of validated paths, the forwarding mechanism is specific to each incoming interface.

Furthermore, we have evaluated the impact of the path diversity to produce an effective coverage if link failure occurs. The coverage can be defined in two versions, local or global, depending on the possibility to notify upstream routers of the detected failure.

We are also interested in traffic engineering issues related to load balancing in case of congestion. To estimate the importance of paths diversity to implement a efficient proportional routing, we have defined a reactive load balancing module. This module is based on a local analysis of residual bandwidth and highlight the performance of our proposed routing scheme. For the sake of credibility, our simulations are based on realistic topologies and traffic generation. The results underline the effectiveness of our algorithms to generate a greater diversity of paths compared to existing propositions. Paths diversity is necessary in order to obtain a sufficient forwarding capacity to circumvent outages and congestion as indicated by our results related to these two types of applications.

Table des matières

Introduction

Internet est très rapidement devenu un média protéiforme de premier plan. Les moteurs de recherches, les sites de partage de vidéos, la visiophonie et la télévision sur IP sont autant d'exemples qui modifient la perception de l'information et des distances. Les technologies de la communication basées sur IP se développent et proposent de plus en plus de services. L'internet moderne est un support majeur du partage de l'information. Ce succès croissant doit aller de pair avec des performances et une fiabilité accrues. Le nombre d'équipements et d'utilisateurs posent des problèmes d'extensibilité pouvant remettre en question la robustesse des architectures actuelles. De plus, la nature des services émergents, tels que les flux vidéos haute définition, provoque une hausse significative de la consommation des ressources. Bien que celles-ci, comme la puissance des processeurs ou la largeur de bande passante, soient elles aussi croissantes, il est nécessaire de développer de nouveaux paradigmes d'acheminement des paquets pour améliorer la robustesse des réseaux.

Les nouveaux services et besoins qui émergent de l'Internet sont très variés. Certains types d'application nécessitent une garantie en bande passante de bout en bout alors que d'autres types de demandes peuvent se contenter d'une qualité *au mieux* (*Best effort*). L'acheminement des flux devrait pouvoir garantir une qualité de service en fonction de leur nature. Pour parvenir à répondre justement à ces différents besoins, il est nécessaire d'agir à plusieurs niveaux. En effet, c'est aussi bien du ressort des protocoles de la couche transport que des mécanismes mis en œuvre au niveau de la couche réseau, de réguler le trafic en fonction du type de demande. Les techniques d'ordonnancemment intégrées aux files d'attentes permettent d'ajouter des mécanismes spécifiques à la qualité de service pour aiguiller les paquets. Ces différentes procédures doivent coopérer pour acheminer efficacement les données dans le réseau.

Le routage est au cœur de ce processus. Plusieurs approches permettent d'accélérer et plus généralement d'améliorer la réactivité du réseau. Le but est de mettre en place un routage dynamique s'adaptant aussi bien au cas moyen qu'aux cas critiques. Le processus de commutation doit permettre de réagir rapidement aux évolutions du trafic et de la topologie. Un autre objectif du multiroutage est de tirer au mieux parti de la diversité des chemins pour augmenter le flot maximal entre chaque paire de nœuds. Par ailleurs, la différenciation des services peut être mise en place au niveau du routage par l'estampillage des flux. Le routage proportionnel doit à la fois être compatible avec les mécanismes de retransmission de la couche transport tout en distribuant les flux associés à une classe de service donnée sur les chemins adéquats.

Dans ce travail de thèse, nous nous intéresserons aux problématiques liées au multiroutage au saut par saut. Plus particulièrement, nous nous focaliserons sur l'utilisation sous-jacente d'une diffusion de l'état des liens. L'activation de plusieurs routes entre une paire de nœuds donnée présente plusieurs avantages par rapport au routage monochemin qui n'utilise qu'une route de meilleur coût. L'usage de routes multiples peut être mise en œuvre, d'une part, pour des questions de robustesse, d'autre part pour des questions d'équilibrage de charge. Pour certaines applications sensibles telles que celles déployant de

la VoIP, la possibilité de détourner le flux de paquets sur une route alternative en cas de panne est primordiale. En effet, le temps de convergence d'un protocole de routage est généralement beaucoup trop élévé pour satisfaire le niveau de qualité de service de ce type d'applications. De manière générale, le routage multichemins permet d'augmenter les débits et de réduire les délais d'acheminement pour chaque couple de nœuds bénéficiant de routes multiples.
Ces objectifs sont généralement assurés via des mécanismes de routage pilotés par la source, ou via un dimensionnement préalable de la valuation des liens du réseau. Les réseaux d'opérateurs sont couramment surdimensionnés pour satisfaire des demandes supérieures au cas nominal. Le but de notre approche est de définir une réaction adéquate en cas de variations de charges importantes et imprévisibles. Notre proposition est une approche réactive ne nécessitant pas d'information a priori sur le trafic et dont la complexité de déploiement est inférieure à celle des méthodes de routage à la source. Cependant, avec un routage au saut par saut, la composition de proche en proche des chemins non optimaux peut provoquer la formation de boucles de routage. Cela implique la définition d'une condition suffisante pour élaguer la composition des chemins non optimaux provoquant des boucles. La réactivité du routage est alors conditionnée par la diversité des lignes de routage positionnées sur chaque routeur. La nature des indicateurs collectés (bande passante résiduelle, pertes de paquets, etc) et la périodicité des mesures influent également sur le niveau de réaction. La diversité des chemins est un enjeu majeur pour que la commutation bénéficie d'un espace de redirection accru. Le nombre de prochains sauts et la faible intersection des routes employées sont deux des facteurs contribuant à la qualité du routage. En termes de protection et de flot maximal, les routes disjointes présentent des caractéristiques avantageuses. La flexibilité des routes est un autre enjeu majeur : une fois aiguillés sur une route donnée, est-ce que les paquets sont susceptibles d'être à nouveau redirigés en cas de problème en aval ? Est-ce que cette redirection s'effectue uniquement par la source ou est-ce que la réaction est déterminée par le routeur le plus proche du problème ?

Dans le premier chapitre, nous analyserons, sous la forme d'un état de l'art, plusieurs politiques de routage multichemins de la littérature. Nous nous intéresserons au multiroutage au saut par saut, à la source et en interdomaine pour souligner, selon la perpective choisie, leurs spécificités respectives en termes de diversité des chemins générés. Nous introduirons les principales notions liées à la connaissance de la topologie, à l'algorithmique de calcul des chemins et à la commutation.
Dans le second chapitre, nous proposerons plusieurs contributions pour mettre en œuvre une politique de multiroutage générant une grande diversité de chemins. Le but de nos propositions est la construction distribuée d'une table de routage présentant une caractéristique particulière : la distinction faite sur l'interface d'entrée du trafic. Notre proposition comporte deux étapes : un algorithme de calcul de multichemins et une procédure de validation en profondeur. La composition de ces deux processus permet de garantir l'absence de boucles de routage tout en activant un nombre d'alternatives de routage élevé.
Le troisième chapitre introduit les problématiques liées à la reprise sur panne et à l'équilibrage de charge. Nous y présenterons les principales catégories de protocoles proposées dans la littérature. Notre

intérêt se portera tout particulièrement sur le paradigme de routage le plus répandu : une commutation par destination avec un routage à état des liens saut par saut. Les techniques de reroutage rapide sur IP et les méthodes d'équilibrage de charge distribuées seront analysées en détails. Par ailleurs, nous exposerons nos contributions, sous la forme de formalisme pour la mesure de la couverture, et en proposant un algorithme incrémentale pour le partage de charge.
Le chapitre 4 présentera les outils d'évaluation utilisés durant ces travaux de thèse. La méthodologie de simulation, et plus spécifiquement les topologies de réseau et les modèles de trafic y seront décrits. Pour finir, nous étudierons plusieurs séries de résultats obtenues par simulation, soulignant l'importance de la diversité des chemins et les améliorations apportées par nos contributions.
Un glossaire récapitulant les abréviations utilisées et les principales notations que nous avons définies est disponible à la fin de ce rapport de thèse.

Chapitre 1

Etat de l'art : Routage et diversité des chemins

Le routage et l'ensemble des mécanismes déployés au niveau de la couche réseau forment l'un des principaux facteurs générant de la qualité de service sur l'Internet. Les protocoles de routage actuellement déployés ne bénéficient pas toujours d'une diversité de cheminement suffisante pour lier une même paire de nœud via plusieurs routes. Cette diversité se caractérise par le nombre de routes et le flot maximal disponible entre deux routeurs.

Dans ce chapitre consacré aux différentes formes de routage multichemins sur réseau filaire, nous présenterons une synthèse de plusieurs modèles de commutation pouvant être mis en œuvre au niveau de la couche réseau pour proposer une diversité de chemins satisfaisante. Cet état de l'art se décompose en trois parties correspondant au principaux types de multiroutage unicast existants. Nous nous focaliserons principalement sur les aspects topologiques de la construction des multiroutes. Bien que profondément liées, l'état de l'art sur l'équilibrage de charge et la fiabilité d'un réseau IP sera approfondi dans le troisième chapitre.

Les auteurs de (AFT07) présentent une évaluation obtenue par trace du nombre de points de partage de charge traversés par couple (source, destination) à l'échelle d'Internet. Un premier constat est la fréquence du multiroutage : le routage multichemins par couple (source, destination) semble être couramment déployé alors que le modèle traditionnel du chemin unique, via un routage monochemin, s'applique encore à une vision par flux. Le second constat est la différence de diversité entre les chemins générés par le routage inter et intradomaine : le nombre de point de partage, dont bénéficie une paire de nœuds, obtenu grâce au routage en intradomaine est nettement plus importante que celle générée par le routage interdomaine. Malgré cette analyse encourageante, la topologie de l'Internet, que ce soit à l'échelle micro ou macroscopique, dispose intrinsèquement d'une diversité de chemins encore inexploitée.

Nous nous intéresserons à ces deux perspectives d'échelle en étudiant aussi bien leurs différences majeures en terme d'implémentation que leurs interactions pour favoriser l'inter-operabilité de leurs déploiements respectifs.

Ce chapitre décrira plus particulièrement les mécanismes de commutation associés aux politiques de routage suivantes :

- routage saut par saut en intradomaine.
- routage par la source en intradomaine.
- routage interdomaine.

Dans chaque partie nous commencerons par introduire les notions relatives à chaque politique de routage dans un cadre monochemin. Nous étudierons ensuite en détails les caractéristiques nécessaires à l'activation de plusieurs chemins entre chaque paire de nœuds. Dans le cadre du routage au saut par saut, nous insisterons notamment sur les contraintes inhérentes au déploiement de proche en proche de

chemins multiples de coûts inégaux.

1.1 Principes de base du routage

1.1.1 Présentation

Le routage est un ensemble de mécanismes permettant à chaque paquet en transit d'être correctement acheminé pour atteindre sa destination. On distingue trois étapes successives constituant le paradigme du routage.
La première étape consiste à transmettre un ensemble d'informations topologiques pour stabiliser la vision partagée du réseau. Dès lors, chaque routeur disposant d'une base de donnée topologique peut construire sa table de routage. Le calcul et la sélection des chemins permettant de construire une telle table sont réalisés en fonction de l'aire de routage considérée. La dernière étape consiste à utiliser la table de routage : il s'agit de la commutation des paquets.
La phase de calcul/sélection des chemins et les mécanismes permettant de mettre en place la commutation associée seront analysés en détails. Nous étudierons l'ensemble des problématiques liées à la contruction d'une table de routage. Les protocoles de routage basés sur la diffusion de vecteurs d'informations nous permettront dévoquer les différences fondamentales en termes de modélisation topologique et de calcul opérationnel provenant de la combinaison des deux premières étapes. Ces protocoles sont en effet basés sur un calcul des chemins distribué. En revanche, nous ne traiterons pas le cas du routage à la demande sur des circuits de type *ATM* où les chemins sont positionnés à la volée depuis la source désirant émettre.

1.1.2 Topologie et calcul des chemins

La phase de calcul des chemins utilise des algorithmes de recherche opérationnelle de la théorie des graphes. Ces algorithmes peuvent être distribués, c'est le cas des protocoles de routage utilisant des vecteurs informatifs, ou se baser sur une connaissance complète du graphe obtenue par inondation de messages topologiques. A chaque réception d'une annonce ou d'un vecteur, l'objectif du nœud récepteur est de déterminer un (ou plusieurs) chemin(s) vers chaque nœud destination d'un graphe G modélisant le réseau[1]. Pour les protocoles saut par saut, il s'agit de choisir un prochain saut pour la commutation, la notion de chemin est implicite.
L'algorithme de recherche opérationnelle correspondant à la phase de calcul des chemins est inhérent à toute forme de routage. La différence de traitement dépend de la vision du graphe G et réside dans le fait que le calcul peut éventuellement être distribué, c'est-à-dire basé sur la diffusion de vecteurs : les nœuds ne disposent pas de l'ensemble de la topologie mais seulement de la vision des nœuds voisins. La phase de calcul peut également être réalisée à la source au sens où le calcul est opéré sur la base d'une connaissance complète de G depuis celle-ci. Chaque nœud du graphe est une racine, et la phase de calcul s'effectue consécutivement à la réception d'une annonce topologique.

[1] Les définitions relatives à la théorie des graphes sont données dans la table 1.1.

Lorsque l'algorithme est distribué, la complexité réside essentiellement dans le contenu et le mode de distribution des messages, alors que, par définition, l'algorithme de recherche est plus complexe lorsqu'il est réalisé entièrement sur chaque nœud. Notons que le routage IP utilise généralement une commutation au saut par saut bien que le calcul des chemins puisse être réalisé à la source. C'est par exemple le cas avec le routage à état des liens ou avec un routage par la source.
Lorsque le routage n'utilise que les chemins de meilleurs coûts, le principe de sous-optimalité des meilleurs chemins suffit à garantir la validité de la commutation de proche en proche. En revanche, avec un routage multichemins au saut par saut, si les chemins calculés sont sous-optimaux, il est nécessaire d'utiliser une procédure de *validation*. Il s'agit de déterminer un sous-ensemble de chemins, parmi l'ensemble calculé à la réception d'une annonce ou obtenue grâce à la diffusion de vecteurs, produisant un routage *cohérent*.
En effet, les prochains sauts sont composés de proche en proche et le routage n'est donc pas explicitement défini de bout en bout lorsque les prochains sauts sont multiples. Le terme cohérent signifie que la commutation n'autorise pas la formation de **boucles de routage** et que les paquets soient effectivement acheminés jusqu'à la destination. En pratique, il s'agit d'éviter qu'un paquet puisse traverser à plusieurs reprises une même entité d'aiguillage. La notion de boucle de routage est formellement définie dans la partie 1.2.3.
Dans le contexte d'un routage saut par saut sur chemins optimaux, ou dans le cas d'un routage par la source, le calcul et la validation des chemins sont des processus confondus : l'acheminement est par définition cohérent pour la commutation. Un processus de sélection peut néanmoins être intérêssant pour satisfaire des critères de qualité de service.

1.1.3 Commutation

Une fois les chemins calculés, et si besoin, validés et sélectionnés par les algorithmes appropriés, plusieurs méthodes sont envisageables pour les utiliser. La commutation est un simple mécanisme de correspondance entre une information incluse dans le paquet IP et l'interface de sortie qui lui est associée dans la table de routage. Ce champ peut désigner une destination, un label ou directement le prochain saut. Lorsque la commutation se fait directement sur la destination (en pratique une adresse IP), il s'agit de déterminer quelle entrée de la table de routage lui correspond. La commutation par correspondance sur la destination est la plus utilisée pour le routage IP au saut par saut. Le routage par la source utilise une correspondance directe pour extraire l'interface de sortie au moyen d'étiquette (sous la forme de *labels*, voir 1.3.1) ou bien en utilisant directement l'information du prochain saut à effectuer : dans ce cas l'information est nécessairement contenue dans l'en-tête du paquet.
La commutation peut être mise en place de deux manières :

- de proche en proche : la décision de routage est déterminée au saut par saut.
- par la source : la décision de routage est conditionnée par le routeur qui calcule les chemins.

La commutation est par nature un mécanisme réalisé au saut par saut. Cependant, les tables de routage permettant son application peuvent être positionnées par le routeur lui-même ou par un autre routeur : la source du calcul. C'est le cas par exemple avec un routage explicite par commutation d'étiquette.

Notations	Définitions
$G(N, E, w)$	Graphe G orienté avec un ensemble de nœuds N, un ensemble d'arcs E et une valuation des arcs w strictement positive.
$\|N\|$,$\|E\|$	Cardinaux respectifs des ensembles N et E.
$e = e.x, e.y$ $e^{-1} = e.y, e.x$	Arc $e \in E$ reliant le nœud x au nœud y, e^{-1} désigne l'arc d'orientation opposée.
$k^-(x)$, $k^+(x)$	Degrés entrant et sortant du nœud x.
$succ(x)$	Ensemble des voisins sortants de x $y \in succ(x)$ s'il existe un arc $e = e.x, e.y \in E$.
$pred(x)$	Ensemble des voisins entrant de x $y \in succ(x)$ s'il existe un arc $e = e.y, e.x \in E$.
$c(e)$	Capacité du lien correspondant à l'arc e.
$d(e)$	Délai de propagation du lien correspondant à l'arc e.
$w(e)$	Valuation de l'arc e (généralement dépendante de $c(e)$ et/ou $d(e)$).
$\{C(s, d)\}$ $\{P(s, d)\}$	$\{C(s, d)\}$ correspond à l'ensemble des coûts calculés selon la valuation des arcs entre une racine s et une destination d. $\{P(s, d)\}$ désigne l'ensemble des chemins associés à ces coûts.
$P_1(s, d)$	Meilleur chemin $(e_1, ..., e_m)_{0<m<\|N\|}$ de coût $C_1(s, d)$ reliant la racine s à la destination d.
$C_1(s, d)$	Coût du meilleur chemin entre s et d. $C_1(s, d) = min(\{C(s, d)\})$
$NH_1(s, d)$	Meilleur prochain saut (*next hop*). $NH_1(s, d)$ est l'extrémité sortante du premier arc $\{s, e_1.y\}$ du meilleur chemin $P_1(s, d)$.

TAB. 1.1 – Notations génériques pour la théorie des graphes

La partie 1.2 présente les principaux protocoles de routage au saut par saut, les algorithmes de calcul de chemins et les processus de validation associés. La partie 1.3 présente le contexte technologique de la commutation par étiquette et les algorithmes de calcul des chemins de la littérature spécifiquement associées au routage à la source.

1.1.4 Contexte

Les différentes technologies que nous présenterons dans ces deux parties sont liées à des problématiques de routage intradomaine (*Interior Gateway protocol*, IGP). La dernière partie 1.4 introduit les concepts hiérarchiques liés au routage interdomaine. Nous illustrerons ces notions avec l'exemple de *Border Gateway Protocol* (BGP) et présenterons plusieurs propositions permettant de mettre en œuvre un routage multichemins entre domaines.
Le cas du routage centralisé ne sera pas traité en profondeur dans ce travail de thèse. Celui-ci est caractérisé par le fait que les décisions de routage sont entièrement calculées par une seule entité centrale quelle que soit la source et la destination du trafic. Le déploiement de ce type de méthode est soumis à des problèmes évidents de résistance aux pannes ainsi qu'aux problèmes de passage à l'échelle si le réseau est relativement grand. Le modèle centralisé peut en revanche s'avérer extrêmemment intérêssant pour les problématiques liées au partage de charge et à l'ingénierie de trafic. L'entité centrale distribuant les décisions de routage peut disposer en théorie d'une vision globale du trafic : l'ensemble des demandes entre chaque couple de routeurs et les ressources résiduelles du réseau.
Dans cet état de l'art, nous nous intéresserons plus particulièrement au routage intradomaine que nous étudierons de manière informelle jusqu'à la partie 1.2.3. Ce paragraphe introduit les méthodes de la littérature relatives au multiroutage saut par saut. Pour chaque type de multiroutage, nous nous focaliserons d'une part sur la phase de calcul des chemins et sur les mécanismes de diffusion d'informations topologiques associés, d'autre part sur les processus de validation.

1.1.5 Dynamicité

En cas de changements topologiques imprévisibles (panne de lien ou de routeur) le réseau doit pouvoir s'auto-configurer par lui-même. Pour cela, il est nécessaire d'employer un routage dynamique. Le routage dynamique peut se contenter de réagir aux changements topologiques ou alors son comportement peut également être influencé par les variations de charge. Par exemple, afin d'assurer une qualité de service temps-réel, il est nécessaire de prendre en compte les congestions. Nous désignerons ce type de routage dynamique par le qualificatif *adaptatif*. Le routage adaptatif est sensible aux fluctuations des disponibilités résiduelles des ressources considérées (bande passante, puissance de calcul des routeurs, etc), et peut se décliner sous deux formes : routage adaptatif monochemin et routage multichemins proportionnel. Ces deux modes de routage possèdent des propriétés de réactivité divergentes. Dans les deux cas, la stabilité de ce type d'approches est à contrôler. Nous reviendrons en détails sur ces différentes caractéristiques à partir du paragraphe 1.2.2.3.
Les deux premiers exemples de routage dynamique que nous allons considèrer ne prennent en compte que les événements liés à des changements topologiques.

1.2 Routage IP classique au saut par saut

Nous commencerons par étudier les protocoles de routage intradomaine communément déployés sur des réseaux IP lorsque la mise en place du processus de commutation est distribuée. Les décisions de routage sont prises par chaque routeur et seuls les chemins optimaux peuvent lier une paire de nœud [2]. Ainsi, calcul, sélection et validation des chemins sont des notions confondues car seul les chemins optimaux définis selon la métrique considérée sont utilisés pour la commutation. Le terme métrique est formellement introduit dans la partie 1.2.2.1.
Une propriété déterminante pour le routage au saut par saut, que l'on peut introduire à partir du moment où la commutation utilise uniquement des chemins de meilleurs coût pour une métrique donnée, est le principe de sous-optimalité des meilleurs chemins.

Definition 1 (**Principe de sous-optimalité**). *Soit un meilleur chemin* $P_1(s,d)=$ $(e_1,\cdots,e_i,\cdots,ej,\cdots,e_m)$, *alors* $\forall 1 \leq i < j \leq m$ *le chemin constitué des arcs* $e_k \in P_1(s,d),\ i \leq k < j$ *est un meilleur chemin* $P_1(e_i.x, e_j.x)$.

La preuve par l'absurde est triviale : si cette propriété n'est pas satisfaite, $P_1(s,d)$ n'est pas un meilleur chemin de s vers d. On notera que l'unicité du meilleur chemin peut être forcée en utilisant un ordre lexicographique.

1.2.1 Routage sur le(s) meilleur(s) chemin(s)

1.2.1.1 Routage à vecteurs de distance

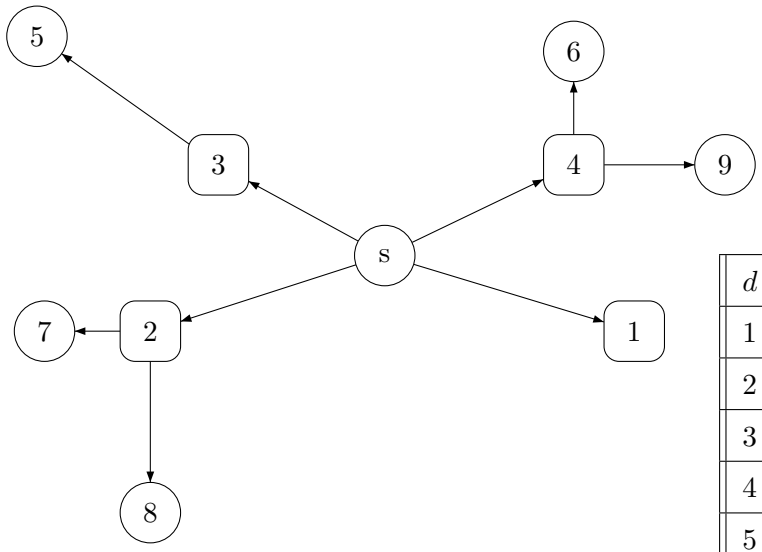
Les protocoles de routage à vecteurs de distance sont entièrement distribués. Leur phase de calcul est basée sur l'algorithme des plus courts chemins de Bellman-Ford (Bel58). Cet algorithme utilise une valuation non nécessairement positive des arcs et s'exécute en $O(|N| \times |E|)$ opérations. En pratique, la valuation des arcs peut être négative s'il n'existe pas de cycles de coût négatif. Dans le cadre de la commutation sur des réseaux IP, la valuation des arcs est strictement positive.
L'algorithme de Bellman Ford est utilisé de manière distribuée, c'est-à-dire que chaque nœud s effectue $k^-(s) \times (|N|-1)$ opérations. Le principe de ce type de routage est caractérisé par la diffusion de vecteurs de distances entre routeurs adjacents. Chaque routeur détermine à la réception d'un vecteur *(destination, coût)* le prochain saut à utiliser pour la commutation, c'est-à-dire le routeur adjacent proposant la route de meilleur coût. Les routeurs ne disposent que d'une connaissance partielle du réseau, ils possèdent uniquement des informations relatives au voisinage à un saut. La cohérence de la commutation est assuré par la diffusion de proche en proche des vecteurs de distance.
Considérons un routeur s possédant une entrée (d, v', c') dans sa table de routage. Si le couple *(destination, coût)*, (d,c), contenu dans le vecteur de distance reçu via un routeur voisin v, vérifie $c+w(s,v) < c'$ pour une même destination d, alors s remplace son entrée (d, v', c') par (d, v, c). L'unicité du meilleur chemin est déterminée par l'ordre de réception des vecteurs.

[2] La symétrie des chemins de coût optimal dépend de l'ordre lexicographique établi et de la symétrie des poids.

En pratique, les implémentations les plus connues du routage à vecteur de distance sont RIP, *Routing Information Protocol* (Hed88), et le protocole de routage propriétaire *Cisco IGRP*, *Interior Gateway Routing Protocol* (Hed91). Le protocole RIP dans sa version de base est destiné aux réseaux dont le diamètre est inférieur à 16. Il utilise une simple métrique additive sur le nombre de sauts, c'est-à-dire une distance en nombre de liens. Le principal problème induit par un routage de cette nature est le risque de comptage à l'infini (jusqu'à 16 en pratique) afin de converger vers un état stable. Bien qu'il existe des solutions simples comme le retour empoisonné ou utilisant des timers spécifiques, il se peut que des boucles de routage transitoires apparaîssent avant la stabilisation des routes (certains mécanismes plus spécifiques peuvent éviter ce problème, voir (SKS01) par exemple).

1.2.1.2 Routage à états des liens

Le routage à états des liens se base sur une connaissance complète de la topologie du domaine de routage. Si le domaine est décomposé en plusieurs aires, cette connaissance est partielle sur l'ensemble mais reste exhaustive sur l'aire considérée. L'algorithme sous-jacent est celui de Dijkstra (Dij59). Celui-ci nécessite la modélisation de l'aire considéré sous la forme d'un graphe. L'algorithme de Dijkstra s'exécute en $O(|N|^2)$ opérations et la valuation des arcs doit être positive.
Chaque routeur possède une base de données topologique. Cette base permet de calculer la *Routing Information Base* (RIB). La figure 1.1 illustre schématiquement la transition de l'arbre des plus courts chemins calculé avec l'algorithme de Dijkstra vers la table de commutation utilisée pour la commutation des paquets. Le terme SPT (*Shortest Path Tree*) désigne l'arbre des plus courts chemins.
Un nœud s, la racine du calcul, dispose grâce à l'algorithme de Dijkstra de l'ensemble des chemins vers chaque nœud du graphe. Cependant, seule l'information du prochain saut est utile pour la commutation.
Sur cet exemple, s dispose de quatre prochains sauts possibles (*1*, *2*, *3* et *4*) pour un ensemble de neuf destinations. Le routeur s utilise par exemple le prochain saut *4* pour commuter le trafic à destination de *4*, *6* et *9*. La base de données topologique est obtenue via la diffusion des annonces de l'état des liens (*Link State Advertisements*, LSA) par le biais d'un protocole d'*inondation*. Les LSA possèdent des numéros de séquence qui sont enregistrés à la réception et associés au routeur émetteur permettant leur identification unique. Lors de la réception d'un LSA, un routeur met à jour sa base de données si le numéro de séquence qu'il reçoit, est strictement supérieur au dernier reçu pour un émetteur donné. Le récepteur utilise ensuite l'algorithme de Dijkstra pour calculer l'ensemble des meilleurs chemins qui le relient aux autres nœuds du graphe.
Une fois cette opération effectuée, celui-ci peut mettre à jour sa table de routage (*Forwarding Information Database* : FIB) pour commuter les paquets. Le principe de sous-optimalité des meilleurs chemins garantit la cohérence de la composition des tables de routage.
Les protocoles à états des liens les plus répandus sont OSPF (Moy98), IS-IS (Ora01) et EIGRP (Pep99). Le calcul des meilleurs chemins est effectué avec l'algorithme de Dijkstra, et la métrique additive utilisée dépend de la valuation des liens. Par défaut, le coût d'un chemin peut être simplement caractérisé par le nombre de sauts. En général, elle prend en compte les capacités et/ou les délais de propagations des



d	$NH_1(s,d)$	$C_1(s,d)$
1	1	1
2	2	1
3	3	1
4	4	1
5	3	2
6	4	2
7	2	2
8	2	2
9	4	2

FIG. 1.1 – Arbre des plus courts chemins (SPT) enraciné sur le nœud s

liens. Ces notions seront détaillées une fois la terminologie définie dans le paragraphe 1.2.2.1.
Les protocoles OSPF et IS-IS implémentent une extension permettant de mettre en œuvre un routage multichemins. Cette extension se nomme *Equal Cost Multi-Path* (ECMP) et présente l'avantage de profiter du principe de sous-optimalité des meilleurs chemins grâce à l'activation des meilleurs chemins de coût égaux entre chaque paire de nœuds. ECMP n'est pas dépendant d'opérations complexes supplémentaires : le processus de validation des prochains sauts est intégré à la phase de calcul. L'algorithme de Dijkstra doit être légèrement modifié pour mémoriser les alternatives présentant un coût égal pour une même destination. Plus précisement, il s'agit de stocker à chaque itération les multiples nœuds prédecesseurs proposant une alternative de même coût pour la destination considérée. Cette information supplémentaire doit être transmise depuis le nœud prédecesseur vers son successeur afin de considérer les chemins de même coût non disjoints. Le temps de calcul du SPT est à peine accru, la complexité étant surtout augmentée en espace. La politique de partage de charge entre chemins de coût égaux est l'équité. Cependant, il est nécessaire de prendre en compte les spécificités du trafic TCP pour réaliser un partage par flux comme nous le détaillerons dans la partie 3.2.3.

1.2.2 Routage IP et qualité de service

1.2.2.1 Qualité de service et métrique

La qualité de service (*Quality of Service*, QoS) au niveau routage désigne une partie des techniques de la couche réseau permettant de garantir un ensemble de contraintes en termes de performances. Il peut s'agir de contraintes concernant l'ensemble ou une sous-partie des communications transitant sur le réseau, par exemple pour diminuer la latence entre les paires communiquantes. Au niveau de la couche réseau, il existe d'autres mécanimes, tels que ceux déployés pour l'ordonnancement des files d'attente, pour mettre en œuvre de la QoS.
La qualité de service est une notion générique dont la mise en œuvre peut prendre plusieurs formes. Nous distinguerons notamment les techniques à différenciation de services des méthodes dont l'objectif est une qualité de service à l'échelle de l'ensemble des flux. Dans le premier cas, la nature du flux (tranfert de fichier, vidéos en streaming, visioconférence, etc) est utilisée pour déterminer le résultat de la fonction de commutation alors que dans le second cas, la commutation est sans distinction sur la classe des flux.
Les protocoles de routage par réservation tel que *Intserv* (pour l'intégration de service) avec un contrôle d'admission *RSVP* (*Resource ReSerVation Protocol*, (BZBH97)) ou des mécanismes comme *Diffserv* illustrent précisement le cas d'une QoS distinguant a priori le type d'application communiquante. Le protocole *RSVP* permet d'assurer une garantie de bande passante (de délais ou d'autres contraintes) de bout en bout. Il s'agit de réserver en mode *soft-state* une certaine quantité de bande passante sur chaque lien constituant le chemin reliant les deux entités de communication (notons les s et d). La signalisation sous-jacente à ce mécanisme fonctionne sur une phase de découverte et vérification à l'aller (message *Path*) alors que la réservation se fait au retour (message *Resv*).
Si s souhaite s'assurer de disposer d'une bande passante minimale jusqu'à d alors il doit initier une demande de réservation jusqu'à d. Le message Path contient un champ *flowspec* décrivant la nature du ou des flux de données émis par s. Il s'agit d'envoyer un paquet vérifiant les capacités résiduelles de chaque lien traversé jusqu'à d. Lorsque la commutation sous-jacente dispose d'interfaces de sortie multiples vers d alors ce paquet est transmis sur chacun des prochains sauts possibles. Si le domaine utilise un routage multichemins, on peut imaginer qu'une fois le(s) message(s) *Path* reçu(s) par le routeur d, celui-ci puisse vérifier la faisabilité de la demande et réserver en *source routing* un des chemins contenus dans l'ensemble des possibilités satisfaisant les contraintes de routage déterminés par s. De même que pour le routage monochemin en y ajoutant un mécanisme de sélection, il faut vérifier que chacun des liens dispose d'une bande passante supérieure à celle réclamée par s.
Les principaux défauts de cette technique se caractérisent par son manque d'extensibilité, même en agrégant les flux, et par la surréservation transitoire pouvant entraîner une dégradation du routage pour le trafic ne bénéficiant pas de ce type de mécanisme.

L'architecture *Diffserv* (voir les RFCs 2474 (NBBB98) et 2475 (BBC$^+$98)) permet de classifier les flux selon leur type de service. Chaque paquet IP est marqué par un champ (*Diffserv Code Point*)

désignant le traitement QoS à lui attribuer. La marquage se fait à l'entrée d'un domaine *Diffserv* et permet, à l'intérieur de ce domaine, de déterminer un traitement de commutation spécifique. Tous les paquets appartenant à une même classe profiteront d'un traitement équivalent, il peut s'agir par exemple de files d'attente plus ou moins prioritaires. Il existe trois types de classes génériques : *Expedited Forwarding*, *Assured Forwarding* ou *Best Effort* pour les paquets non marqués.
La différenciation de services n'est pas le seul moyen de générer de la qualité de service : le routage peut opter pour un traitement équitable. Dans le contexte d'une qualité de service généralisée, il s'agit typiquement de définir une métrique commune assurant un partage équitable et optimal des ressources. La notion de métrique générique est, à la différence de *Diffserv*, utilisée pour permettre à l'ensemble des flux et des applications associées de bénéficier des meilleures conditions de routage possibles. Selon la nature des besoins, plusieurs indicateurs peuvent être utilisés pour mesurer la qualité d'une route. Ces indicateurs peuvent être statiques ou dynamiques. Néanmoins pour éviter de trop fréquentes oscillations, les indicateurs statiques tel que la capacité d'un lien, le délai de propagation, la propabilité de perte (etc) sont plus largement répandus. En fonction de l'indicateur choisi, le coût d'un chemin est déterminé selon la métrique correspondante. En admettant que la valuation w d'un lien e_i de ce chemin reflète l'indication désirée, la métrique peut être (pour un chemin de m sauts) :

- additive : $C = \sum_{i=1}^{m} w(e_i)$, par exemple le délai de propagation.
- concave : $C = min(w(e_i))\ \forall 1 \leq i \leq m$, par exemple la capacité.
- multiplicative : $C = \prod_{i=1}^{m} w(e_i)$, par exemple la probabilité de non perte.

Typiquement, une métrique multiplicative sera utilisée conjointement à un(des) indicateur(s) de type probabiliste (en pratique $w(e_i)$ évaluera une probabilité de transmission avec succès), alors qu'une métrique additive a généralement pour objectif d'estimer un délai d'acheminement en utilisant des indicateurs tels que la capacité et/ou de les délais de propagation. Les métriques concaves sont destinées à vérifier une contrainte, par exemple en bande passante, pour s'assurer que les flux bénéficient d'une garantie de service. La valuation peut être elle-même une combinaison de plusieurs indicateurs, on parle alors de métrique composite. Dans ce cas, il s'agit généralement plus d'une heuristique de routage, car les contraintes ne peuvent plus être vérifiées indépendamment.
Gouda et Schneider définissent dans (GS03) la notion de métrique maximisable en fonction de deux caractéritisques : la monotonicité et la majorabilité. Ils prouvent que ces deux caractéristiques sont nécessaires et suffisantes pour maximiser une métrique. Ils proposent une méthodologie de combinaison de métriques bornées et monotones pour définir des métriques composites maximisables. D'autres travaux de recherches (BBB01) présentent un algorithme paramétrable et distribué pour minimiser à la fois les délais et le coût des chemins. Leur fonction paramétrable permet, durant la construction des chemins, de définir un point d'équilibre entre coût et délai plutôt que de forcer la priorité de l'un des deux paramètres. Les chemins calculés vérifient une contrainte paramétrable en termes de délais.

1.2.2.2 Combinaison de métriques

Les contraintes usuelles sont la bande passante, le délai de propagation, le taux d'erreur ou la gigue. La bande passante d'un chemin est celle de son lien de plus petite capacité (métrique concave) et représente un critère prépondérant de la QoS.
Wang et Crowcroft (WC96) ont démontré, par réduction au problème de partition, que la satisfaction simultanée de deux contraintes, entre des métriques additives et/ou multiplicatives, est un problème NP-complet.
Néanmoins, si la source (la racine du calcul) connaît la topologie, il est possible de trouver une solution exacte en combinant une métrique concave et une métrique additive ou multiplicative. Par exemple, si l'on utilise une métrique concave sur la bande passante et une métrique additive sur les délais, il suffit de supprimer du graphe les arcs dont la bande passante est inférieure à la valeur seuil correspondant à la contrainte.
Les auteurs proposent une heuristique pour utiliser conjointement ces deux métriques dans un contexte de routage saut par saut. Il s'agit de résoudre de proche en proche un problème de type *min-max* avec la combinaison d'une métrique concave d'abord et additive ensuite. La métrique additive assure l'absence de boucle de routage. Leur proposition *Shortest Widest Path algorithm* (SWP) considère dans un premier temps l'ensemble des chemins proposant une bande passante maximale. SWP sélectionne les chemins dont le lien de capacité la plus faible dispose de la plus grande bande passante (maximisation avec une métrique concave). Si cet ensemble est constitué de plusieurs éléments, SWP choisit le chemin proposant le plus petit délai (minimisation avec une métrique additive). Leur algorithme se décline en deux versions distribuées : l'une correspond à une méthode avec vecteurs de distance et l'autre à une méthode à états de liens. Ces deux heuristiques ne garantissent pas l'obtention d'une solution exacte[3] contrairement à la méthode qui supprime du graphe les arcs ne satisfaisant pas la contrainte.
Dans les deux cas, la complexité et l'exécution de leurs heuristiques sont liés à l'algorithme sous-jacent (Bellman Ford distribué ou Dijkstra à la source).
La priorité peut être inversée sur les métriques employées pour définir un routage *Widest Shortest Path* (WSP).

1.2.2.3 Routage adaptatif

Le routage adaptatif utilise une métrique basée sur des caractéristiques dynamiques du réseau. Typiquement, la bande passante résiduelle des liens ou plus généralement une estimation *online* du délai de bout en bout peuvent être utilisées. Le protocole de diffusion de l'état des liens doit alors notifier régulièrement l'ensemble du réseau de ces changements. La diffusion d'annonces peut être déclenchée par le franchissement de seuils ou émis à intervalles fréquentiels (ou selon une méthode hybride) pour avertir le réseau des variations de charge.
La valuation d'un arc peut prendre la forme d'une estimation dynamique de la moyenne du temps de transmission d'un paquet sur le lien qu'il modélise. Toutes les t secondes, si un certain seuil critique est

[3] En théorie, cette solution est aussi applicable à un routage saut par saut à états des liens avec l'algorithme de Dijkstra.

dépassé, pour éviter de considérer les modifications mineures, la nouvelle valuation du lien l est diffusée à l'ensemble du réseau pour un nouveau calcul complet des meilleurs routes.
Il apparaît néanmoins complexe de définir un indicateur pertinent et peu sensible aux variations transitoires. Les effets d'oscillations peuvent se révéler très néfastes pour la qualité du routage. En effet, dans la mesure où l'ensemble des routeurs ont reçu les même indications, le basculement de charge est synchronisé. Cet effet présente des risques d'oscillations : à chaque intervalle de temps consécutif, la charge peut être basculée d'un chemin à l'autre.
Les travaux présentés dans (KZ89) sont un exemple de routage adaptatif potentiellement instable lorsque le réseau est globalement chargé. Chen (CDS99) a analysé une méthode incrémentale où les destinations initient le recalcul global des chemins. Les paquets acheminés vers les destinations non affectées ne subissent pas de changement de commutation. Wang et Crowcoft ont réalisé une analyse détaillée du comportement des algorithmes de routage adaptatif sur des chemins de coûts optimaux (WC92). Ils examinent le problème décisionnel lié au contrôle de charge : un comportement oscillatoire récurrent aboutit à une dégradation générale des performances. Les références (Pax97) et (LMJ98) analysent précisement l'impact négatif des oscillations de charge pour la qualité du routage. Par ailleurs, à une échelle microscopique, lorsqu'un flux est fréquemment dévié, cela dégrade le comportement des protocoles de transport *élastiques* comme TCP (voir le paragraphe 3.2.3).

Le routage proportionnel permet d'introduire un comportement plus stable. Dans le cas d'un routage adaptatif monochemin la réaction de déviation est de type *tout ou rien*. Le routage proportionnel multichemins autorise une déviation de charge potentiellement plus fine. Lorsqu'un routeur dispose d'un ensemble de prochains sauts multiples pour une même destination, la déviation des flux peut être progressive : la politique de commutation évolue au fur et à mesure des indications recueillies. Chaque chemin, ou prochain saut, est associé à une proportion de charge fonction des indicateurs perçus. La fréquence de rafraîchissement des indicateurs résiduels permet alors d'ajuster progressivement la variation des proportions de routage.
Dans la suite de ce chapitre nous nous nous focaliserons sur deux techniques de routage intradomaine, dont les déploiements se prêtent naturellement à l'utilisation de multichemins, mais dont les principes de commutation sont différents :

- La commutation relâchée au saut par saut (partie 1.2.3).
- La commutation explicitement positionnée par la source (partie 1.3).

Bien que le multiroutage à états des liens soit généralement un routage au saut par saut[4], la phase de calcul des multichemins présente des similitudes avec celle utilisée par les techniques de multiroutage à commutation d'étiquettes.
Néanmoins deux différences structurent leur déploiement respectif. D'une part, la phase de calcul des chemins n'a pas besoin de considérer les chemins dont le premier saut est identique si le routage est saut par saut. D'autre part, la méthodologie de commutation, au sens positionnement des FIB, est

[4]Notons que le routage à états des liens *OSPF-TE* peut utiliser *RSVP-TE* pour un établir un positionnement par la source.

différente :

- dans le cas où la commutation est explicitement déterminée par la source, le chemin est positionné de bout en bout.
- dans le cas d'une commutation calculée au saut par saut le chemin est virtuel : sur chaque routeur traversé, le paquet en transit peut être commuté sur un prochain saut différent de celui qui avait été calculé par la source.

Les protocoles multichemins pilotés explicitement par la source ne nécessitent pas de mécanismes de validation. La commutation successive des prochain sauts est conforme au chemin calculé par la source. Le routage au saut par saut utilise une commutation distribuée résultant de la composition de proche en proche des prochains sauts. Dans la suite, nous désignerons ce type de commutation par le terme de *routage relâché*. Ce terme signifie que la FIB est positionnée sur chaque routeur en fonction de ses propres calculs de multichemins. Un tel routage nécessite des contraintes spécifiques pour que la composition des prochains sauts soit cohérente.

1.2.3 Routage multichemins au saut par saut

1.2.3.1 Introduction et généralités

Le multiroutage au saut par saut peut être subdivisé en quatres tâches distinctes :

1- calculer et sélectionner (par validation) les multiples chemins de routage.
2- analyser les ressources résiduelles de chaque lien et en informer le voisinage.
3- distribuer la charge en fonction des informations de disponibilité perçues et/ou calculées.
4- répartir le trafic selon les proportions de distribution établies et selon sa nature.

Nous nous focaliserons dans cet état de l'art sur la première tâche. Les trois tâches suivantes seront abordées dans le chapitre **Fiabilité et équilibrage de charge**.

Notre interêt se portera tout particulièrement sur les aspects de validation topologique permettant de garantir la cohérence de la composition des prochains sauts. Cette étape est inhérente au routage saut par saut et permet de contrôler l'acheminement des paquets sur des chemins non optimaux. Un paquet ne doit pas indéfiniment, ou même seulement plus d'une fois, traverser une même série de routeurs : **Le but du processus de validation est de vérifier l'absence des boucles de routage afin de ne pas activer de route de coût infini**.

Cette partie est consacrée aux principaux protocoles de multiroutage au saut par saut de la littérature. Leur caractéristique commune est leur capacité à utiliser plusieurs chemins de coût inégaux entre une même paire de nœuds (s, d). Cette famille de protocoles est basée sur une commutation par destination et ne nécessite pas de phase de marquage des routes de bout en bout. Les avantages de ce type de technique sont la flexibilité distribuée de commutation, et une complexité de déploiement dépendant seulement du nombre de destinations internes au domaine. La principale difficulté est liée aux contraintes topologiques de validation pouvant restreindre la diversité des chemins.

Dans la suite, un chemin *primaire* désigne un chemin optimal entre une paire de nœuds, l'utilisation d'un ordre lexicographique permet de lever toute ambiguïté en cas d'égalité sur les coûts. Un chemin

alternatif correspond à tout autre chemin reliant cette même paire (à orientation identique). La notion de chemin est *symbolique* : seul le premier saut est explicite. En pratique, les termes primaire et alternatif désignent les prochains sauts associés à ces chemins (*next hop*, NH).
La flexibilité de commutation est induite par la nature relâchée du routage : tout routeur peut décider de lui-même de modifier ses choix de commutation vers chacune des destinations s'il possède une alternative locale (un prochain saut alternatif différent de celui du chemin primaire). Les changements de proportions peuvent être décidés localement sans nécessairement en notifier le voisinage ou un routeur d'entrée de domaine comme avec les mécanismes explicitement pilotés par la source. Le passage à l'échelle de ces méthodes dépend principalement du nombre de destinations considérées.
La difficulté inhérente à ces méthodes est le risque de boucle de routage : un paquet ne peut être acheminé sur le même prochain saut à plusieurs reprises. Pour prendre cet aspect en compte, il faut introduire les notations relatives aux flux et à leurs orientations. Un flux peut être défini comme un ensemble de paquets aux caractéristiques communes : <source, destination, ports, protocole de transport, etc>. Les notions de routeur *en amont* et *en aval* sont dépendantes de l'orientation du flux considéré (voir figure 1.2). Un routeur p, est en amont d'un routeur s s'il existe un chemin vers d sur lequel p précède s. Réciproquement, s est en aval de p pour la même destination. De proche en proche pour une destination donnée, cette notion est transitive, s est en amont des routeurs v_1 et v_2 si par exemple s utilise v_1 comme NH primaire et v_2 comme NH alternatif vers d.

Le tableau 1.2 fournit les notations liées aux calcul de chemins multiples. Celles-ci nous permettront de décrire uniformément l'ensemble des règles utilisées pour mettre en œuvre un routage cohérent.
Les chemins calculés sont ordonnés en considérant uniquement la meilleure alternative disponible via chaque arc sortant. Si deux chemins $(e_1, ..., e_m)$ et $(e'_1, ..., e'_{m'})$ appartenant à $P(s, d)$ partagent un premier saut commun $e_1.y = e'_1.y$, alors seul celui de meilleur coût est retenu.

Dans le contexte d'un routage saut par saut monochemin, le principe de sous-optimalité du meilleur chemin permet de garantir que le chemin calculé est celui utilisé de bout en bout. Dans le cas de multichemins optimaux, la garantie porte sur le coût. Cette condition est suffisante pour garantir l'absence de boucle de routage. Lorsque le routeur d'entrée marque explicitement les multiroutes jusqu'au routeur de sortie, la commutation de bout en bout ne souffre pas d'ambiguïté et le chemin calculé est nécessairement celui utilisé.
Considérons l'exemple d'un routage multichemins à états des liens. Chaque routeur s dispose poten-

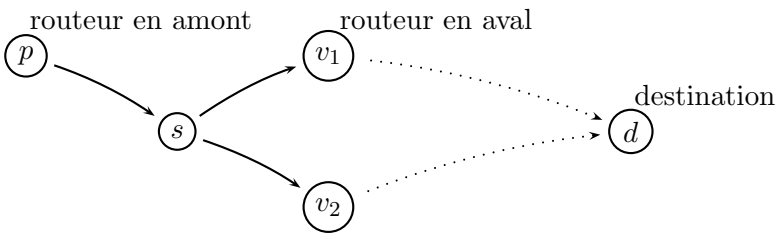


FIG. 1.2 – Validation des voisins

Notations	Définitions
$P_j(s,d) = (e_1, ..., e_m)$	j^{eme} meilleur chemins reliant s à d. Récursivement, il s'agit du meilleur chemin dont le premier arc est différent du premier arc de $P_l(s,d)$ pour tout l tel que $1 \leq l < j$.
$C_j(s,d) = \sum_{i=1}^{m} w(e_i)$	j^{eme} meilleur coût calculé par s vers d, c'est le coût du chemin $P_j(s,d)$ avec $(1 \leq j \leq k^+(s)), (0 < m < \|N\|)$.
$NH_j(s,d)$	j^{eme} meilleur prochain saut calculé par s vers d. Il s'agit du premier saut du chemin $P_j(s,d)$.
$NH(p,s,d)$	Ensemble des prochains sauts activés par le routeur s pour les paquets provenant du routeur p en entrée vers la destination d.
$(p,n,d)_s$ $n \in NH(p,s,d)$	Ligne de routage activé par le routeur s pour l'interface entrante p vers la destination d avec pour prochain saut le routeur n.
$F_s(y)$	Fonction *père*, $F_s(y) = x$ si $\{x,y\} \in P_1(s,y)$.
$f(x,d)$	Fonction *fils*, il s'agit du *fils* de x sur le chemin $P_1(x,d)$. $y = f(x,d)$ si $\{x,y\} \in P_1(x,d)$.
$primaire$	Un chemin $P_1(s,d)$ ou un prochain saut $NH_1(s,d)$.
$alternatif$	Un chemin $P_j(s,d)$ ou un prochain saut $NH_j(s,d)$ tel que $j > 1$.

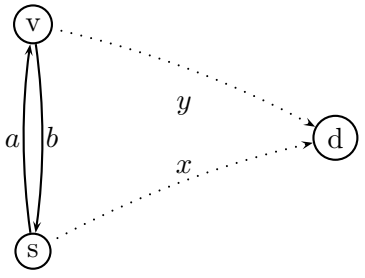
Tab. 1.2 – Notations spécifiques au routage multichemins saut par saut

tiellement de plusieurs alternatives vers une destination d donnée : des premiers sauts obtenus grâce à un algorithme de calcul e multichemins exécuté localement sur s. A ce stade, ces chemins ne contiennent pas de circuits. Admettons qu'un de ces chemins $P_j(s,d)$ utilise un routeur v comme premier saut, $NH_j(s,d) = v$.
Si v dispose d'un chemin passant par s, soit directement en tant que premier saut ou indirectement par composition, alors le routage peut devenir incohérent. Si s ne vérifie pas que les paquets, qu'il génère ou qu'il reçoit, ne passent pas par le chemin contenant v, la commutation induit une boucle. La figure 1.3 illustre ce phénomène avec une boucle de routage directe comportant deux sauts : si s utilise son voisin v pour atteindre d et vice-versa alors cela provoque la formation d'une boucle de routage sur un circuit comprenant deux arcs. Cet exemple peut être généralisé pour un circuit de n arcs si s et v appartiennent à un cycle de n sauts.
Pour formaliser la notion de boucle, il est nécessaire de définir les notions relatives à la composition des prochains sauts pour une destination donnée.

Definition 2 (**Graphe de composition**). *Un **graphe de composition** noté $G_d(N, E')$ pour une destination d désigne un sous graphe de $G(N, E)$. Les arcs $e \in E'$ constituent l'ensemble des premiers arcs utilisés par les chemins $P_j(s,d),\ \forall s \in N \wedge 1 \leq j \leq k^+(s)$.*

Definition 3 (**Graphe de composition élagué**). *L'ensemble E' peut être réduit à un sous-ensemble E'' si l'ensemble des nœuds $s \in N$ appliquent une condition R pour supprimer une partie des arcs $\{\{s, NH_j(s,d)\}\}$ tel que $j \in \{1, ..., k^+(s)\}$. On notera un tel graphe $G_d^R(N, E'')$.*

La condition R désigne la règle de validation utilisée pour l'élagage des circuits appartenant à $G_d(N, E')$. Si la condition R désigne la propriété de sous-optimalité, le graphe G_d^R est un SPT inversé vers d (*reverse SPT*). Cette condition permet de supprimer *statiquement* un sous-ensemble des arcs de $G_d(N, E')$ afin de construire un graphe acyclique orienté vers une destination donnée. Cependant, cette définition n'est pas assez précise pour représenter les règles de routage considérant l'origine des flux (que ce soit la source du flux ou l'interface entrante). Elle permet de décrire les règles basiques utilisant une commutation par correspondance directe sur la destination.



Dest	NH	Coût
D	$NH_1(s,d)$	x
D	v	$a+y$
D	$NH_1(v,d)$	y
D	s	$b+x$

FIG. 1.3 – Une boucle de routage de taille 2

Definition 4 (**Circuit et boucle de routage**). *Un **circuit** désigne un chemin dont les deux extrémités sont confondues. Un chemin* $(e_1, \cdots, e_i, \cdots, e_m)$ *sans circuit est un chemin tel que* $\forall 1 \leq i, k \leq m, e_i.x = e_k.x$ *si et seulement si* $i = k$. *Soit le graphe* G_d^R *généré par la composition de proche en proche des chemins sélectionnés par une condition* R *utilisée par chaque routeur vers une destination donnée* d. *S'il existe* $d \in N$ *tel que le graphe* G_d^R *comporte un chemin contenant un circuit, alors le processus de sélection* R *provoque la formation de boucle de routage au niveau routeur.*

Lorsque l'on considère l'interface d'entrée des flux, c'est-à-dire le routeur en amont, la notion de boucle de routage peut être affinée. Une boucle peut exister au niveau d'un routeur sans que les paquets ne traversent deux fois un même lien. Pour distinguer ces deux cas, il suffit de considérer qu'un chemin sans circuit ne contient pas deux arcs identiques. Dans ce cas, la définition 2 est insuffisante pour définir la prise en compte de l'interface entrante. Le paragraphe 2.2.2 fournira les notations adéquates.
Une boucle de routage peut se définir à deux niveaux :

- boucles de routage sur un routeur.
- boucles de routage sur un lien.

Si la condition de routage sans boucle au *niveau routeur* est vérifiée alors la condition d'absence de boucles au *niveau lien* est également vérifiée.

Definition 5 (**Boucle de routage au niveau routeur**). *Un routage sans boucle au niveau routeur vérifie : tout routeur* s *acheminant un paquet (générés localement par* s *ou provenant d'un routeur en amont* p*) via un voisin* v*, un routeur en aval, et vers une destination* d*, ne peut recevoir à nouveau ce paquet. Un chemin sans boucle de routage au niveau routeur est un chemin* $(e_1, \cdots, e_i, \cdots, e_m)$ *tel que* $\forall 1 \leq i, k \leq m, e_i.x = e_k.x$ *si et seulement si* $i = k$.

Definition 6 (**Boucle de routage au niveau lien**). *Un routage sans boucle au niveau routeur lien : tout routeur* s *acheminant un paquet (générés localement par* s *ou provenant d'un routeur en amont* p*) via un voisin* v *et vers une destination* d*, ne peut à nouveau utiliser* v *comme prochain saut si ce paquet transite à nouveau par lui.*

Le tableau 1.3 énumère une liste de conditions de validation. Chaque règle représente le processus de validation qui détermine l'ensemble $NH(p, s, d)$ pour un triplet : routeur en amont p, routeur courant s et destination d. ECMP correspond au critère de sous optimalité appliqué aux chemins de coûts égaux, LFI est un critère de stricte décroissance Ã un saut, et SPD relâche ce critère de stricte décroissance à deux sauts. De ces ensembles[5], on peut déduire une table de routage multichemins dont la spécificité est de permettre éventuellement, dans le cas de SPD, la distinction de l'interface émettrice : le routeur en amont.
Une ligne de routage $(p, n, d)_s$ se définit par un prochain saut $n = NH_j(s, d)$ appartenant à l'ensemble $NH(p, s, d)$. Si le mécanisme de commutation est le même quelle que soit l'interface entrante p, la table de routage de s contient au plus $|N| \times k^+(s)$ entrées et $NH(p, s, d) = NH(s, s, d)\ \forall p \in pred(s)$, $\forall d \in |N|$. Dans le cas contraire, il peut y avoir jusqu'à $|N| \times k^-(s) \times k^+(s)$ lignes de routage ($\forall p \in pred(s)$, $|N| \times k^+(s)$ entrées au maximum appartiennent à l'ensemble $NH(p, s, d)$).

[5]Les règles LFI et SPD sont décrites en détails dans la suite du document.

Condition sur v	Nom	Références
$C_j(s,d) = C_1(s,d) \wedge v = NH_j(s,d)$	ECMP	(Moy98), (Ora01)
$C_1(v,d) < C_1(s,d)$	LFI	(NS99), (Vut01), (YW06), (Vil99b)
$C_1(v,d) < C_1(p,d)$	SPD	(YW06)

TAB. 1.3 – Règles de validation pour le multiroutage saut par saut sans boucles

Plusieurs mécanismes peuvent être envisagés pour le calcul des chemins et la validation des NHs. Le processus de validation peut être local et intégré dans l'algorithme de calcul des multichemins. Celui-ci peut également prendre la forme d'un protocole distribué permettant de transmettre les résultats issus des calculs des chemins locaux. Dans le premier cas, la complexité de la phase de calcul est liée aux contraintes locales de validation. Dans le second cas, il est nécessaire de diffuser des messages à caractère topologique et de mettre en place un mécanisme de stabilisation.
L'objectif est de déterminer le meilleur coût des voisins vers chaque destination, par exemple par le biais du calcul du SPT des routeurs adjacents. Le processus de validation peut être *local* ou *distribué*. Les paragraphes qui suivent décrivent les méthodes existantes de la littérature avec leur algorithme de calcul de chemins et le processus de validation utilisé.
La première condition de ce tableau correspond à l'utilisation de meilleurs chemins de coût égaux. Il s'agit de stocker, durant l'exécution de l'algorithme de calcul de chemins sous-jacent (Dijkstra, Bellman Ford, etc), les prochains sauts permettant d'atteindre la destination considérée avec un chemin de même coût que le chemin primaire. Cette condition utilisée par l'extension ECMP des protocoles IS-IS et OSPF (Ora01 ; Moy98) génére une diversité des chemins relativement importante lorsque la valuation est uniforme (la métrique correspond aux nombre de sauts). Cependant, pour des métriques plus sophistiquées, par exemple liées aux capacités et aux délais de propagation, cette caractéristique topologique est nettement moins fréquente.
Exemple de validation n^o 1
La topologie présentée dans la figure 1.4(a) sert d'illustration de référence pour la compréhension des différentes conditions de validation. Nous nous en servirons aussi pour illustrer les conditions utilisées pour le reroutage rapide (voir le paragraphe 1.2.4).
Le coût de chaque arc est donné en 1.4(a) et l'arbre des meilleurs chemins enraciné en $s = 4$ est représenté en 1.4(b). L'extension ECMP des protocoles OSPF ou IS-IS permettrait par exemple au routeur numéroté 1 d'utiliser simultanément les chemins $P_1(1,3) = (\{1,2\},\{2,3\})$ et $P_2(1,3) = (\{1,3\})$ pour distribuer son trafic à destination du routeur 3. Cependant, on constate que cette propriété est relativement rare. Le routeur 4 ne dispose d'aucun chemin alternatif avec ECMP quelle que soit la destination considérée. Le protocole ECMP ne permet pas de protéger les flux acheminés par le routeur 4 en cas de panne. En revanche, la règle LFI permet d'utiliser l'arc $\{4,2\}$ pour protéger le trafic à destination des routeurs 1, 2 et 3 ainsi que l'arc $\{6,7\}$ pour la destination 7.
Le draft *OSPF-OMP* (Vil99b) est basé sur la même règle topologique que LFI, mais le partage de

charge n'est pas nécessairement équitable [6].

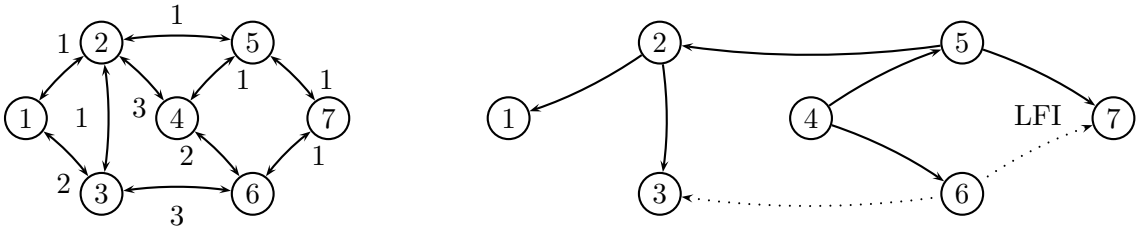
1.2.3.2 LFI-local

Paolo Narvaez et al. définissent, dans (NS99), une méthode de sélection de multichemins entièrement locale. Ils présentent un algorithme de routage multichemins dont la règle de validation est basée sur la condition LFI. Lorsque la condition est vérifiée localement, la phase de validation des NHs peut être confondue avec la phase algorithmique de recherche des chemins. Les auteurs définissent la notion de NH *viable* comme garantissant l'absence de boucle au niveau routeur. Le principal avantage de leur méthode est sa simplicité d'implémentation. Celle-ci peut être utilisée conjointement avec les algorithmes SPT classiques tel que Bellman-Ford, D'Esopo-Pape (Ber91) ou Dijkstra. La différence entre ces algorithmes est déterminée par le mode d'extraction des éléments de la queue des nœuds marqués [7].
Le principe de base sur lequel se fonde leur proposition est la mise en œuvre d'une structure spécifique à chaque routeur s, contenant pour chaque destination d un ensemble d'indications :

- le meilleur coût de s vers d : $C_1(s,d)$.
- un ensemble de voisins v possibles : des prochains sauts candidats.
- pour chaque voisin v, le coût du lien de s à v : $w(\{s,v\})$.
- pour chaque voisin v, le coût du meilleur chemin via $v = NH_j(s,d)$ calculé par s : $C_j(s,d)$.

Ces indicateurs, et en particulier le dernier, peuvent être obtenus lors d'une simple exécution de l'algorithme SPT. Il suffit de considérer les chemins dont le coût n'est pas le meilleur, habituellement ignorés durant la construction de l'arbre des plus courts chemins, et d'enregistrer le premier saut correspondant à la feuille de l'arbre en cours de test, ainsi que le coût du chemin considéré (de coût égal ou supérieur). Ce traitement n'induit qu'une complexité supplémentaire très limitée liée à la mémorisation des multi-NHs possibles. La complexité de l'algorithme SPT reste en $O(|N|^2)$.
Si $C_j(s,d) - w(\{s,v\}) < C_1(s,d)$ alors v est un voisin viable, et le j^e NH calculé par s est utilisable pour le routage vers d sans risque de boucle de routage.

[6] Une description précise du module de partage de charge d'OSPF-OMP sera donné dans le troisième chapitre.
[7] Ces notions seront approfondies dans le second chapitre.



(a) (b) SPT avec $s = 4$

FIG. 1.4 – Figure de base pour les exemples de validation

La relation $C_j(s,d) = C_1(v,d) + w(\{s,v\}) \mid v = NH_j(s,d)$ obtenue avec l'algorithme SPT modifié permet de vérifier la viabilité de chaque NH. La procédure proposée (*Multipath algorithm*, (MPA)) construit un tableau indexé par interface de sortie et trié selon les coûts C_j pour permettre de comparer la qualité des NHs alternatifs. Ainsi, la commutation peut prendre en compte le coût des chemins primaires et alternatifs pour par exemple éviter de commuter les paquets sur les chemins les plus longs. Le calcul local d'un seul SPT ne permet pas d'obtenir tout les NHs. La structure construite avec MPA n'enregistre pas tous les $NH_j(s,d)$ correspondants aux chemins où le meilleur coût décroit strictement entre deux nœuds consécutifs. Par exemple, dans le cas précédent, il peut exister $d \in N$ tel que la relation $C_1(s,d) > C_1(NH_j(s,d),d) > ... > C_1(v_l,d) > C_1(v_k,d) > C_1(v_m,d)$ soit satisfaite. En revanche, si s considère les SPTs de chacun de ses voisins en plus du sien, alors il est capable de déterminer l'ensemble complet de ses voisins v valides au sens $C_1(v,d) < C_1(s,d)$.
Par exemple, sur la figure 1.4, entre les routeurs 2 et 7, un chemin alternatif tel que $(\{2,4\},\{4,5\},\{5,7\})$ ne serait pas enregistré dans la structure MPA car l'algorithme de Dijkstra n'explore pas une deuxième fois les arcs sortants d'un routeur déjà marqué. Or le nœud 5 est marqué avant que le lien $\{4,5\}$ ne soit évalué. Ce chemin pourrait être testé si le routeur 2 avait connaissance du SPT enraciné en 4. Admettons que le coût du lien $\{2,5\}$ soit doublé, alors la condition LFI suffirait à valider ce chemin bien que la structure MPA ne l'ait pas pris en compte.
Pour compenser cette lacune, les auteurs proposent une méthode de validation de NHs plus complète en exécutant en parallèle autant de calculs d'arbres des meilleurs chemins que la racine de calcul a de voisins. En effet, certains coûts $C_j(s,d)$ ne peuvent être déterminés durant le calcul du SPT enraciné sur le nœud s. C'est par exemple le cas si les nœuds non marqués et stockés dans la queue de l'algorithme de calcul sont ordonnés : un nœud $x = v_l$ est extrait de la queue pour l'exploration de ses liens sortants s'il présente le plus petit coût $min(\{C(s,x)\})$ dans l'ensemble des nœuds $\{x\}$ non marqués. A noter que le nœud v_l sera marqué après exploration de tous ses liens sortants. Les coûts de certains chemins ne seront pas enregistrés dans la structure définie par MPA si ils contiennent un chemin de s à v_l suivi d'un arc $\{v_k, v_m\}$ ($v_k \in succ(v_l)$, $\forall v_m \in succ(v_k)$) tel que v_k ai été marqué avant v_l (les arcs sortants de v_k ne sont pas visités plus d'une fois). Sur l'exemple précédent, $v_k = 5$, $v_l = 4 = NH_2(2,7)$ et $v_m = 5$.
Les mêmes auteurs proposent également un calcul de SPT incrémental dans (NST00) plus efficace que les algorithmes SPT statiques traditionnels. Un SPT statique est contraint de recalculer tous les meilleurs chemins lors de la réception d'un LSA de notification alors qu'il est possible qu'aucune, ou très peu, de modifications soient nécessaires pour que la table de routage s'adapte au changement topologique annoncé. Grâce aux algorithmes incrémentaux, seule la partie de l'arbre des plus courts chemins affectée par le changement est recalculée. Les lignes de routages non affectées et acquises lors du précédent calcul SPT sont inchangées pour minimiser les modifications dans la FIB.

Exemple de validation n^o 2

Le routeur 4 est a priori capable d'utiliser l'interface sortante $NH_2(4,3)$ pour acheminer les paquets à destination de 3. En effet, via l'arc $\{4,2\}$, le coût du chemin associé $P_1(2,3)$ vérifie $C_1(2,3) = 1 < C_1(4,3) = 3$. En revanche, aucun chemin alternatif entre 4 et 5 ne peut être validé par la condition LFI. Lorsque les meilleurs coûts des voisins sont connus, la condition LFI produit des résultats topologiques

identiques quelle que soit la nature du processus de validation : locale ou distribuée.

1.2.3.3 LFI-distribuée

Dans le cadre d'un routage à états des liens, il est également possible de vérifier la condition LFI (*Loop Free Invariant*) de manière distribuée en renforcant le critère d'absence de boucles. Chaque routeur s se contente d'un unique calcul du SPT, et diffuse à ses voisins les coûts optimaux ainsi obtenus pour chaque destination. Cette diffusion peut éventuellement être conjointe aux LSA en les utilisant aussi comme des vecteurs de meilleurs coûts. La validation des prochains sauts se fait sur la base de messages contenant les meilleurs coûts via chaque voisin. La complexité de la phase de calcul est ainsi réduite à sa forme minimale et la phase de validation est distribuée.
Vutukury et Garcia-Luna-Aceves ont proposé un ensemble de protocoles de routage basé sur un processus de validation distribué pour assurer l'absence de boucle de routage transitoire. La condition *Loop Free Invariant* permet de garantir un routage sans boucle durant les périodes de transition du réseau. Ce problème concerne aussi bien le routage multichemin que le routage monochemin car des boucles de routage peuvent apparaîtrent durant les périodes transitoires de mise à jour de la FIB consécutives aux changements topologiques.
Ces différentes techniques sont décrites dans le rapport de thèse de Vutukury (Vut01). Ces méthodes se déclinent en deux versions, la première est un protocole de routage à vecteurs de distance (*MDVA*, (VGLA01)) et la seconde est un protocole de routage à états des liens (*MPDA*, (VGLA99)). Nous nous focaliserons sur le routage à états des liens avec *MPDA* dans la mesure où les contributions du chapitre suivant appartiennent à la même classe de routage.
Le processus de validation reste identique quel que soit le mode de routage : les LSA sont modifiés pour contenir des informations de coûts à la manière des vecteurs de distances, nous noterons ces messages des LSU (*Link State Update*) pour les distinguer des LSA usuels. Les LSU sont réactifs aux changements de charge, il s'agit d'un routage adaptatif multichemins.
L'objectif du protocole *MPDA* est la réduction des délais d'acheminement. Les auteurs présentent le problème du routage aux délais minimaux introduit par Gallager (*Minimum Delay Routing Problem*, MDRP (BG87a)). En pratique les solutions algorithmiques proposées par Gallager ne sont pas déployables car dépendent de constantes globales et considèrent que le trafic en transit sur le réseau est quasi stationnaire.
Les auteurs introduisent la notion de coût faisable, généralement égal au coût optimal lors des périodes de stabilité topologique, mais pouvant différer temporairement du meilleur coût durant les périodes de transition topologique du réseau. Le meilleur coût permet d'estimer le délai associé à un chemin. La métrique utilise une valuation à indicateur résiduel sur le délai marginal[8].
Si nous notons ce coût faisable $FC(s,d)$, la condition *Loop Free Invariant* peut alors s'exprimer ainsi :

$$FC(s,d) \leq C_j(s,d) \quad j < k^+(s)$$

[8]Le paragraphe 3.2.2 définit en détails le caractère adaptatif de MPDA.

$$NH(s,s,d) = \{v|\ C_1(v,d) < FC(s,d) \wedge v \in succ(s)\}$$

Un prochain saut candidat v, tel que $v = NH_j(s,d)$, est viable pour le trafic provenant d'une interface d'entrée p s'il appartient à $NH(s,s,d)$ car la commutation avec MPDA est identique quelle que soit l'origine du trafic. Le coût $C_j(s,d)$ correspond au meilleur coût via $NH_j(s,d)$, ces coûts ne sont pas calculés sur s mais sont reportés par les voisins v via les LSU échangés.
Il se peut qu'à un instant donné le meilleur coût faisable diffère du meilleur coût reporté par les voisins pour éviter les boucles de routage durant les périodes de transition. Le terme $FC(s,d)$ est égal au coût minimal entre le meilleur coût actuel et antérieur jusqu'à ce que les routeurs adjacents disposent d'une vision topologique identique.
L'unité d'information échangée est le LSU, son contenu désigne l'état d'un arc e par un triplet correspondant aux nœuds $e.x$, $e.y$ et à son coût $w(e)$. Un LSU annonce l'ajout, le retrait ou changement de coût d'un lien.
La table de topologie principale constituant la RIB est composée d'un ensemble de "sous RIB" construites à partir des LSU reçus par chaque voisin. Ces "sous RIB" sont fusionnées pour former une base de données globale sur laquelle l'algorithme de Dijkstra sera appliqué. Certains liens peuvent poser des problèmes de conflit informationnel. En cas de conflit, c'est la sous RIB offrant le meilleur chemin vers le nœud d'entrée de ce lien qui est utilisée. La RIB globale permet de déterminer les meilleurs coûts $C_1(s,d)$, alors qu'une sous RIB d'un voisin $v = NH_j(s,d)$ permet de connaître $C_j(s,d)$.
Durant une période de transition topologique, le coût faisable peut admettre :
$FC(s,d) = min(C_1^{'}(s,d), C_1(s,d))$ avec $C_1^{'}(s,d)$ désignant le meilleur coût calculé précédemment. Cette différence sera conservée jusqu'à ce que l'ensemble des voisins aient confirmé par l'émission d'un acquittement (noté ACK) la réception et le traitement de la nouvelle information topologique contenu dans le LSU. Pour cela, il faut définir un mécanisme de synchronisation entre routeurs adjacents. Pour satisfaire la condition de routage sans boucles à tout instant, même lorsque les points de vue entre routeurs adjacents peuvent diverger temporairement, il est nécessaire de définir deux états. Un routeur est dans un état ACTIF s'il attend que ses voisins acquittent le LSU qu'il leur a envoyé, ou il est dans un état PASSIF sinon - dès lors que tous les ACK de l'ensemble des voisins sont receptionnés. Durant une période ACTIVE, un routeur ne met pas automatiquement à jour sa base de données topologiques lorsqu'il reçoit de nouveaux LSU liés à l'occurence d'un changement d'état sur des liens $\{e_i\}$. Néanmoins, il peut mettre à jour les sous RIB des voisins annonciateurs, et actualiser le coût des liens vers ces mêmes voisins. Le routeur ACTIF attendra la réception de tous les ACK de ses voisins pour mettre à jour sa base topologique globale. Si l'annonce traitée a modifié sa vision topologique globale (la RIB principale), il retourne immédiatement à l'état ACTIF et diffuse les LSU pour les triplets correspondants aux liens $\{e_i\}$ reçus entre temps - la période PASSIVE est alors implicite. Sinon, dans le cas où sa base topologique principale n'est pas affectée, il peut retourner à un état PASSIF. C'est seulement lorsque le routeur reçoit le dernier acquittement permettant le retour à un état PASSIF que celui-ci peut mettre à jour son coût faisable : $FC(s,d) = C_1(s,d)$.
Chaque routeur s'assure, avant de modifier sa table de commutation FIB, que ses voisins ont bien enregistré les changements topologiques afin que leurs visions topologiques respectives soient cohérentes à

tout instant. Lors du retrait d'un lien par exemple, plutôt que d'acheminer les paquets sur des circuits transitoires, les routeurs continuent à commuter le trafic en fonction de l'ancienne FIB.
Les auteurs proposent également une heuristique de partage de charge à deux échelles de temps. Sur une échelle de temps relativement large T_l l'ensemble des multichemins est recalculé (routage adaptatif). Sur une échelle de temps nettement plus petite, notée T_s, les proportions de répartition de charge sur les différents NHs sont modifiés selon une heuristique de déviation progressive (routage proportionnel). MPDA est un routage multichemins adaptatif et proportionnel. La contrainte sur l'absence de boucle à tout instant est primordiale dans le cadre du routage adaptatif : la fréquence des modifications, et donc le risque de voir apparaître des boucles de routage, est proportionnelle à la durée T_l.
En pratique, leurs propositions ont conduit à l'extension du protocole de routage à états des liens *Cisco*, *Enhanced Interior Gateway Routing Protocol* (EIGRP, (Pep99)). EIGRP utilise une technique de diffusion spécifique pour détecter les boucles transitoires *Diffusing Update Algorithm* (DUAL) et permet de configurer un scalaire de variance pour borner le coût maximal des routes alternatives en fonction du coût optimal. Le protocole de routage à vecteurs de distance *IGRP* (Hed91) permet également le partage de charge sur des routes de coût inégal.

1.2.3.4 Routage sans boucle au niveau lien

Yang et Wetherall proposent un protocole de routage multichemins (YW06) garantissant l'absence de boucles de routage au niveau lien. Sa singularité est la possibilité pour un paquet de traverser plusieurs fois un routeur donné. Les auteurs présentent trois règles de validation adaptées au multi-routage saut par saut.
La première règle (1) est la même que LFI. La seconde règle (2) fait intervenir (1) et la condition SPD donnée dans le tableau 1.3. L'objectif est de définir une décroissance stricte du meilleur coût relâchée à une vision à deux sauts : entre le routeur en amont et le routeur en aval. L'espace des NHs activés est appelé un ensemble de *déflections*. Cet ensemble est relatif à une destination et une interface entrante. D'une part, il est nécessaire de distinguer l'interface par laquelle entre le trafic pour adapter le routage en fonction du routeur en amont. D'autre part, afin d'étendre l'ensemble des prochain sauts activables, le coût du lien prédecesseur p peut être considéré comme infini pour le trafic générée par s (pour $s = p$). Cependant, le routeur prédecesseur p est a priori retiré de l'ensemble des prochains sauts actifs à moins que cet ensemble soit vide si privé de p. Ce retrait permet déviter les boucles de longueur 2 entre p et s, car ces *déflections* sont inintéressantes. Cette seconde condition prévient la formation de boucles de routage au niveau lien. Un paquet traversant plusieurs fois un routeur donné sera commuté sur une interface de sortie différente que lors de ses précédents passages.
La troisième règle (3) est une amélioration de (2), elle permet d'éviter les boucles sur lien générées sur le routeur voisin v. Durant le calcul du SPT, l'arc reliant s et v est retiré du graphe. Par ailleurs, l'objectif de cette dernière condition est de considérer des chemins non activables avec la seconde règle. Soit $G \setminus e$ désignant le graphe G privé du lien e dans ses deux orientations (e et e^{-1}) et soit $C_1 \setminus e(s, d)$ le meilleur coût calculé entre s et d sur le graphe $G \setminus e$. Un prochain saut v est ajouté à l'espace des déflections du routeur s (les NHs actifs $NH(p, s, d)$) pour une destination d, si l'une des conditions

suivantes est satisfaite tel que $v \neq p$:

$$C_1 \setminus \{s, v\}(v, d) < C_1 \setminus \{p, s\}(s, d)$$

$$C_1 \setminus \{s, v\}(v, d) < C_1(p, d)$$

Les espaces de déflections produits par les règles étendues (2) et (3) produisent deux ensembles de NHs actifs différents. Aucun de ces ensembles ne recouvre l'autre grâce au terme $C_1 \setminus \{p, s\}(s, d)$ de la première condition de la règle (3). La règle (2) produit des résultats quantitatifs alors que la troisième assure des résultats qualitatifs en terme de diversité.
Avec (2), dans l'absolu plus de NHs sont activables, cependant la règle (3) construit un espace de déflections possédant une meilleure couverture. Les chemins activés sont plus disjoints car cette règle favorise l'utilisation de chemins plus longs. Le processus de validation est effectué sur un SPT réduit, certains arcs d'orientation opposée à ceux appartenant aux meilleurs chemins sont absents : $G \setminus e$, $e \in P_1$. L'espace de déflection est ainsi constitué de chemins alternatifs dont le coût est plus élevé qu'avec la règle (2). Les conditions de déflections sont moins strictes à mesure que les coûts $C_1 \setminus \{p, s\}(s, d)$ augmentent.
Lorsque le routage est à états des liens, la règle (3) nécessite un calcul de SPT incrémental associé à chaque lien supprimé afin d'obtenir les meilleurs coûts sur les graphes réduits. Si les meilleurs coûts sont diffusés sous la forme de vecteurs de distance, le calcul est alors plus simple car l'information en dérive directement.
Les auteurs proposent en outre une approche analogue à la règle (3) où ce n'est plus l'arc prédecesseur qui est supprimé du graphe mais le nœud d'entrée. Par ailleurs, pour minimiser les problèmes liés aux mécanismes de retransmission TCP, le routage est contrôlé par la source émettrice grâce à un *tag*. Celui-ci est positionné à la source dans l'en-tête des paquets pour choisir à chaque saut la déflection la plus adaptée en fonction de certains critères de QoS.

Exemple de validation n^o 3

La règle (2) active le chemin alternatif reliant 4 et 1 et passant par les nœuds 6 et 3. La règle (1) n'en est pas capable. En effet le meilleur coût sur 3 à destination de 1 est strictement inférieur au meilleur coût de 4 vers 1. La règle (3) permet d'activer le chemin alternatif $P_2(5, 7) = (\{5, 4\}, \{4, 6\}, \{6, 7\})$ pour atteindre la destination 7. Le routeur 4 peut en effet ignorer l'arc $\{4, 5\}$ correspondant à son meilleur NH vers 7, lorsqu'il active les déflections associées à l'interface entrante 5.
Pour illustrer le phénomème de boucles de routage sur liens, on remarque que la règle (2) autoriserait le routeur 2 à utiliser 1 dans son ensemble de déflections vers la destination 7. Or le routeur 1 n'a d'autres choix que de renvoyer le trafic provenant du routeur 2 précisement vers celui-ci. Heureusement, le routeur 2 n'a qu'une seule déflection active via l'interface de sortie 5 lorsque le trafic provient de l'interface entrante 1 à destination de 7. Il n'existe pas de boucle de routage sur les liens $\{1, 2\}$ et $\{2, 1\}$. La règle (3) permet d'éviter ce type de boucle sur routeur. En revanche, des boucles de routage au niveau routeur composées de plus de deux nœuds peuvent se former.

La partie suivante introduit les techniques dites de reroutage rapide. Le changement de commuta-

tion entre le prochain saut primaire et le prochain saut alternatif s'effectue uniquement lors d'une panne. Le routage multichemins n'est pas aussi restrictif, il peut effectuer, en cas de congestion, le reroutage partiel d'un excédent de charge.
Néanmoins, ces deux formes de routage au saut par saut présentent une problématique commune : les paquets déviés ne doivent pas *boucler*. Dans le cadre du contournement de pannes et lorsque le routage mis en place vérifie la propriété de sous-optimalité, les règles pour l'absence de boucles de routage ne sont pas soumises aux mêmes contraintes topologiques.

1.2.4 Reroutage rapide sur IP

Les protocoles de reroutage rapide présentés dans cette partie ne permettent pas l'utilisation simultanée de plusieurs prochains sauts vers une destination donnée. L'objectif est de pré-calculer des NHs de secours pour anticiper l'occurence d'une panne et augmenter la fiabilité du réseau. Les NHs alternatifs garantissent un reroutage plus rapide que lors d'une reconfiguration complète du domaine de routage [9]. Deux modes de routage coexistent, le mode *normal* lorsque le réseau est opérationnel et le mode *reroutage* dès lors qu'une panne intervient. L'activation du NH de secours sera effective uniquement après la détection de la panne.
Les protocoles de reroutage sont soumis à une contrainte de routage sans boucle moins stricte que lorsque le routage est multichemins. Les conditions et règles décrites dans la partie précédente sont trop restrictives.
Une propriété supplémentaire permet de réduire la contrainte sur l'absence de boucles : en mode *normal*, seuls les chemins optimaux sont utilisés. Le chemin primaire $P_1(v,d) = (e_1, ..., e_i, ..., e_m)$ calculé par le voisin v d'un routeur s ne doit simplement pas traverser le nœud s ($e_i.y \neq s \ \forall 1 \leq i \leq m$) pour que v soit un NH de secours viable vers d. En pratique, il peut exister plusieurs chemins optimaux, il est alors nécessaire de garantir cette contrainte sur l'ensemble de ces chemins. Pour cela, il suffit d'analyser les meilleurs coûts du voisinage. La table 1.4 fournit deux règles de reroutage rapide dont la complexité est proportionnelle aux dimensions du voisinage. Pour obtenir les meilleurs coûts des voisins, il suffit de calculer leur SPT.

Pour une destination d donnée, la règle LFA signifie que v dispose d'un meilleur coût strictement inférieur à la somme du meilleur coût de s et du meilleur coût entre s et v : aucun chemin optimal de v vers d ne passe par s. L'ensemble $LFA(s,v,d)$ désigne les alternatives LFA de v vers d ne passant

[9] La partie 3.1 approfondit la notion de protection pour générer une couverture complète.

Condition sur v	Nom	Références
$C_1(v,d) - C_1(v,s) < C_1(s,d)$	LFA	(AZ07)
$C_1(v,d) - C_1(v,s) = C_1(s,d) \wedge LFA(s,v,d) \neq \emptyset$	UTURN	(Atl06)

Tab. 1.4 – Règles de reroutage monochemin sans boucles

pas par s.
Les trois paragraphes qui suivent décrivent les principales méthodes existantes de la littérature. Nous nous focaliserons sur les aspects topologiques sans aborder les détails techniques des implémentations.

1.2.4.1 Sortie de secours

Wang et Crowcroft ont proposé dès 1990 un algorithme de reroutage intitulé *SPF-EE* (*Shortest Path First with Emergency Exit*, (WC90)). Il s'agit d'un protocole de reroutage rapide sensible aux congestions. Dès lors que la file d'attente d'un prochain saut primaire $NH_1(s,d)$ commence à être saturée selon un seuil donné, le routeur s bascule le trafic vers un prochain saut alternatif.
Soit v un NH alternatif candidat pour protéger le lien primaire d'un routeur donné s vers une destination d : si $s = f(v,d)$, c'est-à-dire que s est un routeur traversé par le meilleur chemin de v menant à d ($\exists i \mid e_i.x = s \in P_1(v,d)$), alors v n'est pas directement utilisable comme sortie de secours vers d.
Dans le cas contraire, $s \notin f(v,d)$, s peut utiliser v comme prochain saut de secours vers d. Ce NH alternatif sera utilisé jusqu'à ce que la file d'attente du chemin primaire se vide suffisamment. Par définition, v propose un chemin *Alternate Path* : *AP*.
S'il n'existe aucun voisin proposant un AP, la procédure est alors reportée via un ensemble de requêtes sur le voisinage. Si le voisinage à deux sauts ne dispose d'aucun AP pour s, les requêtes sont propagées jusqu'à ce qu'une alternative soit proposée par un routeur de sortie noté t. Si t n'utilise pas s pour son trafic vers d sur son chemin primaire, on peut alors introduire la notion de chemin *Reverse Alternate Path* (*RAP*). Cela signifie que le routeur t dispose d'un AP par rapport au voisin lui ayant transmis la requête. Le routage entre s et t est de type *source-routing* et peut à nouveau basculer en mode saut par saut depuis t jusqu'à d.
En pratique, lorsqu'il n'y pas d'AP sur le voisinage à un saut du routeur s, celui-ci envoie un paquet de contrôle vers chacun de ses voisins jusqu'à ce qu'un RAP soit trouvé. Il existe une solution dès lors que les sous arbres des meilleurs chemins se séparent, ce qui est toujours possible si le réseau n'est pas partitionné.
La recherche d'AP et de RAP peut se faire dynamiquement ou par anticipation. Le calcul du SPT doit être enrichi d'un calcul incrémental sur les meilleurs chemins des voisins avec un algorithme comme ceux fournis dans (MRR80). La propagation des messages pour la validation des chemins RAP est limitée à un nombre de sauts fini et la maintenance des tables de routage pour les entrées de secours est soumis aux mêmes conditions que pour les entrées primaires.
La décision de redirection se base sur deux indicateurs, d'une part les files d'attente des NHs (primaires ou AP/RAP) sont surveillées au moyen de seuil, d'autre part, les paquets redirigés sont marqués pour empêcher qu'ils ne soient à nouveau redirigés. Le paquet est alors éliminé sinon cela provoquerait la formation de boucles de routage. Par ailleurs, si aucun chemin primaire, AP ou RAP ne satisfait un taux d'occupation inférieur aux seuils choisis, alors une recherche d'AP/RAP est initiée dynamiquement.
Cette méthode a plusieurs avantages, elle est multichemins sur une échelle de temps macroscopique et dans la limite d'un seul *détour* au maximum. Elle permet par évitement de congestion d'augmenter le flot maximal entre une paire de nœuds. De plus, elle ne souffre pas des oscillations de charge du type

tout ou rien qu'entraîne typiquement un routage adaptatif. La déviation est locale et temporaire.

Exemple de validation n^o 4

Pour illustrer les notions d'AP et de RAP, reportons nous à la figure 1.4. Si le lien $\{4,5\}$ est fonctionnel et que le taux d'occupation de sa file d'attente est inférieur au seuil choisi, le routeur 4 utilise le chemin optimal $P_1(4,7) = (\{4,5\},\{5,7\})$. Dans le cas contraire, il peut utiliser le chemin AP $(\{4,6\},\{6,7\})$ car le chemin primaire du routeur 6 vers 7 ne passe pas par 4.

Considérons la protection du chemin primaire $P_1(7,2) = (\{7,5\},\{5,2\})$ entre le routeur 7 et le routeur 2. Le chemin $P_1(6,2)$ du routeur 6 vers 2 contient le routeur 7 ($f(6,2) = 7$), ce voisin ne peut donc pas être utilisé directement en tant que premier saut d'un chemin AP. Néanmoins, le routeur 3, à deux sauts de 7, dispose d'un plus court chemin vers 2 ne passant pas par 7. Le chemin $(\{7,6\},\{6,3\}\,\{3,2\})$ est un chemin RAP. Le routage entre 7 et 3 sera contrôlé par la source (*source-routing*). Pour cela, il faut modifier l'en-tête IP des paquets provenant de 7 et utilisant 6 comme NH de secours. L'en-tête de ces paquets comportera une information notifiant l'obligation de passer par le prochain saut 3.

1.2.4.2 Alternative sans boucle : LFA

Le groupe de travail de l'IETF *IP Fast Reroute* envisage la normalisation d'une solution de reroutage (AZ07) dont l'implémentation est relativement simple. Cette approche, *Loop Free Alternate* (LFA), est semblable aux chemins AP vus dans le paragraphe précédent mais s'adapte uniquement au cas de la panne de lien ou de routeur.

Pour une destination d donnée, la condition de validation d'un NH alternatif v pour protéger le lien primaire d'un routeur s est donnée dans le tableau 1.4. Contrairement à SPF-EE, le test $s \notin\in f(v,d)$ est réalisé sur le coût des meilleurs chemins. Néanmoins, la connaissance de ces coûts nécessite le calcul d'un SPT dont la racine est v. Chaque routeur s doit calculer l'ensemble des coûts $C_1(x,d)$, à la fois pour $x = s$ et aussi pour $x \in succ(s)$, c'est-à-dire déterminer l'ensemble des meilleurs coûts de chaque routeur voisin. Un tel calcul présente une complexité au pire de $0(k^+(s) \times N^2)$ mais l'utilisation d'algorithmes SPT incrémentaux permet de réduire la charge de calcul.

La condition LFA est adaptée à la protection de liens, mais il est possible d'ajouter une condition supplémentaire pour la protection de routeur : si le nœud à protéger est le saut primaire de s vers d, c'est-à-dire $NH_1(s,d)$, alors il suffit que $C_1(v,d) < C_1(v,NH_1(s,d)) + C_1(NH_1(s,d),d)$. Si cette condition et LFA sont satisfaites, le chemin de secours ne peut pas emprunter l'interface de sortie primaire correspondant à $NH_1(s,d)$.

Exemple de validation n^o 5

Sur la figure 1.4, le routeur 1 peut utiliser le chemin alternatif passant par les routeurs 3 puis 2 pour atteindre 4 et ainsi protéger le lien $\{1,2\}$. Cette alternative LFA ne suffit pas à protéger le routeur 2. De plus, le chemin passant par 3 et 6 n'est pas validé avec la condition LFA. On notera que $LFA(2,1,4) = \emptyset$.

1.2.4.3 Alternative U-TURN

Lorsque la topologie est faiblement maillée, la couverture générée par la technique *LFA* en terme de protection de routeurs ou de liens est généralement insuffisante. Ainsi le groupe de travail sur

le reroutage rapide a proposé une extension nommée *U-turns Alternates* (notée *UTURN*, (Atl06)). Il s'agit d'augmenter la couverture en terme de capacité de protection lorsqu'un routeur s n'a pas directement dans son voisinage d'alternative LFA vers certaines destinations. Un voisin v ne vérifiant pas la condition LFA utilise nécessairement s sur son meilleur chemin. Pour une destination d donnée, le routeur s utilisera alors ses voisins v possédant une alternative *LFA* pour profiter des alternatives dans un voisinage à deux sauts. Ces voisins sont des alternatives *UTURN* pour s.
Cette procédure est plus complexe. Deux cas sont possibles : soit v utilise s comme unique prochain saut vers d, soit v dispose d'un ensemble de chemins aux meilleurs coûts égaux vers d dont l'un utilise s comme prochain saut, dans ce cas v est une alternative *ECMP-UTURN*.
Une alternative *ECMP-UTURN* est un routeur aux multiples meilleurs chemins optimaux. Un ou plusieurs de ces chemins assurent la protection du premier saut primaire d'un voisin s pour le lien $e_1 = \{s, NH_1(s, d)\}$). Cette alternative est un routeur adjacent à s vérifiant la condition UTURN du tableau 1.4 et n'incluant pas le lien e_1 dans son meilleur chemin vers d. Un tel routeur v dispose d'une alternative LFA vers d via au moins un de ses prochains sauts optimaux.

Sur la figure 1.5, en considérant la valuation des liens uniforme, le routeur s peut utiliser v comme alternative UTURN pour la destination d. Le lien $\{s, 1\}$ est protégé car v dispose d'une alternative LFA vers d ne passant pas par celui-ci : le routeur 2.
La méthode *UTURN* requiert une coopération entre les voisins v et s pour protéger le chemin primaire de s vers d en cas de panne sans que les paquets n'empruntent un circuit. Le routeur v doit signaler à s qu'il est capable de lui proposer une alternative LFA vers d. Cette technique nécessite donc une distinction sur le lien entrant du trafic pour éviter que les paquets déviés ne puissent parcourir une boucle de routage. L'utilisation explicite de l'interface d'entrée permet un routage avec une granularité plus fine : les flux provenant de s seront traités différemment des flux provenant des autres interfaces. L'alternative *UTURN* vérifiera, avec une table de routage considérant la provenance du flux, que les paquets provenant de s ne lui soit pas retransmis.
Admettons, pour généraliser la méthode *UTURN*, que le voisin v dispose d'un unique meilleur chemin passant par s, tel que la condition *LFA* soit insatisfaite entre s et v. Le routeur v est un voisin *UTURN* pour s vers la destination d seulement si v dispose d'une alternative sans boucle *LFA* vers d, un voisin noté r tel que r satisfait la condition *LFA* pour le couple (v, d). Lorsque v signale à s sa capacité à protéger le lien $\{s, v\}$, s doit s'assurer que le chemin alternatif de v via r ne passe pas par s. De même qu'avec une alternative *ECMP-UTURN*, l'interface d'entrée permettra de distinguer l'origine du trafic.

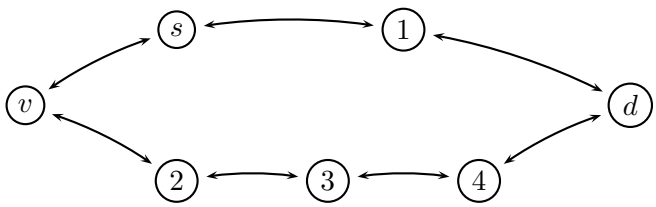


FIG. 1.5 – v est une alternative UTURN pour le couple (s, d)

L'identification d'un paquet dévié avec UTURN peut être implicte ou explicite. Elle est implicite si v contaste simplement qu'un paquet à destination de d provient de son prochain saut primaire pour d. L'identification est explicite si l'utilisation d'un label est nécessaire.
Le niveau de protection peut être ajusté pour protéger le nœud primaire de s en s'assurant que le chemin alternatif de v via r ne passe pas par ce nœud. Dans les deux cas, il est nécessaire de savoir si le chemin alternatif *LFA* de v passe ou non par s (pour le cas de la panne de routeur, si ce chemin passe ou non passe par $NH_1(s,d)$). Ces caractéristiques peuvent être déterminées en considérant le SPT inversé vers s pour connaître $C_1(r,s)$.
Exemple de validation n^o 6
Dans l'exemple donné en figure 1.4, le routeur 3 est une alternative UTURN pour le routeur 2 à destination de 7. Le nœud primaire ainsi protégé est $NH_1(2,7) = 5$. Il est à noter, sur cet exemple, que 2 dispose aussi d'une alternative LFA via le routeur 4 mais que celle-ci ne protège que le lien $\{2,5\}$. Le routeur 3, quant à lui, utilise 2 comme NH primaire vers 7 et dispose d'une alternative LFA via 6 à destination de 7. Or le chemin via 6 ne passe pas par 2 et permet ainsi de protéger le routeur 5. Grâce à UTURN, le routeur 2 dispose d'une couverture complète de son chemin primaire $2-5-7$ via son voisin UTURN 3 et le chemin alternatif $2-3-6-7$. Par ailleurs, on remarquera que le routeur 1 ne peut être un voisin UTURN pour 2 à destination de 7 dans la mesure où son alternative LFA, le routeur 3, utilise le routeur 2 sur son chemin primaire $3-2-5-7$.

1.2.4.4 Discussion

Les algorithmes de routage multichemins doivent vérifier des conditions de validation plus strictes que les méthodes de reroutage rapide. En effet, les routes alternatives calculées par les protocoles de reroutage rapide ne peuvent pas être utilisées conjointement à la route primaire. De plus, en cas de pannes multiples ou lorsque la détection de panne est mal interprétée, la stabilité de ce type de méthode est soumise à conflit.
Lorsque la détection de panne est accélérée pour diminuer le temps de réaction, cela peut aboutir à la confusion entre lien très chargé et lien en panne. Dans ce cas, les détections de pannes multiples peuvent être fréquentes d'autant plus que la restauration d'une route produit de brusques et importants mouvements de charge pouvant à leur tour être interprétés comme des pannes. Les coupures et les redirections de charge qui en résultent génèrent des congestions imprévisibles car le réseau n'est plus correctement dimensionné.
Lorsque des pannes multiples et antagonistes se produisent, des boucles de routage peuvent apparaître alors que les méthodes multichemins se contentent de détruire les paquets des flux non redirigeables jusqu'à ce que le routage se stabilise. L'apparition de boucles de routage persistantes peut entraîner l'émergence de nouveaux problèmes pour les flux non concernés par les pannes. Ce phénomène peut notamment générer de nouvelles confusions récurrentes entre liens chargés et pannes de liens. La durée transitoire d'une telle boucle est conditionnée par l'intervalle de temps commun entre les pannes. Ainsi, le problème peut s'étendre sur une aire de routage relativement large à mesure que les pannes et/ou les confusions se succèdent.

La stabilité des mécanismes de commutation est un enjeu majeur pour garantir la qualité globale du routage. Suite à une panne, lorsqu'un routeur ne dispose plus de son chemin primaire et que le chemin alternatif n'est pas fiable, il est assurément plus prudent de détruire les paquets que de créer des boucles de routage transitoires qui pourraient contribuer à la propagation de situations critiques.

1.3 Routage par la source

La sélection des chemins et le positionnement des informations de routage pour la commutation peuvent être contrôlés par les sources. Le terme source désigne généralement un routeur d'entrée de domaine (un routeur d'accès par exemple), mais peut correspondre à l'équipement émetteur du flux ou désigner un routeur intermédiaire.
Dans cette partie, nous nous intéresserons tout particulièrement au routage à commutations de *labels*. Le routage piloté par la source ne signifie pas nécessairement que le paquet IP contienne explicitement la route dans son en-tête IP. Les mécanismes de ce type inscrivent dans le paquet l'ensemble du chemin que le paquet doit suivre depuis la source émettrice jusqu'à la destination. Les RFCs 791 (Pos81) et 1812 (Bak95) décrivent les prérequis de base pour marquer et traiter ces champs pour des routeurs IPv4. Les méthodes de *source routing* avec marquage de l'en-tête sont très peu extensibles. D'une part, la taille de l'en-tête IP est dépendante de la longueur de la route, d'autre part ce type de routage ne peut s'appliquer de bout en bout sur l'Internet que si l'ensemble des routeurs coopèrent. Le source routing est très peu utilisé pour des considérations de sécurité.
Dans le cas d'un routage à la source *strict*, une route est définie comme étant la liste des routeurs à parcourir entre la source et la destination. Cette méthode est relativement lourde. En revanche, l'utilisation d'un routage à la source relâché (*Loose Source and Record Route*, LSRR) est un mécanisme plus flexible : seul un sous-ensemble de routeurs intervient formellement dans la définition de la route. Le *source routing* peut par exemple être utilisé pour des outils de cartographie type *traceroute*.
Certains principes, comme le marquage relâché avec LSRR, sont aussi utilisés par les mécanismes de routage à commutation d'étiquettes. Dans ce cadre, nous nous focaliserons davantage sur les mécanismes de marquage explicite des routes. La racine de calcul positionne un ensemble de chemins en fonction de critères de QoS configurables. Dans le cas où la commutation par étiquette est relâchée, elle se base alors sur le protocole de routage IGP sous-jacent : la diversité des chemins et la flexibilité du routage dépendent du routage au saut par saut déployé. Cette dépendance est relative au segment de route non explicitement positionné.

1.3.1 Contexte technologique et commutation d'étiquette

Le routage par commutation d'étiquette avec positionnement explicite des routes, est assimilable à un routage par la source. Si le marquage des routes est stricte, alors le routage est entièrement piloté par la source bien que la commutation soit réalisée saut par saut. La source désigne le point de calcul qui détermine de bout en bout le chemin à prendre.
Il s'agit dans un premier temps de positionner des informations de commutation appelés *labels*. L'ar-

chitecture *Multi Protocol Label Switching* ((RVC01), MPLS) avait pour objectif initial d'accélérer la commutation IP traditionnelle par correspondance sur l'adresse destination. L'utilisation d'une FIB basée sur des labels très courts et agrégeant éventuellement un ensemble de destinations permet de réduire le temps de commutation.
MPLS permet de déployer des politiques de QoS et de mettre en œuvre des méthodes d'ingénierie de trafic (voir le RFC 2702 (AMA$^+$99)) plus aisément qu'avec un routage saut par saut classique. La commutation des paquets peut être unifiée avec ce type de méthodes grâce à la mise en place d'une couche d'abstraction supplémentaire. Que ce soit au niveau du protocole (IPv4 ou IPv6) ou du mode de forwarding (unicast avec ou sans QoS, multicast, etc) les différents champs utilisés (destination, champ QoS, etc) pour la commutation sont translatés vers de simples labels. Ces labels représentent une certaine classe d'équivalence de routage (*FEC, Forwarding Equivalent Class*) et permettent éventuellement de définir plusieurs chemins entre certaines paires de nœuds appelées routeurs d'entrée/sortie. Ces chemins sont similaires à des tunnels (ou liaisons logiques) et sont positionnés pour une FEC donnée. Le routeur d'entrée encapsule le paquet avec un label qui dépend de cette FEC. L'attribution d'une étiquette à une FEC autorise l'empilement de labels. C'est le cas lorsqu'un paquet traverse un routeur attributeur de labels alors qu'il a déjà été étiqueté par un autre routeur d'entrée. Le routage dans un tunnel est ensuite réalisé par commutation de labels jusqu'au routeur de sortie. Une FEC permet de définir un traitement homogène en termes de routage (même NH, même file d'attente si QoS, etc) pour l'ensemble des paquets des flux classés dans une FEC donnée. La granularité d'une FEC est très variable, elle peut aussi bien désigner un large préfixe réseau qu'un sous-ensemble de flux aux caractéristiques précises.
La FIB d'un routeur appartenant à un domaine MPLS (un tel routeur est appelé *Label Swith Router*, LSR) utilise une commutation de type :
(port d'entrée, label d'entrée)→(label de sortie, port de sortie)
La commutation dans un domaine MPLS peut prendre trois formes différentes selon qu'elle soit réalisée en interne, en sortie ou en entrée.
En interne, le paquet est étiquetté et suit la règle définie par la FIB MPLS. Celle-ci peut simplement correspondre à la commutation IP classique sur le meilleur prochain saut lorsque le routage est relâché (LSRR).
En périphérie de domaine, le routeur d'entrée (*Ingress LSR*) a la charge de classifier le paquet dans la FEC appropriée et détermine le label à lui attribuer. Le routeur de sortie (*Egress LSR*) désencapsule le paquet en retirant l'étiquette et le commute vers le prochain saut adéquat. L'association entre FEC et label peut se faire dynamiquement à l'arrivée d'un nouveau flux ou être pré-programmée sur des critères indépendants du trafic, mais, dans ce cas, certaines associations peuvent être inutiles et l'ingénierie de trafic n'est pas réactive.
Pour une FEC donnée, il existe un, ou plusieurs, Egress LSR pour plusieurs Ingress LSR possibles. Le routeur de sortie est relié aux routeurs d'entrées via des *Labels Switch Paths* (LSP), on parle de *LSP-tree* pour désigner une union de LSP partageant une FEC et un Egress LSR commun. La création de LSP peut être déclenchée soit par le routeur de sortie, soit par tout LSR du domaine, afin de

paralléliser la construction de LSP (risques d'incohérences). La figure 1.6 illustre schématiquement les principaux termes utilisés pour le positionnment explicite de LSP. Sur cet exemple, le routeur d'entrée numéroté 1 dispose de trois LSP vers le routeur de sortie 8 : il s'agit d'une coupe minimale permettant d'augmenter le débit entre la source 1 et le puits 8. Sur le LSR numéroté 4, un paquet provenant de l'interface 2 et dont le label est L sera commuté sur l'interface de sortie 6 et le label L sera remplacé par L'.
Le positionnement des LSP peut être de plusieurs natures. La distribution de labels dans un domaine MPLS peut être déterminé par le protocole de routage IGP sous-jacent (*loose*) ou avec des protocoles spécifiques comme *Label Distribution Protocol* (LDP, (ADF+01)). On peut donc distinguer la phase de calcul des routes de la phase de distribution des étiquettes : dans le cas d'une positionnement *loose*, les sources ne calculent pas les routes, celles-ci sont mis en place sur la base des FIB du protocole IGP.

1.3.2 Protocole de marquage/positionnement et distribution de labels

Le protocole LDP est fondé sur l'échange de messages d'adjacence entre LSR voisins. La distribution des labels est implicite, le routage est relâché. Si la distribution de labels est initiée depuis le routeur d'entrée, on parle de *Mode dowstream Demand* : les routeurs transmettent de proche en proche des messages *request* sollicitant des réponses de confirmation *mapping*. Si la distribution se fait depuis le routeur de sortie, on parle de *Mode unsollicited downstream*.
Dans le contexte de l'ingénierie de trafic sur MPLS, deux protocoles de positionnement explicite ont été proposés. Ces mécanismes *Explicitly Routed Path* (ER-LSP) sont indépendants du protocole de routage IGP. Les protocoles *RSVP-TE* (ABG+01) et *CR-LDP* (Constraint-Based Routing, (JAC+02)) permettent de marquer les routes de manière stricte sans les insérer dans l'en-tête des paquets. Certains segments d'acheminement peuvent être relâchés (*route pinning*).
Le protocole de marquage avec réservation *RSVP-TE* utilise des principes analogues à RSVP basé sur l'architecture *Intserv*. Le routeur attribuant les labels aux FEC, associe chaque label à un flux *RSVP* en utilisant le mode *dowstream Demand*. Les messages *RSVP-Path* (équivalent aux *Label Request*)

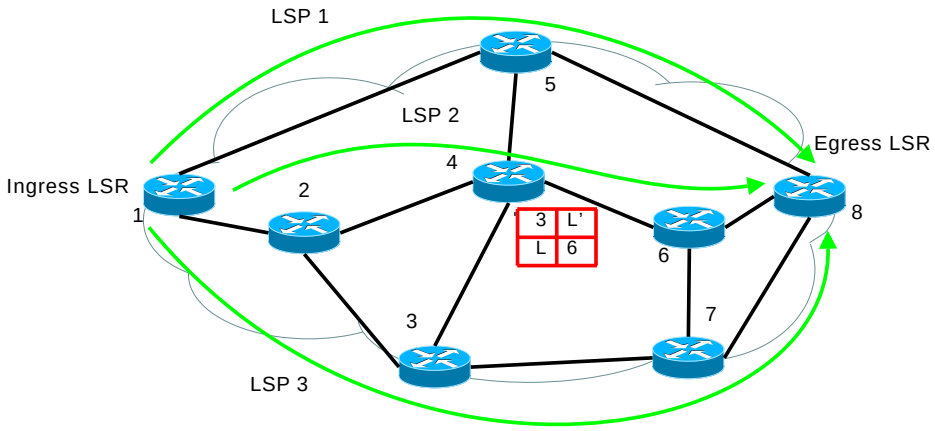


FIG. 1.6 – Label Switch Paths sur un domaine MPLS

contiennent explicitement une liste de nœud abstrait (Adresse IP, Préfixe, etc, stocké dans un champ *Explicit Route Object*, ERO, avec le champ *TLV*). Cette liste est dite lâche (*loose*) si deux nœuds consécutifs ne sont pas voisins le long du LSP. Le prochain saut est strictement défini lorsque le LSP est contraint de vérifier l'ordonnancement exacte de la liste de nœuds abstrait. La réponse *RSVP-Resv* émise par le Ingress LSR distribue les labels sur le chemin parcouru par le message *RSVP-Path*. Ces labels sont assignés à la FEC déterminée par le routeur d'entrée.
Le cheminement du LSP peut être enregistré par les messages *Path* et *Resv* au moyen du champ *Record Route Object* (*RRO*). Ce champ RRO est notamment utile lors de l'établissement d'un LSP relâché car il permet d'enregistrer l'adresse de chaque LSR traversé pour connaître les segments de routage imprécis (route pinning), détecter les boucles de routage éventuelles et définir des LSP disjoints.
Les messages périodiques de rafraîchissement (mode *soft state*) permettent l'adaptation dynamique du LSP vers une modification partielle ou totale lorsque les besoins du flux évoluent ou lorsque la topologie change. Le protocole de distribution de labels *CR-LDP* est au contraire basé sur un mode *hard-state* comme LDP. Les ER-LSP positionnés avec CR-LDP supportent également une fonction de préemption, c'est-à-dire l'établissement ou le maintien de priorité. Actuellement, le protocole de signalisation *RSVP-TE* semble être privilégié au détriment de *CR-LDP*.
L'utilisation d'un routage explicite permet de positionner des routes de bout en bout d'un domaine MPLS selon les critères de QoS souhaités par l'administateur. Le routeur d'entrée de domaine (le Ingress LSR) peut, par exemple, utiliser des algorithmes de calcul de chemins sur des graphes réduits pour satisfaire des contraintes de routage précises.
Par ailleurs, les extensions d'ingénierie de trafic apportées au protocole de routage OSPF dans le RFC 3630 (*OSPF-TE* (KKY03)) permettent de diffuser des messages *Opaque LSA* (*OLSA*, avec champ *TLV*) contenant un certain nombre d'informations utiles à l'établissement et aux choix des LSP. Il peut s'agir d'indicateurs tel que la bande passante réservable maximale, la bande passante non réservée (pour RSVP-TE) et la bande passante disponible.

1.3.3 Commutation d'étiquette orientée QoS

Les techniques de multiroutage à la source basées sur la technologie MPLS sont fondées sur deux opérations successives : calcul de chemins et distribution explicite du processus de commutation. La phase algorithmique est relativement proche des méthodes de calcul des chemins traditionnellement utilisées dans le cadre du routage saut par saut. Il s'agit en général de variantes autour des algorithmes de Dijkstra ou Bellman Ford. En revanche la commutation est stricte et explicitement positionnée par la source : il s'agit d'un ensemble de liaisons logique construit comme un réseau couvrant au-dessus de la couche réseau. Ainsi, la phase de signalisation explicite mise en œuvre avec CR-LDP ou RSVP-TE assure l'absence de boucle de routage si le calcul de chemins à la source est cohérent. Un Ingress LSR souhaitant mettre en place plusieurs chemins vers un Egress LSR pour de l'ingénierie de trafic doit disposer de quatre composants :

1- Un algorithme de calcul à la source pour réduire les délais de routage, augmenter le débit ou calculer des chemins de protection efficace.

2- Un protocole de signalisation pour positionner les chemins calculés en 1- (tel que CR-LDP ou RSVP-TE).

3- Un protocole de notification pour analyser l'état des multichemins explicites (tel que OSPF-TE et les messages OLSA).

4- Un algorithme de répartition de charge réagissant aux informations reçues.

Le paragraphe suivant introduit un ensemble de procédures algorithmiques nécessitant un positionnement explicite et permettant de calculer de multiples chemins vérifiant des contraintes de QoS diverses.

1.3.3.1 Algorithme de sélection de chemins

Le choix de l'algorithme de calcul n'est pas soumis aux mêmes contraintes que dans le contexte du multiroutage saut par saut. D'une part, le premier saut n'est pas nécessairement distinct entre chaque chemin calculé pour une paire de nœuds donnée, et d'autre part, l'utilisation d'un critère de validation est inutile. Contrairement au routage distribué, il est notamment possible de vérifier de manière exacte une contrainte concave tout en assurant l'obtention d'un minimum global sur une métrique additive.
L'utilisation de graphes réduits permet de satisfaire des contraintes de QoS variés. Il s'agit de supprimer un ensemble d'arcs ou de nœuds non conformes à une ressource critique sous réserve que le graphe reste connexe. Une autre approche consiste à proposer une forme d'équité globale entre les flux ou classes de flux. L'objectif est de proposer un partage équitable des ressources critiques et en particulier la bande passante des liens de capacité minimale. Cette méthode est appelée *max min fair share* (voir par exemple (MSZ96)). Il existe deux manières d'implémenter ce type de techniques. Soit l'équité est gérée par une politique d'ordonnancement équitable des files d'attente, soit les débits maximum sont explicitement fournis à chaque source en fonction des demandes. Le principe de base est de classer les n demandes $\{d_1, ..., d_n\}$, traversant un même lien l (de capacité $c(l)$), et émanant de différentes sources, par débit croissant. Notons $\{r_1, ..., r_n\}$, les débits accordés à chacune de ces sources. Récursivement, il s'agit de partager équitablement le volume restant $\alpha = c(l) - \sum_{j=1}^{i-1} r_j$ sur les $n - i$ demandes non encore satisfaites ou saturées. Ainsi, si $d_i \leq c(l)/n$ alors $r_i = d_i$ sinon $r_i = min(d_i, (\alpha/(n-i)))$. Une telle approche est nécessairement centralisée dans le sens où l'ensemble des demandes doit être connu. En pratique, il est possible d'associer l'état d'un lien au débit maximal *max min fair share* qu'il peut accorder à une nouvelle session (source,destination) le traversant en fonction des sessions qu'il supporte déjà et en s'adaptant à l'arrivée et au départ des sessions.
Les auteurs de (MSZ96) proposent un protocole de routage par la source permettant une approximation du problème *max min fair share*. Elle peut s'appliquer à un routage par commutation d'étiquette explicite en utilisant une métrique adaptative basée sur l'estimation du débit disponible (ici au sens *max min fair share*) que peut obtenir une session (source,destination) sur chacun des liens traversés. En particulier, ils proposent un protocole de routage multichemins vérifiant la propriété de l'équité à plusieurs niveaux (*multi level max min fair share*). Il s'agit d'attribuer à chaque chemin d'une session (source,destination) un niveau de priorité. Ce classement permet de récursivement distribuer l'excès de

demande non placé sur un chemin de priorité élevée vers un chemin parallèle de plus petite priorité.
Il peut également s'avérer intéressant d'utiliser des variantes multichemins des algorithmes WSP ou SWP pour définir un ensemble de meilleurs chemins dont la bande passante cumulative est maximale. L'objectif est d'obtenir un flot maximum sur les chemins les plus courts en vérifiant par exemple une contrainte sur le délai maximal d'un chemin. L'algorithme itératif de Ford et Fulkerson (FF62) calcule le flot maximal (une coupe minimale, *min flow cut*) entre une source et un puits avec une seul métrique concave : la capacité des liens. La proposition de Rao et Batsell (RB98) est inspirée de cet algorithme et permet de déterminer un ensemble de multiflot optimal en termes de délais d'acheminement entre une source et un puits. Leur contribution considère la possibilité de réserver plusieurs chemins en fonction de la demande q de la source. Il s'agit de déterminer, à chaque itération, un ensemble de flots compatibles augmentant le flot maximal. L'itération est initiée avec le chemin de plus petit délai de propagation et s'arrête dès lors que le délai de propagation d'un des deux ensembles de flots (celui calculé lors de l'itération courante et la somme de ceux calculés lors des étapes précédentes) est supérieur au délai d'acheminement de l'autre. Le délai d'acheminement d'un chemin P (ou d'un multichemin correspondant à un ensemble de flots) est ici caractérisé par l'expression $\frac{q}{c(P)} + d(P)$ où $c(P)$ désigne la capacité du lien offrant le moins de bande passante sur le chemin P (indicateur concave) et $d(P)$ désigne le délai de propagation de P (indicateur additif). Ainsi, leur méthode permet de déterminer, en un temps d'éxecution polynomial, l'ensemble de chemins suffisant pour transmettre q unités de trafic avec un délai d'acheminement minimal pour une session (source,destination) donnée.
De manière générale, l'objectif des méthodes algorithmiques à la source est la résolution par approximation d'un problème NP-complet de flot multicommodité optimal. L'idée intuitive des heuristiques basées sur la résolution de ces problèmes est la modification incrémentale du coût des liens appartenant au SPT calculé sur un graphe successivement réduit.
L'algorithme *Cycle Removed Algorithm* ((NZ01), CRA) propose par exemple d'utiliser de manière incrémentale l'algorithme de Dijkstra. La première occurence calcule les chemins de meilleur coût selon la métrique additive choisie. Puis, à chaque occurence consécutive, les liens de plus petite bande passante sur les meilleurs chemins calculés précédemment sont supprimés du graphe. Cette opération concerne aussi chaque lien appartenant à ces chemins : la bande passante du lien correspondant au goulet d'étranglement (*bottleneck link*) est soustraite de leur capacité. Le facteur de terminaison d'un tel algorithme correspond soit à l'obtention d'une bande passante cumulative suffisante entre le nœud racine et chaque autre nœud du graphe, soit au partitionnement du graphe. Cette approche est étendue dans (NZD04).
D'autres heuristiques, comme *Discount Shortest Path Algorithm* ((Pal01), DSPA), utilisent également une méthode incrémentale. A chaque occurence de l'algorithme de calcul des meilleurs chemins, le coût des arcs utilisés lors de l'itération précédente est arbitrairement augmenté afin de favoriser l'utilisation d'arcs distincts lors des occurences suivantes. La complexité de ce type d'algorithmes est liée aux nombres d'itérations de l'algorithme de Dijkstra. Ces algorithmes itératifs sont utilisés pour calculer des multi-meilleurs chemins disjoints en bande passante.
Il existe également une famille d'algorithmes pour résoudre le problème des K meilleurs chemins. Certaines solutions sont typiquement orientées *réseaux et flots* comme l'algorithme de calcul des K

meilleurs chemins disjoints en arcs ou en liens proposé par Suurballe (Suu74). D'autres propositions (Che94), considèrant aussi la capacité des liens, résolvent des problèmes de type *Quickest Path* selon une quantité donnée de trafic à transmettre.
Dans leurs travaux, Guérin et Orda (GO02) formulent le problème du chemin optimal selon le nombre de sauts (*All Hops Optimal Paths*, AHOP). Il s'agit d'identifier pour chaque nombre de sauts possibles, le chemin de coût minimal liant une source et une destination. Notons H, le nombre de sauts maximal évalués par le problème AHOP : $H < |N|$ si on ne considère pas les chemins contenant des cycles. La résolution de ce problème permet d'obtenir tous les meilleurs chemins vérifiant une contrainte sur le nombre de sauts maximum acceptable. Chacun de ces chemins peut correspondre à une requête individuelle (une FEC par exemple). Les auteurs démontrent que la borne minimale de la complexité d'une telle approche est en général (par exemple avec une métrique additive) en $O(|N|^3)$. Avec l'algorithme de Bellman-Ford, qui procèdent par nombre de sauts croissants, on obtient plus précisément une complexité de $O(H \times |E|)$. Néanmoins, avec une métrique concave sur la bande passante (lien de capacité minimale, *bottleneck*), il est possible d'atteindre une limite de complexité inférieure : $O(\frac{|N|^3}{log|N|})$. Les auteurs proposent un algorithme permettant d'atteindre cette borne théorique mais admettent, qu'en pratique, l'algorithme de Bellman Ford est le plus adapté grâce à sa simplicité.
Si on néglige les contraintes liées à la bande passante, il est possible de calculer l'ensemble des K meilleurs chemins. L'algorithme *Finding the K Shortest Paths* (Epp94) proposé par Eppstein présente une complexité de $|E|+|N|log|N|+K\times|N|$ qui semble être la meilleure borne connue. Les chemins sont extraits et stockés sous une forme représentative non explicite (le chemin n'est pas enregistré comme une suite de liens ou de routeurs mais est uniquement representé par son coût). L'algorithme présenté permet également de déterminer l'ensemble des meilleurs chemins dont le coût est inférieur à un seuil donné avec le même facteur de complexité. Cet algorithme autorise cependant la présence de circuits ce qui n'est pas adapté aux problématiques orientées réseau de commutation de type IP. Il pourrait néanmoins s'appliquer sur un graphe acyclique orienté (*directed acyclic graph*, DAG) revenant ainsi à un problème équivalent.
Les premières contributions de ce domaine de recherche datent de 1957, on peut en citer plusieurs : Minieka (Min74), Lawler (Law77), Yen (Yen71), Horne (Hor80) et Shier (Shi76) ou plus récemment Brander et al. (BS95) et Martins et al. (MP03).
En pratique, il est nécessaire de déterminer, en fonction de la puissance de calcul disponible ou des objectifs escomptés, le nombre K de chemins à calculer. Il peut s'agir de limites physiques comme le temps de calcul accordé à l'algorithme de calcul ou de limite en termes de signalisation et de maintenance des K routes. Certaines études expérimentales comme (JV06), ou (RSB$^+$03) pour des réseaux mobiles ad-hoc, exposent le problème du nombre de chemins nécessaires en fonction des critères d'équilibrage de charge souhaités. Par exemple, pour des critères de protection, un seul chemin altenatif totalement disjoint du premier (en liens et en routeurs) peut suffire pour prévenir les pannes de liens et de routeurs sur le chemin primaire (protection 1 :1). En termes de répartition de charge, la configuration du paramètre K est plus difficile à évaluer. Cela dépend des connaissances a priori sur le trafic. Ces informations, généralement obtenues avec des outils de métrologie, peuvent être stockées sous la forme d'une série de

matrices. Ces matrices de trafic servent à dimensionner le réseau et permettent de configurer le nombre de chemins nécessaires. Dans un contexte purement réactif, certains chemins peuvent être calculés et positionnés dynamiquement en fonction de l'évolution du trafic mesuré par des estimateurs *online*.
Dans l'absolu, si l'on néglige le coût de positionnement et de maintenance des chemins, plus K est grand, plus le reroutage ou l'équilibrage de charge peuvent être fiables quels que soient les objectifs. La vrai problématique est alors l'utilisation de ces multichemins, comment et à quel moment les activer[10] ?

1.3.3.2 Discussion

Le principal avantage des techniques de routage à la source est la facilité de mise en place et la diversité des routes. Le problème des boucles de routage est hors sujet dans la mesure où le routage est explicite de bout en bout. Pour des aspects de protection notamment, il est primordial que les chemins de secours (*backup paths*) garantissent les mêmes contraintes en bande passante que les chemins primaires et que ceux-ci soient complètement disjoints entre eux. Le calcul des chemins à la source et leur positionnement explicite permet de vérifier de manière exacte ce type de contraintes. De même que pour les métriques, les contraintes peuvent être variées (pertes, probabilité de blocage, délais, etc) et se superposer sans autre limite que la puissance de calcul de l'Ingress LSR. Le paragraphe 1.3.3.1 témoigne de la diversité des méthodes pouvant être utilisée.
L'extensibilité des méthodes utilisant des routes étiquettées est cependant limitée pour mettre en œuvre une réactivité purement locale. D'une part, dans un contexte de positionnement explicite piloté par la source, seul le routeur d'entrée est capable de modifier les labels attribués aux FEC, et le temps de réaction dépend alors du délai de notification entre le routeur détectant un problème et le routeur d'entrée. Par conséquent, que ce soit pour des objectifs de protection/restauration en cas de panne ou pour équilibrer la charge, seul le routeur d'entrée peut répartir le trafic à destination du routeur de sortie.
La répartition de charge est effectuée par la source et nécessite des outils de métrologie et/ou des connaissances a priori sur le trafic. Pae ailleurs, dans le cadre d'un routage par labelisation généralisé à l'ensemble du réseau, le passage à l'échelle implique un nombre de liaisons logiques très élevé dont le coût de maintenance serait trop important en terme de signalisation. Si ce type de méthode assure la protection (au sens panne ou congestion) du réseau lien par lien, la réaction peut être accélérée, mais l'extensibilité paraît alors limitée.
Ainsi, le nombre de paires de routeurs d'entrée/sortie déployant ce type de routage est limité. En pratique, dans le contexte d'un déploiement réaliste, on considère généralement que ces méthodes s'appliquent à certains routeurs de bordure de domaine. Cette limitation peut pénaliser la diversité des chemins alors qu'il s'agit d'un enjeu important pour l'ingénierie de trafic à l'intérieur du domaine. En admettant que chaque LSR d'un domaine MPLS constitué de N nœuds positionne K chemins vers tout autre LSR de ce domaine, la signalisation concernerait $K \times N \times N - 1$ chemins (*N square problem*, voir (WWZ01)).
Nous avons évoqué dans l'introduction de cette partie, la possibilité de mettre en place des LSP de

[10] Ces aspects seront traités dans le chapitre *Fiabilité et équilibrage de charge*.

manière relâchée. Dans ce cas, le domaine MPLS peut profiter de la diversité de chemins et de la flexibilité de routage inhérente aux protocoles de routage au saut par saut sous-jacents. Les labels sont positionnés en fonction des dispositions multichemins de chaque routeur au niveau du routage IGP. Par exemple, lorsque ECMP est déployé, et si un routeur ECMP de cœur du domaine dispose de plusieurs chemins de coût égaux, il peut en théorie faire le choix de l'interface de sortie (et du label associé) de lui-même. A la réception d'un paquet étiquetté, le routeur ECMP est capable de partager la charge localement. Les décisions sont donc potentiellement distribuées et le routage est flexible dans le sens où l'Ingress LSR ne détermine pas de manière stricte l'acheminement de bout en bout jusqu'au routeur de sortie. Plus généralement, avec un processus de validation moins limité qu'ECMP, le routage MPLS à commutation relâchée peut donc en théorie être combiné à un routage multichemins saut par saut pour bénéficier de ces propriétés. Si l'ingress LSR désire satisfaire des critères de QoS précis, comme des contraintes en bande passante, les contraintes peuvent être déterminées à la source. Celle-ci peut refuser, par le biais du protocole de signalisation utilisé, l'acheminement via certains liens activés par le protocole IGP.
MPLS n'est donc pas nécessairement une technologie orthogonale aux protocoles de multiroutage disposant de tables de routage IGP calculées de manière distribuée et réciproquement.

1.4 Routage interdomaine et problématique multichemins

1.4.1 Routage entre domaines

Les protocoles de routage interdomaine ont pour but d'acheminer le trafic entre les systèmes autonomes constituant l'Internet (*Autonomous System*, AS). Les protocoles IGP décrits dans les parties précédentes ne sont pas adaptés au graphe représentant les interconnexions entre AS.

1.4.1.1 Généralités

Les relations entre AS peuvent être de nature complexe. En simplifiant, on distingue deux types de relations : soit un AS est le client d'un autre AS fournisseur, soit ils profitent d'une relation de type pair à pair (*peering*). Dans le premier cas, il s'agit d'un rapport de nature payante (type fournisseur d'accès Internet, noté FAI ou ISP). Dans le second cas, ce sont des accords commerciaux fondés sur l'échange. Cette relation peut être non payante entre FAI équivalents en terme de dimensions.
Le trafic transitant entre AS est soumis à certaines contraintes : les relations ne sont pas nécessairement symétriques ou transitives.

Le graphe des AS est hiérarchisé, on distingue notamment les opérateurs principaux qui sont au *sommet* (*tier one* tel que ATT, Sprint ou GlobalCrossing) et les *stub* AS qui sont les feuilles de ce graphe dans le sens ou leur liaison vers l'Internet passe généralement par un seul AS fournisseur [11]. Les échanges entre les différents AS de ce graphe *pyramidal* sont donc de type *plat* si des liaisons transversales existent, ou bien de type *montée/descente* sinon.

[11] La multidomiciliation est traitée dans la partie suivante.

Un AS de transit est constitué de *point of presence* (POPs) comportant des routeurs de bordure (*Gateway Router*, GWR). Les GWR font office de passerelle de peering vers les pairs et de relais pour les clients.
En termes de routage, les AS de transit diffusent des annonces de large préfixes d'adresses Internet destinés au trafic de transit, alors que les stub AS propagent des annonces de préfixes IP plus petits uniquement pour le trafic qui leur est destiné. Ces annonces permettent de définir la politique de routage choisie indépendamment par chaque AS. Un AS de transit décidera notamment de sélectionner tel ou tel AS voisin en coordonnant ses décisions avec le routage interne pour le trafic en aval et d'influencer le choix de ses AS en amont pour recevoir ou refuser certains type de flux. Techniquement, deux AS sont connectés par des routeurs qui leur sont internes, ainsi la fiabilité de la liaison point à point est respectivement limitée au lien sortant. Ces points d'échanges sont appelés des GIX (*global Internet exchange*) ou NAP (*Network Access Point*, l'équivalent européen du GIX). Les GIX/NAP locaux permettent les échanges à plats entre AS plutôt qu'une *montée/descente* par le biais d'un AS d'un niveau supérieur. Les annonces de préfixes diffusées par les GWR sont utilisées pour construire la FIB vers les adresses externes. Le routage est alors déterminé en fonction du plus long préfixe correspondant à l'adresse destination. La taille de la FIB est proportionnelle au nombre de préfixes enregistrés.

1.4.1.2 Border gateway Protocol, BGP

BGP est le protocole interdomaine de référence (voir le RFC 1771 (RL95)). Il utilise la classification de routage interdomaine (*Classless interDomain routing*, CIDR) défini dans les RFCs 1518 (RL93) et 1519 (FLYV93). L'allocation des blocs d'adresses est préfixée et hiérarchique. De cette manière, les tables de commutations ne sont pas directement relative aux nombres d'adresses IP. L'internet est découpé en *super réseaux* de taille 2^n, n étant le complémentaire sur 32 de la taille du préfixe. L'objectif de BGP est la diffusion de l'existence et de l'accessibilité des préfixes externes (les adresses IP sont englobées dans un préfixe CIDR) et le contrôle de la qualité et de la validité des routes. Il s'agit de déterminer dynamiquement les meilleurs routes sans boucles en fonction de critères globaux (l'ensemble de la route sur le graphe des AS) et locaux à l'AS (le segment de route appartenant au graphe de l'AS). La signalisation entre voisins BGP peut être de deux types :

- entre routeurs GWR internes à l'AS : *interior BGP* (iBGP).
- entre routeurs GWR appartenant à des AS différents : *exterior BGP* (eBGP).

Le contenu d'une annonce BGP contient un certain nombre d'attributs permettant de sélectionner la meilleure route et d'influencer le comportement des routeurs en aval. Chaque routeur BGP maintient une base des annonces acceptées (Adj-RIB). Il propage à ses voisins, en fonction de la politique de filtrage et de transit, la meilleure annonce retenue pour un préfixe donné (afin d'assurer l'absence de boucle de routage interdomaine). Le protocole iBGP assure la une vision topologique cohérente pour tous les routeurs du domaine et nécessite une connexion *full-mesh* entre routeurs BGP de l'AS. Pour éviter les boucles, les annonces reçues par iBGP ne sont pas propagées en interne. Afin de limiter les problèmes de complexité liés au caractère *full-mesh* des connexions iBGP, des solutions tel que les *route*

reflectors et les *confederations* existent (voir par exemple (BUQ04)). Le principe général est la prise en charge des sessions par un seul routeur et la subdivisions en sous aires de routage.
Les routes sélectionnées par BGP doivent être redistribuées aux routeurs IGP et réciproquement, or si les métriques utilisées ne sont pas les mêmes entre GWR la comparaison n'a pas de sens. La coopération entre routage inter et intradomaine repose donc sur la redistribution des meilleurs routes annoncées.
Lors de la réception d'une annonce de préfixe Pr, le fonctionnement d'un routeur BGP est décomposé en trois étapes :

- filtrage des annonces indésirables selon la politique définie.
- insertion de l'annonce dans la base Adj-RIB et calcul de la meilleure annonce vers Pr.
- si la meilleure route a changé ou à la réception d'un nouveau préfixe Pr :
  - la FIB est mise à jour et les routes externes sont éventuellement redistribuées sur le routage IGP. Pour cela, les routeurs de bordures communiquent entre eux au moyen du protocole iBGP et peuvent décider ou non d'injecter l'annonce dans le routage IGP. Si c'est le cas, les routeurs de cœur doivent choisir, en fonction de critères locaux, un des routeurs de bordure ayant enclenché une redistribution de la commutation vers Pr.
  - en fonction des filtres de routage de l'AS (selon l'influence qu'il veut exercer sur ses voisins AS) l'information peut être ou non diffusée en externe (eBGP).

La phase de calcul de la meilleure route dépend de l'ordre des attributs spécifiés dans la métrique composite. Ces attributs sont classés par ordre d'importance pour déterminer un unique meilleur chemin (*tie breaking rule*). Pour établir cet ordonnacement, des attributs tels que la préférence locale, la taille du chemin inter-AS, l'origine de la première annonce, le discriminant multi-sortie, le plus proche NH de sortie sont utilisés.
L'émission et la propagation d'une annonce par un routeur BGP tient compte de ces attributs, mais l'ordre de traitement n'est pas spécifiquement défini. Ainsi, un routeur de bordure peut configurer ses préférences et influencer le comportement de ses voisins pour modifier le trafic de transit en amont et en aval. Nous allons rapidement décrire la nature des attributs cités pour illustrer le fonctionnement de BGP.

- L'attribut de chemin inter-AS, l'*AS-Path* indique la suite d'AS traversé par l'annonce. BGP est un protocole à *vecteurs de chemins* permettant de détecter les boucles en refusant simplement une annonce dont l'AS-Path contient son propre identifiant d'AS. Le choix se fait sur le nombre d'AS à traverser pour atteindre le préfixe.
- L'attribut *Origin* indique comment le premier AS de l'AS-Path connaît le préfixe Pr. Il peut provenir du routage intradomaine, du routage interdomaine ou être inconnu (l'ordre de préférence suit cette loi).
- La préférence locale *Local Preference* a une signification locale (pour iBGP). Cet attribut est utilisé pour permettre de sélectionner la meilleure sortie en influençant la décision.
- Le discriminant multi-sortie, *Multi Exit Discriminator* (MED), permet d'influencer cette fois par voie inter-AS (eBGP), le choix de la porte d'entrée dans l'AS.
- L'attribut de plus proche prochain saut, *NextHop* : il s'agit pour les routeurs internes à l'AS de

choisir le prochain saut IGP de coût minimal vers un des routeurs eBGP ayant reçu l'annonce du préfixe Pr. Cet attribut provoque un routage de type *patate chaude* (*Hot potatoe routing*). Le routage interdomaine n'est pas symétrique et l'objectif est généralement de se débarasser au plus vite du trafic traversant l'AS.

L'exemple donné sur la figure 1.7 fait intervenir trois domaines distincts : les systèmes autonomes numérotés 1, 2 et 3. Les données émises par l'hôte A doivent traverser ces trois AS pour communiquer avec son homologue B. L'adresse IP de B est contenue dans le préfixe correspondant au domaine 3. Ce préfixe est annoncé par les routeurs de bordure de ce domaine. De proche en proche, l'annonce de ce préfixe atteind l'AS 1 et celui-ci peut redistribuer à ses routeurs internes l'information nécessaire au routage vers le préfixe de l'AS 3.

Le routage interdomaine est basé sur des accords commerciaux et/ou politiques. Alors que le routage IGP présente généralement des objectifs d'optimisation tels que la réduction des délais d'acheminement, un protocole comme BGP n'est pas conçu pour favoriser les performances en terme de latence. Avec BGP, ce n'est pas nécessairement la meilleure route qui est utilisée pour relier deux équipements appartenant à des domaines pairs, qu'ils soient ou non concurrents. Un AS n'a pas d'interêt individuel à relayer le trafic interdomaine de ses pairs, il peut donc influencer le routage externe pour supporter le moins de charge interdomaine possible et rejeter au plus vite le trafic qui lui est confié (seuls des compensations financières entre pairs favorise un comportement moins *égoïste*).

Les mécanismes asymétriques type *hot potatoe* vont, par définition, à l'encontre d'une commutation optimale. Par ailleurs, le nombre de sauts d'AS n'est pas une indication précise sur la latence réelle de la route. Le routage interdomaine est peu enclin à un calcul de routes optimales comme c'est le cas en IGP.

Le routage multichemins interdomaine, au sens plusieurs sorties annoncées par des routeurs de bordures différents pour un préfixe donné, est en théorie possible avec BGP sur la base de l'égalité et de l'équité. Il est impératif que la longueur des multichemins en nombres d'AS soit égale. De manière générale, de nombreux attributs doivent être identiques et cela restreint le déploiement du multiroutage interdomaine.

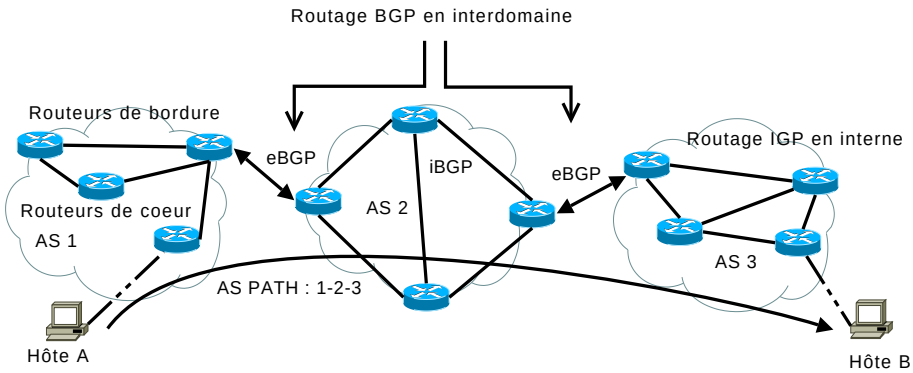


FIG. 1.7 – Routage interdomaine : BGP

Les recherches présentées dans l'article (TGRR05) fournissent une méthode de *tie breaking* pour choisir un routeur de bordure permettant un acheminement efficace vers un préfixe donné. Les mesures réalisées dans les travaux (QUP+03) démontrent qu'entre 30% et 50% des décisions de sélection des routes de BGP sur Internet sont dues aux règles de *tie breaking*. Leurs recherches mettent également en avant la difficulté pour un AS de prédire et donc de contrôler le trafic entrant.
Les deux parties suivantes décrivent les principaux mécanismes proposés dans la littérature permettant de contourner les restrictions dues à BGP. L'objectif est de généraliser le multiroutage interdomaine afin d'accroître la fiabilité et les performances du réseau.

1.4.2 Multidomiciliation et réseaux couvrant

Les limitations en terme de fiabilité dues à BGP ont suscité l'émergence de plusieurs solutions permettant de contourner ces restrictions. Nous énumérerons dans cette partie les principales propositions adaptées au routage multichemins interdomaine.

1.4.2.1 Multihoming

Le *Multihoming* se définit par la capacité d'un nœud (un routeur, un hôte ou plus généralement un domaine) à disposer de plusieurs interfaces de communication dont les extrémités sortantes sont logées chez des fournisseurs d'accès différents (des ISP distincts). Ainsi, un routeur domicilié chez plusieurs fournisseurs favorise le déploiement d'un routage multichemins via plusieurs ISP en fonction de l'interface sortante choisie.
La multidomicialition permet de déployer un routage interdomaine multichemins au sens où le choix du premier saut du nœud multidomicilié pour un préfixe donné permet d'évaluer et de mettre en compétition plusieurs routes inter-AS. Ces routes pouvant éventuellement se rejoindre par la suite sur un AS commun (*Path merging*). Le contrôle de la commutation multiple n'est donc pas global : seul le premier ISP peut être choisi.

La figure 1.8 donne un exemple représentatif de la multidomiciliation. Au sens large, un AS est multidomicilié si il possède une connectivé via plusieurs ISP (ici avec les IPS 1, 2 et 3). Un routeur est multidomicilié lorsqu'il possède plusieurs interfaces distinctes vers des ISP différents (ISP 1 et 2). Sur l'exemple, on observe que l'hôte source bénéficie de trois choix pour atteindre l'hôte situé dans l'AS correspondant au préfixe destination. La multidomiciliation d'un hôte -non représentée ici- est le fait que l'équipement terminal soit directement raccordé à des FAI différents.
Le multihoming introduit une problématique liée au chemin retour. La symétrie n'est pas garantie : un flux envoyé sur une route donnée via l'ISP choisi n'implique pas que la réponse soit elle-aussi transmise en retour via le même ISP.
L'évaluation de performance réalisée dans l'article (AMS+03) mesure par émulation les bénéfices tangibles que la multidomicialiation peut offrir. Leurs ensembles de données ont été prélevés sur des serveurs appartenant au réseau de distribution de contenu *Akamai* et concernent deux types de trafic : un trafic orienté entreprise et un trafic orienté fournisseur de contenu.

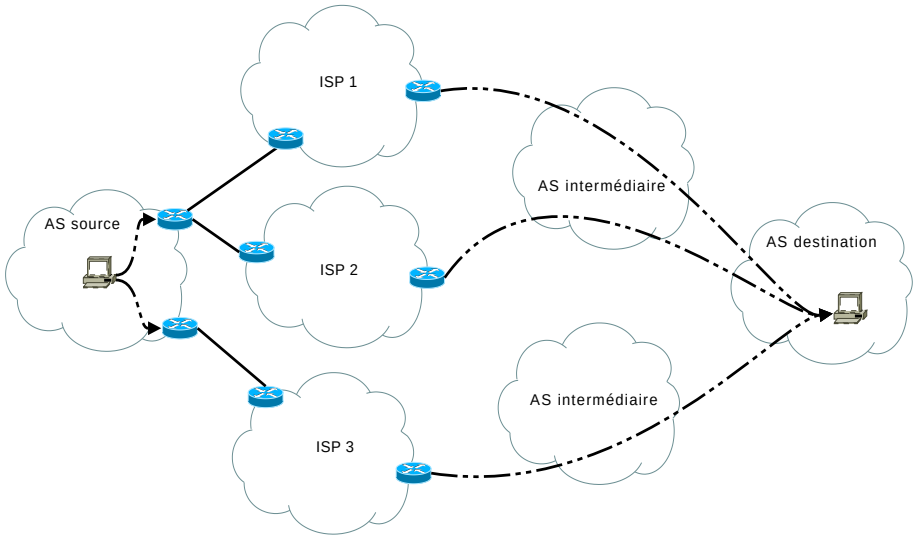


FIG. 1.8 – Multidomiciliation de routeur et de domaine

Dans leur modèle d'analyse, un abonné peut employer jusqu'à k ISP en utilisant, à un instant donné, celui qui semble proposer la meilleure latence. Les performances d'une route sont mesurées par k nœuds moniteurs connectés chacun à un seul ISP (celui-ci communique à des serveurs Akamai placés dans de nombreuses aires métropolitaine). Les auteurs affirment qu'à partir de $k = 2$, et dans une moindre mesure pour $2 \leq k \leq 4$, les performances en terme de latence peuvent être considérablement améliorées. Le choix des k ISP ne doit pas seulement tenir compte des performances individuelles de chacun mais surtout de leur complémentarité pour distribuer la charge. Leurs travaux mettent en évidence l'importance du choix stratégique des ISP pour que les routes alternatives se recouvrent le moins possible. Le choix du meilleur ISP à un instant donné, doit se baser sur les mesures obtenues lors des périodes d'analyses les plus récentes.

Goldenberg et al. proposent dans (GQX+04) des algorithmes *offline* et *online* en fonction des connaissances a priori sur les demandes de trafic. Leur procédures sont basées sur l'optimisation du coût financier de la multidomiciliation. Il s'agit d'évaluer le montant à payer par l'utilisateur (une entreprise ou un stub ISP) à chacun des ISP fournisseurs selon le modèle ordonné du p^e pourcentile. La quantité de trafic à payer est estimée à partir du volume de charge mesuré sur le q^e intervalle (le rang q correspondant au pourcentage p est $q = p/100 * I$, avec I désignant le nombre d'intervalles). L'objectif est de générer une distribution optimale des flux sur les k ISP pour minimiser le montant que le client aura à payer.

La distribution est fractionnelle si un même flux peut être acheminé via plusieurs ISP, elle est entière sinon. Dans les deux cas, il s'agit d'abord de déterminer la borne minimale de la somme des volumes de charge correspondante au q^e intervalle sur chacun des k ISP : celle ci est égale au volume de charge globale pour le $1 - \sum_{\forall k} (1 - p_k)$ pourcentile (p_k désigne le pourcentage choisi par le k^e ISP). Ensuite, il faut dynamiquement déterminer les volumes de charge minimisant le montant satisfaisant la borne minimale. Enfin, il s'agit de distribuer le trafic en fonction des volumes de charge déterminés par

ISP en considérant le nombre d'intervalles déjà classés au-dessus et en-dessous du volume de charge précédemment calculé par ISP.

1.4.2.2 Overlay networks

Un réseau overlay est un réseau de nœuds virtuels reliés par des liens logiques déployés au-dessus du réseau physique. La virtualisation des liaisons permet de proposer des services supplémentaires pour diversifier le routage. Internet est une forme de réseau d'overlay composé d'entités hétérogènes : les AS. Les domaines sont assimilables à des nœuds virtuels et les accords commerciaux les reliant constituent les liens logiques.
La notion de réseau couvrant s'adapte aussi à une échelle plus petite si on ajoute une couche d'abstraction au-dessus des routeurs pour virtualiser les connexions. Considérons par exemple le cas d'un ensemble de routeurs liés par un groupe de *switches* de niveau 2, l'utilisation d'un réseau d'overlay au-dessus de cette structure physique permet de configurer un réseau *full-mesh* au niveau réseau et donc au niveau du routage. La figure 1.9 illustre les bénéfices en termes de diversité de chemins qu'apporte le routage sur des réseaux overlay. Un chemin indirect désigne un chemin formé d'au moins deux indirections d'overlay successives. Avec le routage BGP, l'hôte source ne dispose que d'un seul chemin pour atteindre l'équipement situé dans le domaine annoncant le préfixe contenant la destination. En cas de panne de lien, le temps de convergence de BGP conditionne le temps de restauration. Le routage logique permet de contourner la panne si un chemin d'overlay adéquat existe. Le routage entre nœuds virtuels utilise en général la politique de routage interdomaine sous-jacente. Un chemin d'overlay est composé de plusieurs sous-segments de chemins BGP.
La proposition *Resiliant Overlay Network* ((ABKM01), RON) développée par Andersen et al. augmente les performances des protocoles de routage à grande échelle comme BGP : les nœuds virtuels sont reliés par un graphe logique complet au-dessus des liaisons physiques. Cette virtualisation permet d'obtenir des latences plus faibles entre les nœuds participants et assure un reroutage rapide en cas de panne. RON utilise un protocole *sonde* entre l'ensemble des nœuds participants pour contrôler l'état et la qualité des liaisons logiques. Ces mesures doivent être réalisées à une fréquence suffisamment élevée pour déterminer le meilleur chemin à un instant donné.
L'article (RKSB06) propose une évaluation des gains de performance apportés par les réseaux d'overlay avec des ensembles de données obtenus sur des réseaux IP commerciaux. Leurs travaux mettent en évidence que le choix des chemins inter-RON est surtout important pour le gain de performance en terme de latence alors qu'une technique de sélection aléatoire de chemins peut suffire pour fournir une couverture suffisante pour la protection. L'évolution des moyennes de latence obtenues sur des périodes de quelques minutes ne sont pas déterminantes pour le classement des K meilleurs chemins. Pour améliorer les délais d'acheminement, l'utilisation simultanée de 2 chemins concurents semble suffisante pour obtenir une latence proche de l'optimum si ces chemins sont analysés avec une fréquence de rafraîchissement d'une dizaine de minutes (l'indicateur de performance est le RTT).
Savage et al. proposent dans (SAA+99) une architecture analogue : *Detour*. Chaque AS dispose d'un nœud de routage virtuel : un routeur de bordure dont le rôle est d'agréger les flux sortants de l'AS.

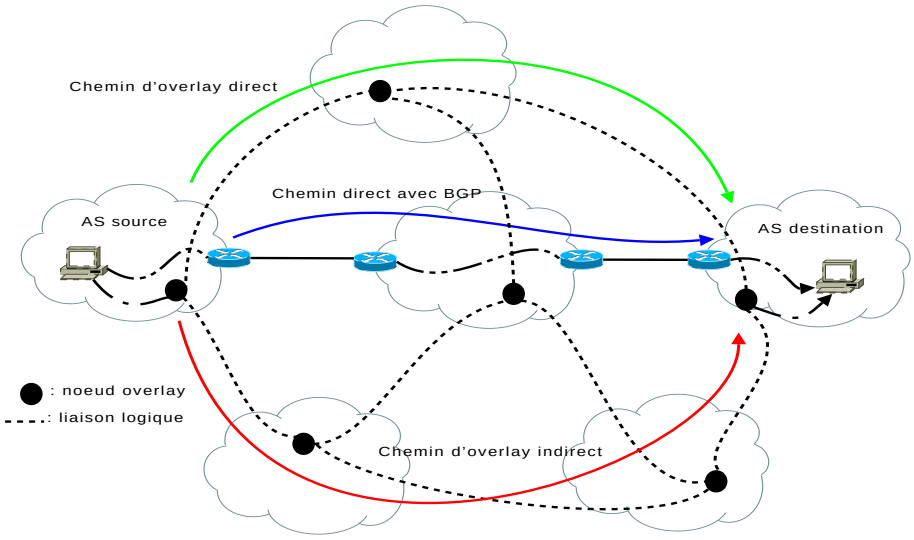


FIG. 1.9 – Réseau couvrant en interdomaine

Chacun de ces routeurs virtuels est lié au moyen de connexions logiques. De même que pour RON, les avantages potentiels sont multiples : fiabilité des connexions, diminution de la latence, augmentation du débit et classification du trafic. L'utilisation d'une notification des ressources résiduelles permet d'obtenir un routage adaptatif sans nécessairement s'exposer au problème de l'instabilité des routes (Bre95).

D'autres travaux comme ceux de Li et Mohapatra (LM04) (QoS-RON) généralisent cette démarche en définissant une (ou plusieurs) entité(s) stratégique(s) de virtualisation par AS : les *Overlay Broker* (OB). Ils proposent plusieurs algorithmes de sélection de chemins entre OB pour améliorer la qualité de service de bout en bout avec des métriques composites (bande passante et capacité de calcul disponibles). Les OB sont organisés en différent niveau de *clustering* pour créer une structure hierarchisée permettant de prendre en compte les problèmes d'extensibilités inhérents aux méthodes de virtualisation à grande échelle.

Pelsser et Bonaventure (PB03) proposent d'étendre le protocole de signalisation RSVP-TE pour renforcer la robustesse des communications interdomaine. Le déploiement de tunnels LSP est équivalent à la mise en place de liaisons logiques entre des couples de nœuds de routage virtuels (dans ce cas il s'agit des Ingress/Egress LSR). Une des difficultés liée au passage à l'échelle est la confidentialité architecturale que les ISP souhaitent généralement conserver.

Contrairement au LSP intradomaine, l'établissement de LSP interdomaine peut se faire sur la base d'un préfixe plutôt qu'avec une adresse destination IP précise. Le message *Path* est retransmis sur la base du chemin explicite (ou relâché) en fonction du préfixe jusqu'à ce qu'un LSR récepteur appartienne au domaine. Les auteurs proposent de placer la complexité en calcul sur les routeurs de bordures des AS (les ASBRs). A l'entrée de chaque AS, l'ASBR calcule le LSP interdomaine jusqu'au routeur de sortie correspondant à l'AS en amont. L'ASBR complète alors l'ERO en fonction de son choix. Le LSR d'entrée du LSP spécifie dans l'ERO les adresses IP de son domaine et éventuellement un chemin

d'AS. En intradomaine, le champ RRO permet d'identifier le chemin suivi par le LSP ce qui peut poser des problèmes liés à la politique de confidentialité des ISP. Par conséquent, les auteurs définissent une procédure d'agrégation de RRO. Le routeur situé en sortie de domaine doit supprimer du RRO les LSR parcourus et de les remplacer par l'identifiant de l'AS, l'adresse de l'ASBR d'entrée et la sienne. Ces informations suffisent pour déterminer les critères de séparation entre les LSP destinés à la protection. En interdomaine, la protection de bout en bout est complexe à mettre en œuvre dans la mesure où chaque opérateur local peut avoir ses propres préférences. Les auteurs envisagent des solutions de protection segmentées selon la situation des liens et des routeurs (liaisons internes ou externes).
Leurs propositions prennent également en compte les liaisons groupées. Certains liens peuvent partager une infrastructure physique commune : leurs pannes sont potentiellement corrélées. On appelle un tel groupe de liens un *Shared Risk Link Group* (SRLG). Lorsqu'un lien appartenant à un SRLG subit une panne, tous les autres liens du même SRLG peuvent aussi être affectés. Une des principales difficultées est liée au moyen d'identication d'un SRLG. Celui-ci peut en effet être désigné via des identifiants différents entre les AS qui partagent cette infrastructure.

1.4.2.3 Signalisation, placement, combinaison et comparaison

Il est également possible de définir un environnement de routage multichemins et explicite sans signalisation (en inter et intradomaine) en se basant sur une fonction d'identification globale d'un chemin à l'échelle considérée. L'architecture *BANANAS* (KKW+03) utilise la notion de *PathID* pour identifier une route (inter ou intra-AS) en opposition aux identifiants de labels définis localement sur le domaine. Le PathID est un champ de taille restreinte contenu dans l'en-tête IP d'un paquet. Sa signification globale permet de contourner l'utilisation d'une signalisation inhérente aux mécanismes à commutation d'étiquettes comme MPLS. Ce champ contient la somme des caractéritiques d'un chemin (identifiants des routeurs participants et des liens à traverser) condensée par application d'une fonction de hachage. Entre routeurs participants, la commutation utilise le plus long préfixe IP commun et le champ PathID du paquet. Le PathID est diminué de l'identifiant du routeur commutateur à chaque saut, tel que, de proche en proche, l'identifiant de hachage ne prenne plus en compte tous les routeurs déjà traversés. Le suffixe haché est noté *SuffixPathID* et correspond à l'ensemble des équipements restant à traverser. Sur les routeurs BANANAS, la phase de calcul/sélection des chemins est décomposée en deux étapes. Les routeurs participants calculent et trient par nombre de sauts les k meilleurs chemins de l'ensemble des routeurs BANANAS. La seconde phase vérifie l'absence de boucle de routage. La validation se fait de manière incrémentale sur le nombre de sauts des chemins.
Pour simplifier la commutation et la complexité de stockage des routeurs de cœur, les adresses IP correspondant aux liens sortants des routeurs à traverser sont indexées. Les complexités en calcul et en validation sont principalement concentrées sur les routeurs de bordure.
Pour le routage interdomaine, le passage à l'échelle est réalisé en déclinant le PathID en *E-PathID*. Cet identifiant permet de déterminer explicitement une suite d'AS à traverser.
Cette proposition permet de réduire l'*overhead* de signalisation mais pose le même problème que le routage explicite à la source. La complexité en calcul est proportionnelle au nombre de routeurs BA-

NANAS or la flexibilité du routage dépend du nombre de routeurs participants.
Les auteurs de (CMPS06) formalisent la problématique du placement des nœuds d'overlay à l'intérieur d'un AS pour minimiser l'intersection entre chemins primaires et alternatifs. Ils commencent par définir une variable probabiliste indiquant l'impact de la panne d'un lien sur un ensemble de chemin par couple (source, destination). Cette variable est étendue pour reflèter la qualité d'un chemin d'overlay avec un indicateur de *pénalité*. La mesure de pénalité évalue la manière dont la panne de chaque lien affecte l'ensemble des multichemins.
L'objectif est de placer k nœuds d'overlay dans le domaine en minimisant l'indicateur de pénalité pour l'ensemble des paires (source, destination). Les résultats obtenus par simulation tendent à montrer que l'utilisation de $10\%|N|$ de nœuds de relais permet de générer une solution proche de l'optimum.
Les auteurs de (ZDA07) proposent une combinaison entre multidomiciliation et réseau logique dans la perpective de maximiser les gains d'un opérateur logique (*Overlay Service Provider*, OSP). Un OSP doit être attractif en termes de coût financier et de performances pour attirer des clients. L'objectif de leur proposition est double, il s'agit de déterminer l'emplacement optimal des nœuds de routage appartenant à l'OSP et de choisir un ensemble d'ISP en aval de ce routeur. L'OSP est un réseau d'overlay constitué de nœuds de routage virtuel multidomiciliés (*Multihomed Overlay Network*, MON) et placés stratégiquement au niveau des GIXP.
Formellement, les auteurs définissent un problème d'optimisation maximisant les gains de l'OSP en placant N nœuds MON, chacun connecté au maximum à K ISP. Ce problème est soumis à une contrainte d'attractivité : l'OSP doit fournir des services de meilleur qualité que l'ISP natif pour une large proportion des flux. Les auteurs prouvent, par réduction à un problème de recouvrement par ensemble, que la résolution exacte de ce problème est NP-complet. Ils proposent quatre heuristiques : aléatoire (expérience témoin), orientée utilisateur (les N nœuds sont placés là où se trouve une majorité de clients), orientée trafic (plutôt que de considérer les utilisateurs, il s'agit de considérer les volumes échangés) et orientée performances (basée sur une métrique permettant de prendre en compte la qualité des chemins d'overlay). Les auteurs utilisent dans tous les cas une métrique statique : les délais de propagation.
Akella et al. comparent dans (APM+04) l'utilisation du multihoming et de l'overlay routing. Ils introduisent la notion de k-overlay pour désigner une multidomicialiation sur k ISP associée au routage logique d'overlay. Ils réalisent un ensemble de mesures, sur un sous-ensemble de paires de nœuds appartenant au réseau de distribution de contenu Akamai. Les indicateurs utilisés sont la latence (RTT), le débit et la disponibilité des chemins. Leurs résultats d'émulation indiquent que l'overlay routing semblent moins performant, en termes de latences et de débits mesurés, que le k-multihoming pour $k \geq 3$. Le routage k-overlay ne produit pas des gains de performances très significatifs par rapport au k-multihoming seul. En revanche, en terme de fiabilité, le routage k-overlay se révèle dans une majorité de cas plus performant que le 3-multihoming pour générer une couverture élevée.
Le gain substantiel généré par l'association avec l'overlay routing pourrait devenir négligeable si les politiques de peering entre ISP étaient moins *égoïstes*. La coopération entre ISP est en théorie possible en utilisant le principe de *late exit* (ou *cold potatoe routing*) qui permet de redistribuer la métrique MED apprise via eBGP en interne.

Une grande variété de propositions existe pour que l'acheminement des paquets en interdomaine puisse lever les restrictions monochemin liées à BGP. Dans le chapitre suivant, bien que davantage focalisés sur la diversité des chemins en intradomaine, nous étudierons différents modes de coopération entre ces deux formes de routage.
Le but de nos contributions est de favoriser la diversité des chemins. Pour cela, nous définirons un ensemble de règles et d'algorithmes permettant de d'utiliser des conditions moins strictes que celles définies dans ce chapitre. Diversité intra et interdomaine peuvent être combinées pour offrir au réseau une couverture efficace en améliorant sa fiabilité.
Dans le chapitre suivant, nous définirons un ensemble de procédures pour garantir un routage multichemins sans boucle au niveau routeur. L'objectif de notre proposition est de mettre en œuvre un routage multichemins tenant compte de l'interface entrante des paquets pour la commutation. Cette distinction permet d'utiliser des règles de validation proche d'une condition nécessaire au routage sans boucles. En effet, les règles statiques que nous avons présentées jusqu'ici sont suffisantes mais non nécessaires à la cohérence du routage.

Chapitre 2

Contributions : Recherche et validation des chemins

Notre proposition au routage multichemins se décompose en deux étapes : d'une part, chaque nœud calcule et sélectionne un ensemble de multichemins sous la forme de prochains sauts, d'autre part la composition de ces chemins est élaguée pour former des routes viables et cohérentes. Nos propositions s'inscrivent dans le contexte d'un routage saut par saut à état des liens. L'objectif des algorithmes que nous décrirons ici est la mise en œuvre d'une commutation disposant d'une diversité de chemins accrue par rapport aux méthodes de la littérature.
Il s'agit d'un avantage considérable dans la mesure où le nombre de prochains sauts activés de proche en proche et la faible intersection, en termes de liens, entre les chemins générés conditionnent directement les capacités en termes de protection et de flot maximal entre toutes les paires de nœuds. La commutation mis en œuvre par nos contributions dépend de l'interface d'entrée. Les tables de routage disposent de plusieurs prochains sauts dont l'utilisation peut être simultanée.
Les deux premières parties de ce chapitre décrivent la solution théorique envisagée pour le multiroutage en intradomaine. La dernière partie introduit les notions plus techniques liées aux différentes formes d'implémentations possibles. Nous y verrons notamment comment combiner routage multichemins inter et intradomaine pour accroître la diversité des chemins inter-AS.
Nos contributions sont basées sur l'existence sous-jacente d'un protocole de notification topologique par inondation des états des liens. Dans la suite, nous considérerons que nous pouvons tirer parti de la notification d'adjacence.
Le paragraphe 2.1 décrit la première étape de nos procédures. Celle-ci est inhérente au routage à états des liens et s'adapte au contexte multichemins saut par saut : chaque nœud calcule au moyen d'un algorithme de recherche opérationnelle spécifique un ou plusieurs chemins aux premiers sauts distincts vers chaque autre nœud du graphe. En pratique, tous les routeurs implémentant notre contribution déterminent un ensemble de sous-arbres dont ils sont la racine. Nous proposons un algorithme de recherche dans les graphes fondé sur celui de Dijkstra : *Dijkstra transverse* (DT). *Dijkstra transverse* calcule un ensemble de chemins aux coûts non nécessairement égaux d'un nœud racine vers chaque destination. L'algorithme DT permet de trouver au minimum un prochain saut alternatif vers chaque nœud du graphe si celui-ci est suffisamment maillé pour que le graphe ne soit pas partitionné en cas de rupture de lien (pour toute paire de nœuds appartenant à la même composante biconnexe).
La seconde étape consiste à éliminer les boucles de routage induites par la composition de chemins multiples aux coûts hétérogènes. Le principe de sous-optimalité n'est pas satisfait si les coûts calculés sont différents. Notre processus de validation, appliqué au graphe généré par DT, DT(p), a pour objectif d'activer uniquement les prochains sauts garantissant l'absence de boucle de routage au niveau routeur. Il élague, dans un rayon de p sauts au maximum, le graphe composé par adjacence distribuée des prochains sauts (ou *next hop* : NH) calculés par DT.

L'ensemble de nos procédures algorithmiques et protocolaires est fondé sur la décomposition du graphe acyclique orienté obtenu via l'algorithme de Dijkstra en sous-arbres des meilleurs chemins distincts. L'algorithme de Dijkstra partitionne les arcs du graphe en deux composantes relatives à une racine de calcul donnée, le nœud courant noté s. Selon leurs appartenances ou non à l'arbre des plus courts chemins enraciné sur s, certains arcs sont ignorés. Contrairement à cette distinction négligeant les arcs liant les sous-arbres formant des ensembles de meilleurs chemins, DT profite de ceux-ci pour bénéficier d'une diversité de chemins accrue.
Par ailleurs, le principe de sous-optimalité des meilleurs chemins est dépendant de la métrique choisie. Cela présente l'inconvénient d'utiliser systématiquement, entre toutes paires de nœuds du graphe, des sous-ensembles d'arcs faiblement valués. Certaines parties du graphe sont favorisées par le routage au détriment d'autres pouvant s'avérer précieuses lorsque le réseau présente une charge élevée et hétérogène.
Notre proposition permet de prendre en compte les liens a priori peu utilisés. Le nombre de chemins calculés ne préjuge pas de leur utilisation dans un contexte réel : ces problématiques sont liées à la nature du trafic et seront abordées dans le chapitre suivant. La principale contrainte pouvant représenter un obstacle est la complexité de l'algorithme de recherche opérationnelle et le temps nécessaire à l'activation des prochains sauts par le processus de validation.
La première partie de ce chapitre est consacrée à notre algorithme de recherche opérationnelle. A partir du paragraphe 2.2, nous introduirons l'ensemble des notions relatives à la distinction du trafic et nécessaires pour définir notre processus de validation.

2.1 Dijkstra transverse (DT)

2.1.1 Principe et nomenclature

Dans le cadre d'un routage à états des liens, l'algorithme de Dijkstra est utilisé pour déterminer, à partir d'un nœud racine s, le meilleur saut possible vers chaque autre nœud d d'un graphe G. Cet algorithme prend donc en entrée une racine s et un graphe $G(N, E)$ doté d'une valuation positive w.
Avec un routage au saut par saut, un chemin *alternatif* est un chemin dont au moins le premier arc est distinct du chemin primaire. Le chemin *primaire* désigne le meilleur chemin, ou chemin de coût optimal, calculé selon une métrique donnée, en utilisant un ordre lexicographique pour un tri sans ambiguïté. L'ensemble des définitions est donné dans le tableau 2.1.
Dijkstra Transverse décompose un graphe $G(N, E)$ en triant les arcs $e \in E$ en quatre catégories :

- les arcs correspondant aux premiers sauts des meilleurs chemins $P_1(s, d)$: les interfaces sortantes primaires.
- les arcs appartenant aux sous-arbres constituant les *branches*.
- les arcs *transverses* reliant les branches entre elles ou connectant la racine s à une branche sans être inclus dans un chemin $P_1(s, d)$
- les arcs *internes* reliant les nœuds d'une même *branche* sans y appartenir.

Cette décomposition partitionne l'ensemble des arcs : un arc ne peut appartenir qu'à une et une seule de ces catégories. Par ailleurs, si on néglige l'orientation des arcs, ces quatre ensembles décrivent l'intégralité des arcs appartenant à E. Si deux branches distinctes sont distantes de plus d'un saut, alors il existe une ou plusieurs branches qui, deux à deux, sont distantes d'un saut via un arc transverse. Dans le cas contraire, cela signifierait qu'il existe un nœud n'appartenant à aucune branche, or cela contredit le fait que l'algorithme de Dijkstra calcule un plus court chemin vers chaque destination du graphe. Notre contribution utilise l'algorithme de *Dijkstra*, en lui apportant des modifications déclinées selon trois axes.

Une première *passe* de DT (correspondant, à complexité équivalente, à l'exécution classique de Dijkstra), permet de trouver des chemins alternatifs présentant deux caractéristiques spécifiques. D'une part, le premier saut d'un chemin alternatif diffère nécessairement de celui du chemin primaire, d'autre part le dernier saut d'un chemin transverse simple (un chemin alternatif calculé lors de la première passe de DT) correspond à un arc n'appartenant pas l'arbre des plus courts chemins : un arc *transverse*. Ces chemins forment ainsi l'ensemble des *chemins transverses simples*. Les arcs *transverses* connectent des sous-arbres comportant des sous-ensembles de meilleurs chemins : les *branches*. Une branche regroupe l'ensemble des chemins primaires dont le premier saut est identique. Une fois la première passe de DT effectuée, deux étapes supplémentaires enrichissent cet ensemble de chemin *transverse simple* par composition avec les branches. Dans un premier temps la composition est *retour* car elle utilise les arcs d'orientation opposée au sens de la branche pour former des *chemins transverses retours*. La dernière étape utilise les arcs dont l'orientation suit la direction de la branche pour former par composition *avant* des *chemins transverses avants*.

Pour la composition *retour*, l'algorithme DT considère par défaut que les arcs appartenant à l'arbre des meilleurs chemins sont symétriques en existence et en valuation, ce qui semble une hypothèse raisonnable pour un cœur de réseau d'opérateur. Les liens sont généralement *full duplex* et présentent les mêmes caractéristiques en terme de délais de propagation et de capacité quelle que soit l'orientation. Cependant ces hypothèses ne sont pas primordiales : chaque nœud racine dispose d'une matrice d'adjacence valuée lui permettant de déterminer l'existence et la valuation d'un arc. En effet, pour les réseaux d'accès, ces hypothèses ne sont pas nécessairement vérifiées.

Pour simplifier le présentation de nos procédures, nous ne considérons pas le cas d'un multigraphe aux arêtes multiples entre deux sommets donnés, bien que cette hypothèse ne soit pas un obstacle non plus. En pratique, DT calcule une matrice de coût notée Mc de dimension $k^+(s) \times (|N| - 1)$ contenant un sous-ensemble des meilleures alternatives disponibles vers toutes les destinations via chaque routeur voisin (au plus $k^+(s)$ par destination, en général moins car DT ne stocke qu'un sous-ensemble d'alternatives). Cette matrice servira de support à l'analyse de la validité des prochains sauts définie dans la partie 2.2. Cet ensemble de coût stocké sous la forme d'un tableau à deux dimensions constitue la représentation des prochains sauts à valider : les NH-candidats.

Les chemins k-transverse ne sont pas calculés mais ils peuvent être optimaux. Il s'agit des chemins alternatifs de même coût que le meilleur chemin dont le classement lexicographique est supérieur à celui du chemin primaire. Pour les obtenir, il faut dès la première passe modifier l'algorithme de Dijkstra de

Termes	Définitions
branche $branch_h(s), h \in succ(s)$	Sous-arbre enraciné en h regroupant l'ensemble des chemins $P_1(s,d),\ \forall d \in N$ ayant un premier arc commun $e_1 = \{s,h\}$.
arc transverse	Un arc e est transverse si $\exists h' \neq h \in N$ tel que $e.x \in branch_h(s) \wedge e.y \in branch'_h(s)$ ou si $e.x = s$ et $e \notin P_1(s,d),\ \forall d \in N$
arc interne	Un arc e est interne s'il connecte deux nœuds $e.x$ et $e.y$ d'une même branche $branch_h(s)(e.x, e.y \in branch_h(s))$ et que $e \notin branch_h(s)$
chemin k-transverse	Un chemin est k-transverse si il contient exactement k arcs transverses et pas d'arc interne.
chemin *transverse simple* $\mathfrak{P} \in Pt(s,d)$	Un chemin 1-transverse de m sauts $(e_1, e_2, ..., e_m)$ tel que $(e_1, e_2, ..., e_{m-1}) = P_1(s, e_{m-1}.y)$ et tel que e_m soit un arc transverse $(e_m.y = d)$.
chemin *transverse retour* $\mathfrak{P} \in Pbt(s,d)$	Un chemin de m sauts $(e_1, e_2, ..., e_m)$ tel qu'il existe z $(1 < z < m)$ avec $(e_1, ..., e_z) \in Pt(s, e_z.y)$ et tel que $(e_m^{-1}, e_{m-1}^{-1}, ..., e_{z+1}^{-1}) = P_1(d, e_{z+1}.y)$.
chemin *transverse avant* $\mathfrak{P} \in Pft(s,d)$	Un chemin de m sauts $(e_1, e_2, ..., e_m)$ tel qu'il existe z $(1 < z < m)$ avec $(e_1, ..., e_z) \in Pt(s, e_z.y) \vee Pbt(s, e_z.y)$ et tel que $(e_{z+1}, e_{z+2}, ..., e_m) = P_1(e_{z+1}.x, d)$.
$F^i,\ 0 \leq i < \lvert N \rvert$	Composition i sauts en arrière de la fonction père. Un nœud $n = F_s^i(a) = F_s^j(b),\ a, b \in N$, tel que $\sum i + j$ soit minimal désigne l'ancêtre commun le moins éloigné des nœuds a et b.
DT-voisin	Prochain saut calculé avec DT et enregistré dans la matrice Mc sous la forme d'un coût non infini.

Tab. 2.1 – Terminologie

la même manière que pour le protocole de routage ECMP. Le paragraphe 2.1.4 propose une extension adaptée au calcul de l'ensemble des chemins alternatifs de coût optimal et fournit une généralisation de cette extension dans le cadre d'un routage multichemins aux coûts inégaux.

2.1.2 Algorithmes et complexité

Les calculs opérés par l'algorithme *Dijkstra transverse* se décomposent en trois étapes :

a) Calculer l'arbre des meilleurs chemins (l'ensemble des chemins $\{P_1(s,d)\},\ \forall d \in N$, Alg. 1 : bloc 1) et l'ensemble des chemins *transverses simples* ($\{Pt(s,d)\},\ \forall d \in N$, Alg. 1 : bloc 2).

b) Construire l'ensemble des chemins *transverses retours* ($\{Pbt(s,d)\},\ \forall d \in N$, Alg. 2 : bloc 1) et l'ajouter à l'ensemble précédent.

c) Construire l'ensemble des chemins *transverses avants* ($\{Pft(s,d)\},\ \forall d \in N$, Alg. 2 : bloc 2) et l'ajouter à l'ensemble précédent.

L'ensemble des chemins P_{DT} calculé sur un nœud s avec DT est donc l'union de quatre sous-ensembles disjoints :

$$P_{DT} = \{P_1(s,d)\} \cup \{Pt(s,d)\} \cup \{Pbt(s,d)\} \cup \{Pft(s,d)\},\ \forall d \in N$$

Le pseudo code de l'algorithme *Dijkstra transverse* est donné ci-après. Par rapport à l'algorithme de Dijkstra classique, le premier algorithme introduit simplement un calcul supplémentaire sur Mc permettant de stocker des chemins alternatifs disjoints du chemin primaire : les chemins transverses simples. De même, le premier bloc du second algorithme ne construit que des chemins disjoints entre eux. Par définition, les arcs appartenant à deux sous-arbres sont distincts, un arc transverse n'appartient pas à un chemin primaire et un arc appartenant à une branche ne peut être utilisé dans ses deux orientations sans créer un circuit. En revanche, les chemins transverses avants partagent par nature un certain nombre d'arcs en commun avec les chemins primaires.

Algorithme 1 L'algorithme Dijkstra-Transverse (DT), première passe

Procédure Dijkstra-Transverse
($G(N,E)$: **graphe** $w : E \rightarrow R^+$: **valuation** s : **racine**)

$Mc_{k^+(s),|N|-1}$: **Matrice de coût par prochain saut.**

$Tc_{|N|-1},\ Tp_{|N|-1}$: **Vecteurs des meilleurs coûts et prochains sauts.**

$F_{|N|-1}$: **Liste des nœuds pères.**

$To_{|N|}$: **Liste des nœuds marqués.**

$Mc(k,d)$ et $Tc(d) \leftarrow \infty, \forall d \in N,\ k \in succ(s)$.

$Tc(s) \leftarrow 0$.

Tant que $(|To| < |N|)$ **faire**

Sélectionner le nœud x $(x \notin To)$ de plus petit coût $Tc(x)$

Pour $y \in succ(x)$ **faire**

Si $Tc(x) + w(x,y) < Tc(y)$ **Alors**

[Bloc 1 : Arbre des plus courts chemins]

Mettre à jour $Tc(y)$ et $Tp(y)$, $F_s(y) = x$

Mettre à jour $Mc(Tp(y), y)$

Sinon

[Bloc 2 : Ensemble des chemins transverses simples]

Si $Mc(Tp(x),x) + w(x,y) < Mc(Tp(x),y)$ **Alors**

Mettre à jour $Mc(Tp(x), y)$

Fin Si

Fin Si

Placer x dans To

Fin Pour

Fait

Retourner Mc

Fin

Algorithme 2 Composition *retour* et *avant* avec DT

Procédure DT-composition
($G(N,E)$: **graphe,** $w : E \to R^+$: **valuation,** s : **racine,** Mc : **Matrice de coût,** $F_{|N|-1}$: **Liste des nœuds pères,** $To_{|N|}$: **Liste des nœuds marqués (les trois derniers objets sont instanciés via l'algorithme 1)**)

[Bloc 1 : composition retour]
Pour i **de** $|N|$ **à** 1 **faire**
  Pour $y \in succ(s)$ **faire**
    Si $Mc(y, To(i)) + w(To(i), F(To(i))) < Mc(y, F(To(i)))$ **Alors**
      Mettre à jour $Mc(Tp(To(i)), F(To(i)))$
    Fin Si
  Fin Pour
Fin Pour
[Bloc 2 : composition avant]
Pour i **de** 1 **à** $|N|$ **faire**
  Pour $y \in succ(s)$ **faire**
    Si $Mc(y, F(To(i))) + w(F(To(i)), To(i)) < Mc(y, To(i))$ **Alors**
      Mettre à jour $Mc(Tp(F(To(i))), To(i))$
    Fin Si
  Fin Pour
Fin Pour

Retourner Mc
Fin

Les vecteurs Tc et Tp sont utilisés pour stocker les informations permettant la construction directe de la table de routage monochemin. Un chemin est implicitement représenté par son coût et son premier saut. Ces informations sont transmises de père en fils de la même manière qu'avec l'algorithme de Dijkstra. La liste des nœuds pères F permet de transmettre par *filiation* les caractéristiques de base d'un meilleur chemin : le coût par addition, la liste des prédecesseurs par composition et le premier saut qui représente l'ancêtre commun le plus lointain en dehors de la racine s. La liste To permet d'enregistrer l'ordre de visite des nœuds marqués. Cette liste est utilisée pour la cohérence de la composition retour et avant. Les branches sont utilisées selon l'ordonnancement de leur construction. Les procédures de mise à jour de Tc, Tp et Mc sont dépendantes du nouveau meilleur coût à enregistrer. Pour les vecteurs Tc et Tp, il suffit de réaliser les opérations suivantes : $Tc(y) = Tc(x) + w(x,y)$ et $Tp(y) = Tp(x)$. Initialement $Tp(y) = y$ si $y \in succ(s)$. Pour mettre à jour Mc, il suffit d'enregistrer le meilleur nouveau coût découvert dans Tp. Si la mise à jour de Mc fait intervenir le nœud racine s, alors, par défaut, le

coût est considéré nul.
DT ne calcule pas de chemins contenant des circuits car la procédure de mise à jour de Mc vérifie que le nouveau coût enregistré soit strictement inférieur à l'ancien pour un même premier saut. Un chemin contenant un circuit est nécessairement plus long que le même chemin privé de ce circuit car la valuation est strictement positive.
Preuve :
Si un chemin $P_j(s,d) = (e_1, ..., e_a, ..., e_b, ..., e_m)$ calculé en phase de composition avant contient un circuit entre les nœuds $e_a.x$ et $e_b.y$ $(a < b)$, alors il existe un chemin $P_i(s,d) = (e_1, ..., e_k, ..., e_n)$ avec $\forall e_k \neq e_l$ pour $a < l < b$, tel que $C_i(s,d) < C_j(s,d)$. Le chemin $P_i(s,d)$ est nécessairement calculé antérieurement à $P_j(s,d)$ car $P_j(s,d)$ est par définition composé du chemin $P_i(s,d)$.

La complexité d'exécution de DT est directement liée à celle de Dijkstra dans le pire des cas. Sans utiliser de structure particulière pour implémenter le vecteur Tc, la complexité est la suivante sur chaque nœud s :

$$O(|N|^2 + |E| + |N| \times k^+(s)) = O(|N|^2)$$

DT introduit une complexité supplémentaire proportionnelle au degré sortant du nœud s considéré, or $k^+(s) \leq |N|$, ainsi les termes $|E|$ et $|N| \times k^+(s)$ sont couverts par $O(|N|^2)$.
En utilisant un tas de Fibonacci (CSRL01) pour implémenter Tc, il est, en théorie, possible d'obtenir une complexité de : $O(|N|log_2|N| + |E|)$. L'économie en complexité d'exécution d'un tas de Fibonacci est relative à la procédure nécessaire pour obtenir $min(Tc)$. Le coût d'extraction d'un minimum est unitaire alors que le coût nécessaire à la suppression d'un minimum est en $log_2|N|$.
En pratique, un tas de Fibonacci est une structure complexe utilisant des pointeurs circulaires, et le coût d'implémentation serait trop élevé pour que le temps d'exécution de l'algorithme bénéficie d'une réelle économie.

2.1.3 Propriétés

Après exécution des deux procédures de DT, la matrice Mc contient une majoration du meilleur coût des chemins calculés via un ensemble de prochains sauts candidats pour toutes les destinations. Ainsi le meilleur coût via un voisin $v = NH_j(s,d)$ d'un routeur racine s, et vers une destination d vérifie :

$$C_1(v,d) \leq C_j(s,d) - w(s,v)$$

Il se peut en effet que le coût $Mc(v,d)$ soit supérieur à celui du meilleur chemin alternatif existant via v car les arcs reliant deux nœuds d'une même branche (les arcs internes) sont ignorés dans l'exécution de DT. Ainsi les chemins *transverses avants* peuvent être composés d'un chemin transverse retour qui constitue un détour par rapport au chemin alternatif de meilleur coût (voir l'exemple introduit dans le paragraphe 2.1.5). Cependant, d'une part, la majoration éventuelle de ce coût permet de conserver la garantie que le processus de validation utilisé s'effectue correctement, d'autre part, le prochain saut alternatif est enregistré par DT. L'arc ignoré par un nœud s car interne ne le sera pas nécessairement

par les routeurs en aval (ici v).

Par ailleurs, il est à noter que par définition, tout chemin contenant au moins deux arcs transverses, passe par un nœud appartenant à une branche commune avec celle utilisée par un chemin transverse simple (sans compter la branche contenant la destination). Cela implique qu'un chemin k-transverse est nécessairement au moins aussi long qu'un chemin 1-transverse. En ce sens, DT considère uniquement les candidats représentant les *meilleurs voisins*, car il sélectionne seulement les meilleures alternatives locales : les NH-candidats.
Notre algorithme DT calcule tous les prochains sauts correspondant à l'ensemble des chemins primaires et des chemins 1-transverse. L'ensemble de ces chemins peut être défini sous le terme *1-branche distance*. Autrement dit, s'il existe une ou plusieurs alternatives de routage au chemin primaire entre deux nœuds, alors au moins l'une d'entre elles est comprise dans les prochains sauts correspondant à l'ensemble des chemins 1-branche distance.

Definition 7 (**Ensemble des chemins 1-branche distance**). *L'ensemble 1-branche distance est l'union des chemins primaires et des chemins* 1*-transverse.*

Propriété 1 (**Couverture 1-branche distance**). *S'il existe un chemin alternatif alors il existe un chemin* 1*-transverse.*

L'ensemble des chemins *1-branche distance* contient au moins deux chemins, d'une racine donnée vers chaque nœud du graphe pour lequel il existe un chemin alternatif.
La démonstration de cette propriété repose sur deux propriétés structurelles. D'une part, un chemin *k-transverse* est en partie formé par un chemin *l-transverse* avec $l = k - 1$, et donc par récurrence d'un chemin *1-transverse*. D'autre part, il existe toujours un chemin *1-transverse* calculé par DT utilisant le même premier saut qu'un chemin contenant un ou plusieurs arcs internes. Bien que ne pouvant emprunter d'arc interne, il existe toujours un chemin 1-transverse contenant le premier saut primaire menant à la destination. Cette propriété sera détaillée dans la preuve.
La preuve par récurrence sur les chemins k-transverses est intuitive : par définition, pour atteindre une destination d située sur une branche différente, il est nécessaire d'utiliser un arc transverse. Néanmoins, il est possible que le premier saut d'un chemin k-transverse ne soit pas considéré dans les premiers saut calculés par DT si $k > 1$. Dans ce cas, il existe un chemin alternatif, présentant nécessairement un coût inférieur ou égal au chemin alternatif k-transverse, qui peut utiliser le premier saut menant à l'avant-dernière branche traversée par le chemin k-transverse. Notons a le premier nœud traversé sur l'avant-dernière branche, et considérons l'alternative 1-transverse telle que : d'une part, elle soit composée du meilleur chemin menant à a et d'autre part elle utilise la même portion de chemin entre a et d utilisé par le chemin k-transverse contenant le dernier arc transverse. On peut en conclure que l'existence d'un chemin k-transverse implique que DT enregistre, implicitement sous la forme d'un coût, au moins une alternative 1-transverse.
La figure 2.1 illustre la notion de couverture 1-branche distance. Les arcs pleins désignent les arcs du SPT enraciné en s, alors que l'arc en pointillé (noté i) est un arc interne et les arcs à tirets (noté t) sont

des arcs transverses. Cette figure peut servir de base à la démonstration 1 et permet de constater la relation de récurrence entre chemins 2-transverse et 1-transverse. Le chemin 2-transverse entre s et d via le nœud 6 utilise la branche $branch_1(s)$ pour relier l'arc transverse $\{b, c\}$. De plus, bien que le chemin $(\{s, 1\}, \{1, b\}, \{b, c\}, \{c, 11\}, \{11, d\})$ ne soit pas pris en compte par DT, le chemin transverse avant $(\{s, 1\}, \{1, b\}, \{b, c\}, \{c, n\}, \{n, 11\}, \{11, d\})$ permet de considérer l'interface de sortie correspondant au nœud 1.

Preuve 1 (Couverture 1-branche distance). *Procédons avec un raisonnement par l'absurde, et supposons qu'il existe un chemin alternatif* $(e_1, ..., e_m)$ *reliant* $s = e_1.x$ *et* $d = e_m.y$ *alors qu'aucun chemin* 1*-transverse ne peut relier* s *à* d*.*

Ce chemin alternatif utilise un premier saut $e_1.y$ non contenu dans l'ensemble *1-branche distance*. Soit $P_1(s, d) = (a_1, ..., a_l)$, le meilleur chemin reliant s à d. Par définition, on sait que $e_1 \neq a_1$ (un chemin alternatif est au moins disjoint du premier saut par rapport au chemin primaire). Si e_1 n'appartient à aucune branche, il s'agit alors d'un arc transverse. Or $branch_{e_1}(s)$ et $branch_{a_1}(s)$ ne peuvent avoir une intersection non nulle sinon cela contredirait l'unicité du chemin primaire. Notons que par construction, nous utilisons un ordre lexicographique pour différencier les chemins de coût égaux. Par conséquent, le chemin alternatif comporte nécessairement un arc transverse $e_i = \{b, c\}, 1 \leq i \leq m$ pour atteindre la branche contenant d.
Par construction, le chemin de s vers c passant par b est un chemin transverse simple $\mathfrak{P} \in Pt(s, c)$. Notons n le nœud désignant l'ancêtre commun le plus proche en nombre de sauts de c et de d.
Si $n = c = F_s^0(c)$ alors il existe un chemin transverse avant entre s et d (formé d'un chemin transverse simple de s vers c et d'un meilleur chemin entre c et d). Si $n = d = F_s^0(d)$ alors il existe un chemin traverse retour entre s et d (formé d'un chemin transverse simple de s vers c et d'un chemin d'orientation opposée au meilleur chemin entre d et c). Donc aucun des ces deux cas n'est possible car il s'agit de chemin 1-transverse.
Admettons maintenant que le chemin alternatif contienne un ou plusieurs arcs internes (cela exclut les deux cas précédents). L'existence d'un arc interne signifie que la branche contenant d se subdivise à partir du nœud n en deux sous-branches. Cela implique alors qu'il existe un chemin transverse avant entre s et d et que $C_1(s, c) > C_1(s, n)$ et $C_1(s, d) > C_1(s, n)$. Il s'agit de la composition d'un chemin transverse retour appartenant Ã l'ensemble $Pbt(s, n)$ et d'un meilleur chemin $P_1(n, d)$.
Dans ce cas, l'arc interne est contournable par une composition retour puis avant [1]. Tout arc interne peut être contourné de cette manière : il suffit de *retourner* au nœud n subdivisant la branche contenant d et c en deux sous branches. Ainsi, il existe nécessairement un chemin *1-transverse* permettant d'atteindre d, ce qui contredit la supposition de départ.
En d'autres termes, le segment de chemin contenant un ou plusieurs arcs internes entre c et d est remplacé par deux segments de chemin : entre c et n, puis entre n et d. L'algorithme DT construit un ensemble de prochains sauts candidats entre la racine s et chaque nœud destination d. La propriété 1 nous permet de conclure que le cardinal de cet ensemble est au moins égal à 2 s'il existe une alternative

[1] C'est précisment ce type de composition qui induit la majoration des coûts des chemins transverse avant

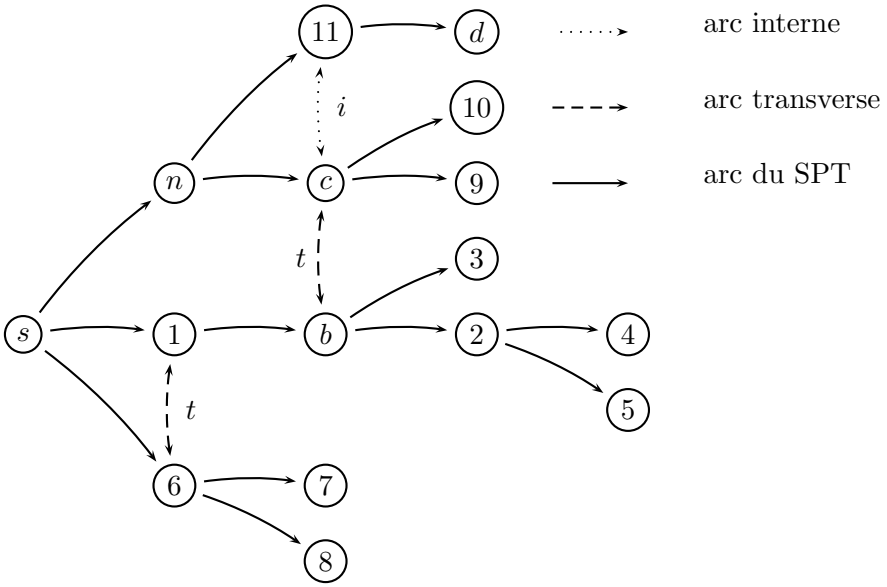


FIG. 2.1 – Couverture une branche distance : arcs internes et chemins k-transverses

de routage entre s et d.

Definition 8 (**DT-voisinage**). *Un **DT-voisin** d'un nœud racine s vers une destination d est un voisin v ($v \in succ(s)$) contenu dans l'ensemble des NH-candidats obtenu avec DT sur s.*
*Le **DT-voisinage** désigne l'ensemble des prochains sauts calculé par DT pour un nœud s et une destination d donnée.*
Le DT-voisinage à p sauts désigne, pour un couple (s, d), l'ensemble des compositions obtenues de proche en proche sur les DT-voisins depuis s à destination de d.

Definition 9 (**DT-espace sortant**). *On notera $k^+_{DT}(s, d)$ le nombre de NHs calculés avec DT sur s vers la destination d. Il s'agit du cardinal du DT-voisinage à un saut.*

Le terme NH-candidat n'est pas spécifique à DT, il désigne simplement la capacité d'un algorithme de recherche opérationnelle à calculer un ensemble de NHs dont le cardinal est supérieur à 1 pour un couple (s, d) donné.
L'ensemble des DT-voisins peut être enrichi si DT ne limite pas ses calculs aux chemins 1-transverse. Il est notamment possible de stocker l'ensemble des NHs correspondant à des chemins de coûts optimaux bien qu'ils soient k-transverses ($k > 1$). Le paragraphe suivant énonce les modifications à apporter à la première partie de l'algorithme *Dijkstra Tranverse*.

2.1.4 Multi-DT

Dans cette partie, nous proposons une extension de notre algorithme de calcul de chemins multiples apportant deux améliorations . D'une part, l'algorithme *Dijkstra Transverse* se contente d'enregistrer

les alternatives de coût optimal correspondant à des chemins 1-transverse alors qu'il est possible qu'il en existe d'autres, avec des premiers sauts différents, utilisant des chemins k-transverses ($k > 1$). D'autre part il est possible d'améliorer la majoration des coûts qui sont stockés dans la matrice Mc. Pour rappel, cette imprécision est due aux chemins transverses avant constitués d'un chemin transverse retour alors qu'il existe un chemin plus court, de même premier saut, contenant un ou plusieurs arcs internes (voir la figure 2.1).

Il est possible de généraliser la démarche de l'algorithme de Dijkstra utilisé pour calculer l'ensemble des chemins de coûts égaux. Pour cela, il suffit de stocker à chaque visite d'une extrémité terminale $e.y$ d'un arc e sortant d'un nœud $e.x$ l'ensemble des premiers sauts qui permettent de l'atteindre, c'est-à-dire, tous les premiers sauts permettant de relier la racine s à $e.x$.

La notion de *transmission* de père à fils est multiple. Le terme transmission désigne ici l'héritage en termes de prochains sauts caractérisé par la relation de mise à jour du vecteur Tp dans l'algorithme 1. Dans le cas de multi-DT, cette mise à jour est généralisée : toute extrémité sortante d'un arc visité bénéficie par filiation des prochains sauts déjà enregistrés vers l'extrémité entrante de l'arc. L'héritage n'est plus réservé aux nœuds fils car la transmission des prochains sauts est indépendante de la notion de branche.

Par rapport à DT, il faut alors mémoriser l'ensemble des prochains sauts vers chaque destination dans une matrice. Ces prochains sauts peuvent être représentés par une relation d'existence ($\neq \infty$) dans une matrice que nous noterons Tp. Cette matrice n'est pas nécessaire en pratique car cette information est déjà implicitement contenue dans Mc sous la forme d'un coût, mais nous l'utilisons pour simplifier la présentation de l'algorithme *multi-DT* (Alg. 3) et mettre en avant l'analogie avec le vecteur Tp utilisé dans la version initiale de DT (Alg. 1).

Algorithme 3 L'algorithme multi-DT

Procédure multi-DT
($G(N,E)$: **graphe** $w : E \to R^+$: **valuation** s : **racine**)

$Mc_{k^+(s),|N|-1}$: **Matrice de coût par prochain saut.**

$Tp_{k^+(s),|N|-1}$: **Matrice des prochains sauts.**

$Tc_{|N|-1}$: **Vecteur des meilleurs coûts.**

$F_{|N|-1}$: **Liste des nœuds pères.** $To_{|N|}$: **Liste des nœuds marqués.**

$Mc(k,d)$,$Tp(k,d)$ et $Tc(d) \leftarrow \infty$, $\forall d \in N,\ k \in succ(s)$.

$Tc(s) \leftarrow 0$.

Tant que $(|To| < |N|)$ **faire**

Sélectionner le nœud x $(x \notin To)$ de plus petit coût $Tc(x)$

Pour $y \in succ(x)$ **faire**

Pour $k \in succ(s) | Tp(k,x) \neq \infty$ **faire**

Mettre à jour $Tp(k,y)$

Fin Pour

Si $Tc(x) + w(x,y) < Tc(y)$ **Alors**

Mettre à jour $Tc(y)$, $F_s(y) = x$

Sinon

Pour $k \in succ(s) \mid Tp(k,x) \neq \infty$ **faire**

Si *Mc(Tp(k,x),x)+w(x,y)<Mc(Tp(k,y),y)* **Alors**

Mettre à jour $Mc(Tp(k,y),y)$

Fin Si

Fin Pour

Fin Si

Placer x dans To

Fin Pour

Fait

Retourner Mc

Fin

La procédure de mise à jour de $Tp(k,y)$ est appelée à chaque visite d'un arc. L'enregistrement suit une règle transitive : $Tp(k,y) = Tp(k,x)$. De manière plus intuitive, tous les prochains sauts enregistrés jusqu'à x sont tramsmis à y. Initialement, si $x = s$ alors $Tp(y,y) = y$.
Pour un protocole de routage comme ECMP, la mise à jour de Tp est réalisée seulement si $Tc(x) +$

$w(x,y) \leq Tc(y)$. Nous avons choisi de généraliser cette démarche pour également minimiser, lorsque c'est possible, la majoration du coût des chemins transverses avants composés avec un chemin transverse retour.
En effet, lorsque le nœud y n'a pas encore été marqué, cela permet de transmettre à ses voisins toutes les opportunités de routage proposées par x. Si le lien de y vers certains de ses voisins z est un arc interne, la transmission des prochains sauts d'un père vers ses fils n'est plus limitée aux branches : z n'est pas le fils de y sur le meilleur chemin, mais il peut toutefois profiter de tous les prochains sauts découverts jusque là vers y.
L'ensemble des DT-voisins n'est plus limité aux alternatives 1-transverses. Cet algorithme génère un ensemble de chemins dépendant de l'ordre de visite des nœuds et donc du tri des coût lorsque plusieurs éléments dans Tc présentent un coût identique. La seule certitude est le calcul exhaustif de tous les meilleurs chemins de coût égaux : avant de passer à la visite d'un nœud x de coût supérieur, tous les chemins égaux vers le dernier nœud marqué ont nécessairement été visités. Par récurrence, la transmission des coûts tiendra compte de l'ensemble des chemins de coûts égaux vers tous les nœuds déjà marqués.
La complexité de *multi-DT* est légèrement plus élevée que celle de *Dijkstra-Transverse*, car à chaque itération de la boucle *Tant que*, $k^+(s)$ calculs sont effectués pour la transmission des prochains sauts et des coûts. Or le terme $k^+(s) \times |E|$ n'est pas nécessairement couvert par la complexité de la sélection du plus petit coût dans Tc, par conséquent la complexité dans le pire des cas de multi-DT est en $O(|N|^2 + E \times k^+(s))$. Néanmoins, cette compléxité reste inférieure à $k^+(s)$ itérations successives de Dijkstra.
Afin d'illustrer le fonctionnement de l'algorithme *Dijkstra Transverse* et des différents concepts qui s'y rattachent, nous développerons une série d'exemples sur un réseau de petite taille dans le paragraphe qui suit. Nous y illustrerons également le principe de sous-optimalité des meilleurs chemins en utilisant la notion de branche.

2.1.5 Exemples

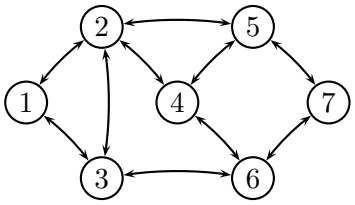
La figure 2.2 illustre le fonctionnement de DT. La figure 2.2(a) représente un réseau constitué de sept routeurs avec une valuation uniforme : le coût d'un chemin est son nombre de sauts, c'est-à-dire le nombre d'arcs qui le composent. Sur les figures 2.2(b),(c),(d) et (e), les arcs pleins représentent les liens appartenant à l'arbre des meilleurs chemins enraciné en s. Il peut aussi s'agir d'arc *retour* e_i^{-1} tel que $e_i \in P_1(s,d) = (e_1, ..., e_i, ..., e_m)$ avec $i > 1$. Ces arcs sont utilisés pour la phase de composition retour (chemins transverses retours). Les arcs à tirets sont les arcs transverses alors que les arcs pointillés sont des arcs internes aux branches. Pour mieux intégrer la composition distribuée de DT et introduire le processus de validation que nous présenterons dans le paragraphe suivant, cet exemple nous donne une représentation visuelle de l'ensemble des chemins 1-transverse pour $s = 4$, $s = 2$, $s = 5$ et $s = 6$.
Pour $s = 4$, on dénombre trois branches : $branch_2(4)$, $branch_5(4)$ et $branch_6(4)$ alors qu'il existe quatre branches enracinées en 2 (2.2(c)) : $branch_1(2)$, $branch_3(2)$, $branch_4(2)$ et $branch_5(2)$.
On peut observer le principe de sous-optimalité grâce aux arcs pleins avant (ou avec des arcs transverses

en cas d'égalité de coût). Par exemple, les chemins optimaux de 4 vers 1 et 3 contiennent des chemins optimaux de 2 vers ces mêmes destinations.

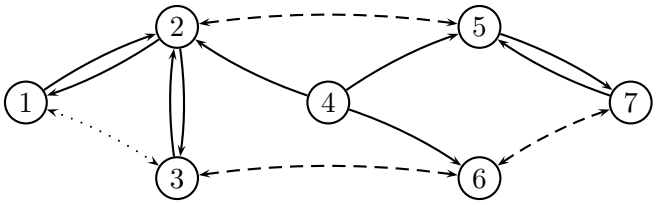
L'algorithme DT, exécuté sur chaque nœud, calcule une matrice de coût dont le nombre de lignes est fonction du degré sortant. Les colonnes représentent l'ensemble des destinations du graphe en dehors de la source (pour $s = 4$, $d \in \{1,\ 2,\ 3,\ 5,\ 6,\ 7\}$) et les lignes l'ensemble des successeurs possibles pour le nœud s considéré (ici : $\{2,\ 5,\ 6\}$). Par exemple, pour $s = 4$, on obtient :

$$Mc(4) = \begin{array}{c|cccccc} & 1 & 2 & 3 & 5 & 6 & 7 \\ 2 & 2 & 1 & 2 & 2 & 3 & 3 \\ 5 & 3 & 2 & 3 & 1 & 1 & 2 \\ 6 & 4 & 3 & 2 & 3 & 1 & 2 \end{array}$$

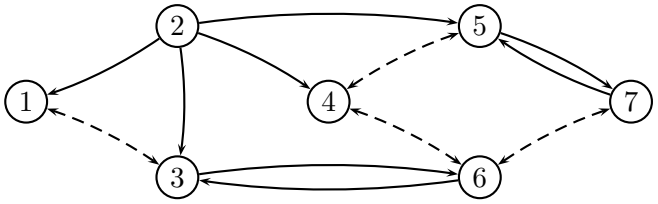
Un meilleur chemin correspond à l'indice de plus petit coût sur une colonne, par exemple sur la colonne correspondant à la destination 7, on dénombre deux meilleurs chemins ($P_1(4,7) = (\{4,5\},\{5,7\})$, $P_2(4,7) = (\{4,6\},\{6,7\})$, c'est l'ordre lexicographique $5 < 6$ qui permet de trier les chemins) de coût égaux ($C_1(4,7) = C_2(4,7) = 2$).
Sur la première ligne, on compte trois meilleurs chemins ($P_1(4,1),\ P_1(4,2),\ P_1(4,3)$), deux chemins transverses simples ($P_2(4,5) \in P_t(4,5),\ P_3(4,6) \in P_t(4,6) =$) et un chemin transverse avant ($P_3(4,7) \in P_{ft}$). Un exemple de chemin transverse retour est le chemin $P_3(4,5) = (\{4,6\},\{6,7\},\{7,5\}) \in P_{bt}(4,5)$: il s'agit d'un arc constituant un meilleur chemin $P_1(4,6)$, un arc transverse $\{6,7\}$ et un arc dont l'orientation opposée forme un meilleur chemin $P_1(5,7)$.
Un exemple de chemin transverse avant formé d'un chemin transverse retour est $P_3(4,1) =$ $(\{4,6\},\{6,3\},\{3,2\},\{2,1\}) \in P_{ft}(4,1)$. Le coût $C_3(4,1) = 4$ est une majoration du meilleur chemin réel via le saut 6 utilisant un arc interne $\{3,1\}$. Le nœud 4 a donc bien connaissance d'une alternative via 6 pour atteindre 1. Cependant, comme l'indique la sous-figure (e), le nœud 6 connaît le meilleur coût de son chemin passant par 3 pour atteindre 1 : l'arc $\{1,3\}$ n'est pas considéré comme interne. Ainsi, étant donné que 4 se base sur l'information dont dispose 6 pour le processus de validation, le meilleur coût réel vers 1 sera considéré. De la même manière, l'arc $\{1,3\}$ et son arc d'orientation opposée $\{3,1\}$ sont considérés comme internes par 4 alors que le routeur 2 les considère comme des arcs transverses.
Vers la destination 3, le nœud 5 ne stocke que l'alternative via son chemin transverse simple $P_3(5,3) =$ $(\{5,4\},\{4,6\},\{6,3\}) \in P_t(5,3)$, l'alternative $(\{5,4\},\{4,2\},\{2,3\}) \in P_{ft}(5,3)$ n'étant pas enregistrée car elle est considérée a posteriori dans l'exécution de DT. Un exemple de chemin 2-transverse non enregistré est $(\{5,7\},\{7,6\},\{6,3\})$, mais il existe un chemin 1-transverse $P_3(5,3) = (\{5,4\},\{4,6\},\{6,3\})$. De cette manière, il existe toutefois un prochain saut alternatif via 4 pour la destination 3.
Une autre matrice de coût intéressante à évaluer est celle calculée par le nœud 2 car elle met en avant l'absence de certaines alternatives. Dans ce cas, le signe ∞ désigne l'absence de solutions c'est-à-dire que DT n'a pas envisagé de chemin par ces successeurs, dans la mesure où il s'agit de chemin k-transverses ($k > 1$). Si on trie les successeurs et les destinations par ordre lexicographique sur leurs identifiants



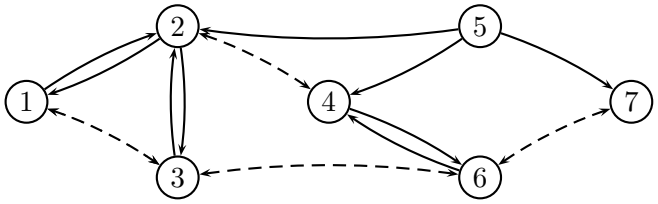
(a)



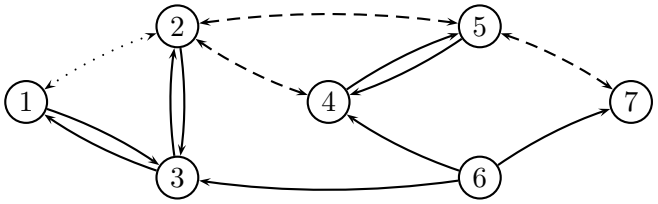
(b) s=4



(c) s=2



(d) s=5



(e) s=6

FIG. 2.2 – Graphe des chemins 1-transverse selon la racine

($succ(2) \in \{1,3,4,5\},\ d \in \{1,3,4,5,6,7\}$), on obtient $Mc_{4,6}$:

$$Mc(2) = \begin{array}{c|cccccc} & 1 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 2 & \infty & \infty & 3 & \infty \\ 3 & 2 & 1 & 3 & \infty & 2 & 3 \\ 4 & \infty & 3 & 1 & 2 & 2 & 3 \\ 5 & \infty & 4 & 2 & 1 & 3 & 1 \end{array}$$

Néanmoins, comme démontré dans la preuve 1, il existe toujours au moins un sucesseur via lequel une alternative de routage existe.
Les prochains sauts calculés par DT sont considérés comme des candidats jusqu'à leur activation par un protocole de validation. Les NH-candidats ne correspondant pas à des chemins primaires deviennent réellement *utilisables* qu'une fois évalués par le processus de validation.
La partie suivante a pour objectif d'élaguer le graphe des compositions de NH-candidats. Les mécanismes de validation qui y sont présentés ne sont pas dépendants de la méthode spécifique au calcul de chemins. L'algorithme DT permet de réduire la complexité de ces mécanismes par une pré-sélection des voisins constituant les meilleurs alternatives possibles pour un sous-ensemble de destinations donné. Le traitement de validation est alors plus léger que lorsque l'ensemble des voisins est évalué pour toutes les destinations. DT(p) désigne précisement l'élagage réalisé par validation composée sur le graphe construit par DT. Dans la suite du chapitre, nous utiliserons, par abus de langage, la notation DT(p) pour désigner le processus de validation dans sa globalité.

2.2 Validation et activation

Dès lors qu'un routeur a fini sa phase de calcul avec DT, consécutive à la réception d'une annonce de modification topologique, il enclenche la phase de validation. L'algorithme DT calcule des chemins aux coût inégaux pour une paire de nœuds donnée. L'ensemble des DT-voisins correspondant à ces chemins ne peut être utilisé avant que chaque routeur ne se soit assuré que ces candidats ne présentent aucun risque de boucles de routage.
La procédure de validation appliquée aux NH-candidats que sont les DT-voisins est notée DT(p). Cette procédure garantit l'absence de boucles de routage dans un état stable.
Pour une meilleure compréhension des mécanismes de base, nous allons commencer par décrire notre procédure de validation à une profondeur d'un saut, c'est-à-dire pour $p = 1$. Nous introduirons notamment une nuance terminologique entre NH validé et NH activé pour que la validation distribuée ne soit pas bloquante. DT(1) est la première brique du processus de validation et ses caractéristiques sont différentes de celles de la procédure de validation à profondeur $p > 1$.
Le protocole DT(p) est une procédure distribuée permettant d'activer parallèlement les prochains sauts du DT-voisinage. DT(p) permet de distinguer l'interface d'entrée dans le processus de commutation. La table de routage comporte un champ **interface d'entrée** (noté ie) générant une granularité plus

fine qu'avec un routage IP classique ne considérant que la destination.
Une table de routage construite avec DT(p) présente au pire des dimensions proportionnelles au degré entrant, au degré sortant et au nombre de destinations. Le nombre de lignes de routage sur un routeur s est borné par :

$$k^-(s) \times k^+(s) \times (|N| - 1)$$

Dans cette partie, nous considérerons que le réseau est dans un état stable. Les routeurs disposent de la même vision topologique globale et possèdent une matrice de coût stabilisée sur cette vision topologique commune. Nous étudierons les techniques permettant de parvenir à la stabilisation du routage dans le paragraphe 2.3.2.
La figure 2.3 illustre les étapes successives du fonctionnement distribué de notre proposition de routage avec DT(1). Le champ *ie* désigne l'interface par laquelle le trafic entre pour mettre en œuvre une commutation spécifique à l'origine du trafic. Le premier diagramme présente une décomposition schématique de la procédure complète de construction des tables de routage multi-dimensionnelles : au champ *ie* s'ajoute par définition la dimension générée par le choix des multiples interfaces de sortie.

2.2.1 Validation par interface entrante

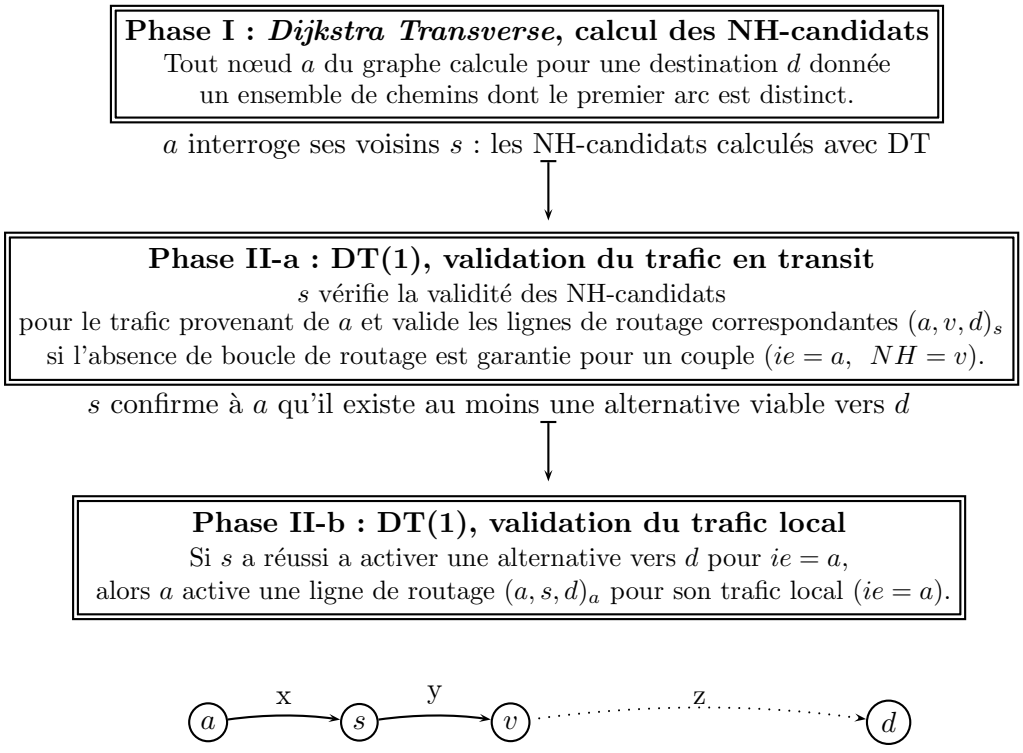
2.2.1.1 Principes

La matrice Mc permet à chacun des routeurs de disposer d'une majoration du meilleur coût de ses DT-voisins vers l'ensemble des destinations du domaine. Grâce à l'algorithme DT, chaque DT-voisin d'un routeur s dispose d'une liste de coûts via ses propres DT-voisins pour toutes les destinations $d \in N$.
Soit un nœud v, un DT-voisin de s tel que $NH_i(s,d) = v$, alors on sait que $C_1(v,d) \leq C_j(s,d) - w(\{s,v\})$. A l'opposé, soit un nœud a voisin de s mais non inclus dans le DT-voisinage de s, alors on sait seulement que $C_1(a,d) \leq C_1(s,d) + w(\{a,s\})$. Ainsi, chaque routeur pourrait localement utiliser la condition LFI ou SPD sur le DT-voisinage.
Pour accroître la diversité des chemins validables localement grâce à cette majoration, nous avons utilisé deux mécanismes :

- l'utilisation d'un protocole de validation de NHs utilisant une diffusion sélective à profondeur variable selon les besoins en alternatives de routage.
- la distinction faite sur l'interface d'entrée pour utiliser des lignes de routage différentes selon la provenance du trafic.

Avec les protocoles basés sur la condition LFI, l'espace des routes validés est le même quelle que soit l'origine du trafic, ainsi, la condition de validation est plus stricte que celle que nous utilisons.
Nous distinguons deux modes de commutation pour le trafic.

Definition 10 (**trafic local et trafic de transit**). *Le trafic **local** à un routeur s est l'ensemble des paquets générés par s ou son sous-réseau, tandis que le trafic en **transit** désigne les paquets provenant des autres routeurs.*



Phase	D	NH	$Coût$
I- (sur a)	d	s	$x+y+z$
I- (sur s)	d	v	$y+z$

DT : Table des NH-candidats

Phase	ie	D	NH	$Coût$
II-a (sur s)	a	d	v	$y+z$
II-b (sur a)	a	d	s	$x+y+z$

DT(1) : Table de routage

FIG. 2.3 – Création d'une ligne de routage par interface d'entrée

Un sous-réseau est considéré ici comme un ensemble de nœuds pour lesquels la connexion au domaine de routage utilise, directement ou indirectement, s comme passerelle. Pour simplifier, nous assimilerons une interface d'entrée à un routeur en amont.

Definition 11 (**validation et activation**). *Une ligne de routage $(a, n, d)_s$ est **valide** sur s si elle est utilisable pour le trafic de transit provenant du routeur a à destination de d sans risque de boucles de routage sur le routeur a. Cette ligne est **active** si elle est également valide pour le trafic local.*

Pour une destination donnée, une ligne de routage est activée sur s pour a en entrée à condition qu'elle ne présente aucun risque de boucles de routage sur les routeurs s et a. La composition récursive de cette propriété, assure de proche en proche qu'une ligne de routage est active si elle n'induit pas la formation de boucle de routage au niveau routeur. La distinction entre les termes valide et active pour le trafic en transit se fera par l'enregistrement de la ligne de routage précédent son activation éventuelle lors de la validation locale.
Cette activation, réalisée de proche en proche pour le trafic en transit, est définie par la relation $NH(a, s, d) \subset NH(s, s, d)\ \forall a \in pred(s)$. L'activation d'une ligne de routage dépend de l'ordre d'exécution distribuée du protocole. Un NH-candidat permet l'activation d'une ligne de routage s'il passe par ces différents états : validé → (enregistré) → activé. La phase d'enregistrement permet d'attendre une éventuelle activation ultérieure.
La transition de l'état valide à actif, lorsque l'ensemble des lignes de routage pour le trafic local du voisin sondé est en cours de validation, permet de prendre en compte le caractère distribué de la validation : le DT-voisin n'a pas nécessairement fini sa propre phase de validation pour son trafic local lorsque le routeur initiant le processus de validation vérifie chez lui des lignes de routage pour le trafic en transit.

2.2.1.2 Validation à un saut

Initialement, et avant l'éxecution de DT(1), seuls les chemins de meilleurs coût sont activés. La commutation est déjà fonctionnelle sur les routes optimales et DT(1) peut commencer, en tâche de fond, à valider les chemins alternatifs calculés avec DT. Néanmoins, les messages de validations sont également transmis aux meilleurs NHs pour activer des lignes de routage pour le trafic de transit.
Le pseudo-code du processus de validation à profondeur 1 est donné dans l'algorithme 4. Il s'agit d'activer des lignes de routage pour le trafic provenant d'un routeur s sur un DT-voisin $v = NH_i(s, d)$. Ce code est simplifié en considérant une seule destination d. L'algorithme décrit le comportement d'un routeur v à la réception d'un message *query* émis par le routeur initiateur s. Nous considérons que l'état du réseau est stable (voir 2.3.2). La partie 2.3 présentera nos choix d'implémentations pour l'agrégation des requêtes.
Le processus de validation à un saut se base sur un échange de messages $query(d, c, p)$ et $response(d)$ entre routeurs adjacents. Le message *query* contient l'identifiant de destination d, la valeur $c = C_1(s, d)$ du meilleur coût calculé avec DT vers d et la profondeur p du processus de validation. Le message $response(d)$ contient l'identifiant de destination d et constitue une réponse positive. A la réception d'un

message $response(d)$ depuis un voisin v, s peut activer la ligne de routage $(s, v, d)_s$ pour son trafic local et activer toutes les lignes enregistrées $(a, v, d)_s$ pour le trafic en transit. Si aucun message $response(d)$ n'est reçu, seuls les meilleurs prochains sauts seront utilisés pour la commutation à destination de d.
La procédure DT(1) correspond également à l'enclenchement du processus de validation distribuée à p sauts. Si la requête *query* contient $p = 1$, alors la validation se fait uniquement sur le routeur adjacent. Si $p > 1$, alors une requête de validation plus complexe $Query\text{-}P(s, d, c, p-1, (v))$ est transmise sur les DT-voisins de v ne satisfaisant pas le critère de validation. Cette extension est décrite dans le paragraphe 2.2.2.
L'éxécution de DT(1) se décompose selon les deux étapes décrites ci-dessous. Sur tout routeur initiateur s, et pour tout DT-voisin v :

[Initialisation] Avant le retour des messages *response*, sur s, seuls les prochains sauts $v = NH_i(s, d)$ tel que $C_i(s, d) - w(s, v) < C_1(s, d)$ sont validés et activés pour la commutation. A ce niveau, le routage est réalisé sans considérer l'interface entrante des paquets, de la même manière qu'avec la condition LFI.

Dès que la phase de calcul de chemins est terminée, s envoie, pour toutes les destinations $d \in N$, un message $query(d, c, p)$ vers ses NH-candidats $v = NH_i(s, d)$ contenant son meilleur coût $c = C_1(s, d)$ pour atteindre d. Plus précisement, avec DT(1), un NH-candidat correspond à un DT-voisin stocké implicitement dans Mc sous la forme d'un coût non infini. On remarquera que l'algorithme DT est déjà, en soi, une forme de pré-sélection. La diffusion des messages de validation est sélective : seuls les DT-voisins sont concernés.

[Validation] La phase de validation distribuée permet de sélectionner certains NH-candidats pour une activation éventuelle. La condition IIC (*Incoming Interface Condition*) utilisée pour la validation est la suivante :

$$C_j(v, d) \leq C_1(s, d) \tag{2.1}$$

Si cette condition est vérifiée sur v, pour un routeur en amont s et vers une destination d, la ligne de routage $(s, NH_j(v, d), d)_v$ correspondant à $NH_j(v, d)$ est validée. Pour tout j vérifiant IIC, le prochain saut correspondant, $NH_j(v, d)$, est validé pour le trafic provenant de s et à destination de d. Le routeur v peut directement activer la ligne $(s, NH_j(v, d), d)_v$ si la ligne $(v, NH_j(v, d), d)_v$ est déjà activée pour son trafic local. Tous les prochains sauts $NH_j(v, d)$ tels que $C_j(v, d) - w(v, NH_j(v, d)) < C_1(v, d)$ sont d'office considérés comme activés car cette condition garantit qu'au moins le meilleur prochain saut du routeur $NH_j(v, d)$ est valide et activable pour v en entrée [2]. Cette condition est notamment satisfaite pour les routes multiples de coût optimal. Si la condition IIC est vérifiée pour au moins un j et que ce j correspond à un NH activé sur v (c'est au moins le cas pour $j = 1$), v transmet une réponse positive à s. Le routeur s positionne $v = NH_i(s, d)$ comme un NH valide et l'active automatiquement car il s'agit d'une ligne de routage concernant le trafic local de s vers d. Ainsi, s est capable d'utiliser v pour commuter ses paquets

[2] Ce mécanisme correspond à une vision à deux sauts

à destination de d ($v \in NH(s,s,d)$) et s active toutes les lignes de routage enregistrées mais non activées pour ses routeurs en amont (par exemple, pour un triplet $(a,v,d)_s$ correspondant au trafic de transit provenant d'un routeur prédécesseur a).

L'enregistrement des DT-voisins et du coût des chemins associés permet de bénéficier d'une vision à deux sauts lors de la phase de validation. Cette vision à deux sauts en aval permet d'améliorer la probabilité de succès de la procédure de validation. Pour cela, il suffit de retrancher la valuation de l'arc reliant le nœud sondé au coût associé au DT-voisin pour connaître le coût un saut en aval. Si $C_1(v,d) \leq C_1(s,d)$, tous les j^e NH appartenant à v ne vérifiant pas la condition IIC mais satisfaisant la relation $C_j(v,d) - w(v, NH_j(v,d)) < C_1(s,d)$ sont alors validés pour le trafic provenant de s.
Si $C_j(v,d) - w(v, NH_j(v,d)) \leq C_1(v,d) < C_1(s,d)$, alors l'activation de $NH_j(v,d)$ est directe. Si $C_j(v,d) - w(v, NH_j(v,d)) \leq C_1(v,d) = C_1(s,d)$, le NH-candidat est enregistré en attendant que v active éventuellement les lignes de routage pour son trafic local. Cette analyse à 2 sauts de profondeur permet de valider et éventuellement d'activer directement les NHs proposant une décroissance stricte du meilleur coût à un saut.
DT(1) permet de construire par validation et activation successive les ensembles de prochains sauts $NH(s,v,d)$ sur v pour le trafic en transit, et $NH(s,s,d)$ sur s pour le trafic local. Ces ensembles correspondent aux lignes de routage activées.

Algorithme 4 Processus de validation à profondeur 1 appliqué à DT : DT(1)

Procédure Validation à profondeur 1
($query(d, c, p)$: **message reçu par** v **depuis** s)
Si $C_1(v, d) \leq c$ **Alors**
valider et activer la ligne $(s, NH_1(v, d), d)_v$
envoyer $response(d)$ à s
Pour j **de** 2 **à** $k^+_{DT}(v, d)$ **faire**
Selon que
$\mathbf{C_j(v, d) - w(v, NH_j(v, d)) \leq C_1(v, d) < c}$:
valider et activer la ligne $(s, NH_j(v, d), d)_v$
$\mathbf{C_j(v, d) - w(v, NH_j(v, d)) < c}$:
valider et enregistrer la ligne $(s, NH_j(v, d), d)_v$
$\mathbf{p > 1}$: envoyer *Query-P(s,d,c,p-1,(v))* vers $NH_j(v, d)$
Fin Selon que
Fin Pour
Sinon
Pour j **de** 1 **à** $k^+_{DT}(v, d)$ **faire**
Si $C_j(v, d) - w(v, NH_j(v, d)) < c$ **Alors**
valider et enregistrer la ligne $(s, NH_j(v, d), d)_v$
Sinon
Si p > 1 Alors
envoyer *Query-P(s,d,c,p-1,(v))* vers $NH_j(v, d)$
Fin Si
Fin Si
Fin Pour
Fin Si
Fin

Grâce aux NH-candidats, il est possible d'activer chaque prochain saut candidat v individuellement dont ceux tels que $C_j(v, d) = C_1(s, d)$. Si l'interface d'entrée du trafic n'est pas prise en compte et que la commutation des flux est réalisée sans distinction sur leurs origines respectives, il est alors impossible d'activer les NH-candidats des voisins proposant un coût égal au meilleur coût local.

Exemple de validation :
La figure 2.4 illustre les possibilités de routage accrues par rapport à la condition LFI grâce à DT(1). La valuation des liens et le coût des routes sont notés à proximité des liens.

Sur cette figure on observe que les notions de routeurs en amont/aval ne sont pas antagonistes pour une destination donnée. La condition IIC permet à un routeur s de commuter des paquets via un voisin v à destination de d bien que v puisse également envoyer du trafic vers d via s. Les paquets échangés n'appartiennent pas aux même flux sinon cela provoquerait la formation de boucles de routage.
Cet *échange croisé* est impossible avec les règles de routage classique : les notions de routeurs en amont/aval sont déterminées pour une destination donnée. La condition IIC permet de redéfinir cette règle en fonction de la provenance d'un flux. La direction d'un flux est caractérisée par sa destination et la dernière interface traversée.
Sur cet exemple, on constate que s dispose de trois solutions de routage pour atteindre d, il peut utiliser directement son chemin de meilleur coût x passant par $n = NH_1(s, d)$, ou bien ses voisins a et v. Or, a et v sont également capables d'utiliser s pour router leur trafic à destination de d. En revanche, la condition LFI ne peut pas tirer partie de ce cas de figure. Bien entendu, dans le cadre du partage de charge cela n'implique pas nécessairement que s et v soient amenés à *croiser* leur flux à destination de d. Si s cherche à accroître son débit vers d alors que son chemin de coût x est déjà saturé, ses deux chemins alternatifs de coût $x + y$ peuvent lui permettre d'augmenter son flot maximal - sous réserve que ces chemins soient peu chargés et disjoints sur le(s) lien(s) saturés.
Il peut être également intéressant dans le cadre du routage à qualité de service d'étudier la possibilité de *croiser* les flux selon les besoins demandés. Pour la destination d, à la granularité de l'interface entrante des paquets (ie), la table de routage de s prend alors la forme de la table 2.2.

La première ligne indique notamment que s dispose de trois interfaces de sorties pour acheminer les paquets générés localement à destination de d. La première colonne désigne l'origine du trafic, c'est-à-dire l'interface d'entrée.
Le routeur v proposera nécessairement une ligne de routage pour $ie = s$ comme celle donnée dans la figure 2.3.

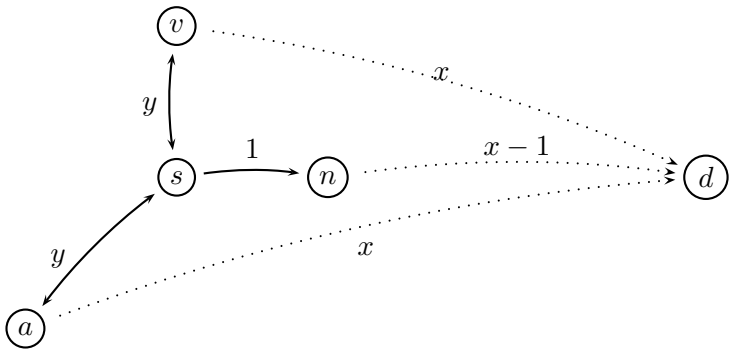


FIG. 2.4 – Validation des NHs sur des voisins à coûts égaux

ie	Dest	NH	Coût
s	d	n	x
s	d	v	$x+y$
s	d	a	$x+y$
a	d	n	x
a	d	v	$x+y$
v	d	n	x
v	d	a	$x+y$

TAB. 2.2 – Table de routage sur s pour la destination d

s	d	$NH_1(v,d) \in NH(s,v,d)$	x

TAB. 2.3 – Ligne de routage sur v vers d via s

La condition IIC signifie qu'un premier saut $NH_j(v,d)$ d'un chemin $P_j(v,d)$ est valide pour un couple (s,d) (il en est de même pour les NHs associés à un chemin de meilleur coût). Pour une destination d donnée, si v, un voisin de s, garantit un coût C égal au meilleur que s a calculé, alors un saut en aval de v, le meilleur coût est strictement inférieur à C (la valuation w des liens est strictement positive). Il est à noter que $NH_j(v,d)$ est réellement activé sur v pour s en entrée seulement si v valide $NH_j(v,d)$ pour son trafic local. Cette propriété est essentielle car il revient individuellement à chaque routeur d'élaguer les circuits dont il fait partie. La preuve de l'absence de boucle de routage est donnée dans la partie 2.2.2.2. Celle-ci est généralisée au processus de validation à profondeur p.
L'idée intuitive se base sur deux principes fondamentaux :

- La décroissance stricte des coûts au plus à *p+1* sauts (ici à 2 sauts).
- La *couverture* de la commutation locale sur la commutation de transit, les NHs activés entre routeurs adjacents vérifient une inclusion : $NH(a,s,d) \subset NH(s,s,d)$ (a désigne une interface d'entrée sur s).

La diffusion sélective de DT(1) explore uniquement les routeurs adjacents correspondant aux chemins 1-transverse : les DT-voisins. De proche en proche, DT(1) ne considère que la composition des NHs calculés avec DT. Cela permet de diminuer la charge liée au nombre de messages *query/response* échangés.
Dans la mesure où IIC est moins stricte que LFI, notre procédure de validation permet d'activer un nombre de lignes de routage plus élevé qu'avec un critère de décroissance stricte des meilleurs coûts à un saut. Si la politique de routage multichemins se base sur la condition LFI et ne considére que les DT-voisins, alors l'ensemble des NHs activés avec DT(1) inclut l'ensemble des NHs activés par LFI.
Pour augmenter le nombre d'alternatives de routage et accroître la diversité des multichemins, il est

nécessaire d'approfondir la recherche. Plus précisement, il s'agit de transmettre les requêtes de validation *query* via les NH-candidats à au plus p sauts pour tester la composition des chemins. DT(p) désigne l'approfondissement de la procédure de validation sur le sous-graphe généré par la composition des DT-voisins. DT(p) élague ce sous-graphe pour vérifier l'absence de boucles de routage. L'éxécution en profondeur est initiée sur les routeurs adjacents. Pour cela, le paramètre de profondeur p, transmis dans le message *query*, doit contenir une valeur supérieure à 1.

2.2.2 Validation à p sauts

Pour rappel, une ligne de routage peut être dans trois états avant d'être utilisée pour la commutation :

- candidate : il s'agit de son état initial, elle est stockée sous la forme d'un DT-voisin.
- validée : il s'agit d'une ligne enregistrée par le processus de validation mais non activée pour le trafic de transit. Elle correspond à l'association d'un DT-voisin et d'une interface entrante.
- activée : il s'agit d'une ligne validée pour le trafic local. S'il existe un couple (DT-voisin, interface d'entrée) enregistré tel que le DT-voisin soit un prochain saut actif pour le trafic local, alors la ligne de routage correspondante est activée pour le trafic en transit provenant de cette interface d'entrée.

2.2.2.1 Description du protocole

Afin de présenter formellement DT(p), nous commençerons par définir la notion de ***route*** par opposition au terme ***chemin***. La notion de chemin est virtuelle car relative à un calcul local, seul le premier saut est réellement utilisé pour la commutation saut par saut. Une route désigne une composition de prochains sauts activés par un processus de validation.

Definition 12 (Route). *Une route $\mathfrak{R}$ de m sauts reliant une source s et une destination d est une composition de prochains sauts activés de proche en proche en fonction de l'interface d'entrée et prend cette forme ($s = r_0$, $d = r_m$) :*

$$\mathfrak{R} = (r_1, r_2, \cdots, r_i, r_{i+1}, \cdots, r_m)$$

avec $r_{i+1} \in NH(r_{i-1}, r_i, d)$ et $r_1 \in NH(s, s, d)$ ($r_i \in N$).

Le terme $R_m(s, d)$ désigne l'ensemble des routes de m sauts activées par le processus de validation entre s et d.
Une route $\mathfrak{R}' \in R_m(s, d)$ est également strictement définie par une série d'identifiants représentant le rang des NHs utilisés parmi ceux activés tel que :

$$\mathfrak{R}' = (n_1, n_2, \cdots, n_i, n_{i+1}, \cdots, n_m)$$

avec n_{i+1} désignant le rang d'un NH activé parmi $NH(r_{i-1}, r_i, d)$. Le routeur r_i correspond au NH de rang n_{i-1} activé par r_{i-1} ($NH_{n_{i+1}} \in NH(r_{i-1}, r_i, d)$).

Avec cette terminologie, nous pouvons décrire notre processus de validation avec une recherche distribuée à p sauts et en largeur d'abord (*breadth first search*, BFS). Dans la suite de ce paragraphe, nous décrirons cette vague de requêtes BFS pour une route $\mathfrak{R} = (r_1, r_2, \cdots, r_i, r_{i+1}, \cdots, r_m)$ dont l'activation est réalisée avec DT(p). Pour simplifier, de même que pour DT(1), la description de nos algorithmes représente seulement le processus de validation pour un couple (s,d) donné. Une requête en profondeur, *Query-P*, est déclenchée sur chaque DT-voisin $v = r_1$ d'un routeur s, si DT(1) a échoué pour un NH-candidat de v. Ce processus est initié par le routeur racine s si la profondeur p choisie est strictement supérieure à 1. Lorsque DT(1) n'a pas réussi à valider, pour un routeur en entrée s, un k^e NH-candidat de r_1, $NH_k(r_1, d)$, alors une vague BFS est déclenchée sur celui-ci. Dans l'algorithme 4, cet échec à profondeur 1 correspond à l'émission d'un message *Query-P(s,d,c,p-1,(v))* vers $NH_j(v, d)$ $(j = k,\ v = r_1)$.

Les messages *Query-P(s,d,c,q,P)* contiennent :

- un champ $c = C_1(s, d)$: le meilleur coût de s vers d.
- un champ q $(1 \leq q \leq p)$: le nombre de sauts encore autorisé avant d'atteindre la profondeur maximale p. L'indice $p - q$ désigne la distance en nombre de sauts entre s et le routeur ayant reçu le message *Query-P*.
- un champ P : l'ensemble des routeurs déjà visité. P est un chemin dont la taille est nécessairement inférieur ou égale à p, il désigne une **composition** de DT-voisins entre s et le routeur ayant reçu le message *Query-P*. Il s'agit d'un chemin testé par la procédure de validation.

L'objectif de la vague de messages *Query-P* est de déterminer si un NH est valide et donc potentiellement activable lorsque celui-ci ne satisfait pas la condition IIC.

Pour *p>1*, DT(p) ne peut tenir compte de l'interface d'entrée sur les routeurs à p sauts. En effet, pour profiter d'un tel grain à p sauts, la commutation devrait tenir compte de l'ensemble des p nœuds traversés par le message *Query-P* [3]. Notre proposition ne considère que la dernière interface traversée pour simplifier la commutation. Par conséquent, la condition IIC ne peut être utilisée NH par NH. Elle doit être satisfaite pour l'ensemble des DT-candidats. L'algorithme de calcul DT restreint cet ensemble aux DT-voisins : toutes les compositions de prochains sauts possibles ne sont pas explorées. D'une part, cela permet de réduire le nombre de messages, d'autre part la probabilité de validation est augmentée. Par ailleurs, le processus de validation initié par un routeur s est seulement sensible aux boucles de routage faisant intervenir s, c'est-à-dire lorsqu'un message *Query-P* (ou un NH testé) explore par composition celui-ci. Un routeur r_θ $(1 \leq \theta < p)$ peut être visité à deux reprises ou plus dans la phase de validation sans être discriminant. Dans la mesure où r_θ active une ligne de routage pour un prédécesseur seulement si celle-ci est également active pour son trafic local, si le champ P contient r_θ alors que le candidat testé est égal à celui-ci, l'interface de sortie testée est ignorée.

La vague BFS déclenchée sur un routeur r_1 qui n'appartiendrait pas à *NH(s,s,d)* avec *p=1* (ou plus généralement si un routeur $r_2 = NH_k(r_1, d)$ n'appartient pas à $NH(r_1, s, d)$ car DT(1) a échoué sur le NH de rang k), explore, dans un rayon de *p-1* au maximum, la composition des NH-candidats

[3] Cela pourrait être possible avec une fonction de hachage appliquée à cet ensemble, comme avec le protocole BANANAS présenté dans la partie 1.4.2.3

pour tester la validité de proche en proche des prochains sauts candidats calculés avec DT. Si r_1, un routeur adjacent de s, ne satisfait pas la condition IIC pour $NH_k(r_1, d) = r_2$, il transmet un message *Query-P(s,d,c,q-1,*r_1*)* vers r_2 et attend en retour un message $response - P(s, d, c, q, P)$. Les messages $response - P$ contiennent un code $c \in \{LOOP,\ VALID,\ SKIP\}$, une profondeur de test q et le chemin testé P. Un routeur recevant un message $Query - P$ génère un code :

- $LOOP$: si un circuit a été détecté sur le routeur initiateur ou si la profondeur maximale est atteinte sans résultat $VALID$ ou $SKIP$.
- $VALID$: si la condition IIC est satisfaite sur l'ensemble de ses NH-candidats.
- $SKIP$: si un circuit est détecté sur un des routeurs appartenant au champ P d'un message $Query - P$.

Ces codes sont ordonnés de cette manière :

$$LOOP > VALID > SKIP$$

Lorsqu'un routeur $r_{i+1} = NH_j(r_i, d),\ 0 < j \leq k^+_{DT}(r_i, d)$ reçoit une requête *Query-P(s,d,c,q,P)* d'un routeur r_i, il se comporte selon l'algorithme 5. Cet algorithme nécessite l'emploi d'une structure contenant les codes correspondant aux DT-voisins testés pour un triplet (s, d, P) donné. Pour réaliser l'ensemble des tests de validation, chaque routeur r doit disposer d'une telle structure. Le dernier champ de cette structure à quatre dimensions contient autant de codes que de DT-voisins : pour une destination d donnée, $k^+_{DT}(r, d)$. Le triplet (s, d, P) désigne le routeur initiateur, la destination et le chemin de composition visité. La liste Lv est relative aux requêtes reçues : le premier champ indique la source émettrice, le second la destination, le troisième les routeurs déjà visités, et le dernier contient le code associé à un DT-voisin donné. La complexité de Lv est proportionelle au nombre de message *Query-P* en instance.

Algorithme 5 La vague avant : message $Query - P$

Procédure DT(p)-Query-P
($Query - P(s, d, c, q, P)$: **Message reçu sur** r_{i+1} **depuis** r_i)

s : **Routeur initiateur**

d : **Destination**

c : $C_1(s, d)$

q : **Profondeur restant à parcourir**

P : **Liste des routeurs visités**

Lv : **Liste des NHs testés**

Pour j **de** 1 **à** $k_{DT}^{+}(r_{i+1})$ **faire**

Selon que

$\mathbf{NH_j(r_{i+1}, d)}$ **vérifie la condition IIC** $(C_j(r_{i+1}, d) \leq c)$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = VALID$

$\mathbf{NH_j(r_{i+1}, d)} \in \mathbf{P}$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = SKIP$

$\mathbf{NH_j(r_{i+1}, d) = s}$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = LOOP$

envoyer $response - P(s, d, LOOP, p - q, P)$ à r_i

$\mathbf{q > 0}$: envoyer $Query - P(s, d, c, q - 1, P \cup r_{i+1})$ à $NH_j(r_{i+1}, d)$

$\mathbf{q = 0}$: envoyer $response - P(s, d, LOOP, p - q, P)$ à r_i

Fin Selon que

Fin Pour

Si $|Lv(s, d, P)| = k_{DT}^{+}(r_{i+1}, d)$ **Alors**

envoyer $response - P(s, d, max(Lv(s, d, P)), p - q, P)$ à r_i

Fin Si

Fin

Lorsqu'un routeur r_i reçoit un message $response - P(s, d, c, q, P)$ d'un routeur r_{i+1}, il se comporte selon l'algorithme 6. Dans ces messages, c désigne le contenu du code obtenu via les messages $Query - P$. Le paramètre q indique le nombre de sauts du chemin P testé entre la source s et l'émetteur r_{i+1} du

message $response - P$. Si $q = p$ alors la profondeur maximale a été atteinte et $r_{i+1} = r_p$.
Lorsqu'un routeur r_i retourne un message $response - P(s, d, max(Lv(s, d, P)), q - 1, P \setminus r_i)$ vers r_{i-1}, il retire son identifiant du chemin de composition $P : P \leftarrow P \setminus r_i$ $P(q-1) = r_i$ et $P(q-2)$ désigne le routeur en amont à qui transmettre le code obtenu. Sur un routeur r_1 adjacent à s, un message $response - P$ contenant un code $SKIP$ est ignoré : cela signifie que la seule alternative du NH testé sur r_2 est un retour sur le routeur adjacent r_1.

Algorithme 6 La vague retour : message $response - P$

Procédure DT(p)-response-P
($response - P(s, d, c, q, P)$: **Message reçu sur** r_i **depuis** r_{i+1})

c : **Code de validation** ($c \in \{LOOP, VALID, SKIP\}$)

q : **Distance en nombres de sauts depuis la source** ($q = i + 1$)

P : **Liste des routeurs visités**

Lv : **Liste des NHs testés**

Selon que

q > 2 : *[$r_i = P(q-1)$, il s'agit d'un routeur testé]*

Si $c = LOOP$ **Alors**
transmettre $response - P(s, d, LOOP, q-1, P \setminus r_i)$ à $P(q-2)$
Sinon
$Lv(s, d, P, r_{i+1}) = c$
Fin Si
Si $|Lv(s, d, P)| = k^+_{DT}(r_i, d)$ **Alors**
envoyer $response - P(s, d, max(Lv(s, d, P)), q-1, P \setminus r_i)$ à $P(q-2)$
Fin Si
q = 2 ∧ c = VALID : *[$r_i = r_1$, il s'agit d'un routeur adjacent à s]*

Si $\exists (r_1, r_2, d)_{r_1}$ activé **Alors**
activer $(s, r_2, d)_{r_1}$ et envoyer $response(d)$ à s
Sinon
enregistrer $(s, r_2, d)_{r_1}$
Fin Si
q = 1 : *[$r_i = s$, il s'agit du routeur initiateur]*

activer $(s, r_1, d)_s$

Si $\exists$ une ligne $(a, s, d)_s$ enregistrée **Alors**
activer $(a, r_1, d)_s$ et transmettre $response(d)$ à a
Fin Si
Fin Selon que
Fin

Lorsque r_i, un routeur testé ($i > 1$), dispose d'un code pour chacun de ses DT-voisins, obtenu localement ou par l'intermédiaire d'un message $response - P$, il calcule la réponse à transmettre en

retour, qui est le maximum des codes obtenus. Ce message est transmis à $P(q-2) = r_{i-1}$, le routeur en amont ayant relayé la requête *Query-P* via r_i.
L'ensemble des NHs activés sur un routeur r_{i+1} pour un routeur en amont r_i vérifie $NH(r_i, r_{i+1}, d) \neq \emptyset$ si et seulement si l'un des deux cas suivants est satisfait :

- r_i a reçu un message $response(d)$ via DT(1) depuis r_{i+1} (activation d'une ligne de routage $(r_i, r_{i+1}, d)_{r_i}$).
- r_{i+1} a reçu un code $VALID$ (via DT(p), $p > 1$, et un message $response - P$) depuis un routeur r_{i+2} pour le couple (r_i, d), et la ligne de routage $(r_{i+1}, r_{i+2}, d)_{r_{i+1}}$ est activée.

Par exemple, si r_1 reçoit un code $VALID$ de r_2 pour le couple (s, d), il valide une ligne de routage $(s, r_2, d)_{r_1}$. Cette ligne est activée et un message $response$ est transmis à s si $r_2 \in NH(r_1, r_1, d)$. L'activation d'une ligne de routage, et a fortiori d'une route, correspond à une composition de NH garantissant une stricte décroissance du meilleur coût à au plus $p + 1$ sauts.

Notations 1 (DT-espace entrant). *Le DT espace entrant d'un routeur r désigne, dans un rayon de p sauts, l'ensemble des caractéristiques topologiques orientées de son voisinage. Nous utiliserons les notations suivantes :*

- $|N^p_{DT}(r)|$ *: cardinal de l'ensemble des voisins distants d'au plus p sauts de r et générant des demandes de validation sur r.*
- $|N_{DT}(s, r)|$ *: nombre de destinations pour lesquels r peut recevoir une demande Query-P depuis s ($s \in N^p_{DT}(r)$).*
- $|P^p_{DT}(s, r)|$ *: cardinal de l'ensemble des compositions de p sauts sur les DT-voisins entre s et r.*

Avec ces notations, il s'agit de définir le nombre de requêtes potentielles que r peut recevoir de son voisinage. L'ensemble de ces termes dépend directement du nombre de NH-candidats sélectionnés par DT, les DT-voisins.
Les dimensions maximales de la structure Lv sont liées au caractéristique de l'espace défini par le sous-graphe de DT. Plus précisement, elles dépendent du DT-espace entrant sur un rayon de p sauts.
En pratique, Lv est une structure dynamique dont chaque entrée correspond à une demande émanant d'un routeur initiateur. Le nombre maximal d'entrées est borné par $|N^p_{DT}(r)|$. Si r recoit une requête depuis s, alors $Lv(s)$ comporte au plus $|N^p_{DT}(s, r)| \times |P^p_{DT}(s, r)| \times k^+_{DT}(r, d)$ codes.

2.2.2.2 Propriétés et preuve

Le processus de validation présenté dans le paragraphe précédent est indépendant du sous-graphe auquel il est appliqué. Néanmoins, la difficulté liée à la satisfaction de la règle de validation IIC est, à profondeur $p > 1$, dépendant du nombre de NHs évalués. Plus cet espace est réduit, ce qui est le cas avec DT, plus les chances de succès sont grandes.
La preuve donnée ci-après n'est pas dépendante de l'ensemble des NH-candidats généré. Par raccourci, nous considérerons, dans la preuve, que l'espace des NH-candidats est limité au DT-voisinage et au

processus de validation associé : DT(p). Le processus de validation DT(p) garantit l'absence de boucle de routage au niveau routeur.

Propriété 2 (DT(p) garantit un routage sans boucle). *DT(p) assure la suppression des circuits éventuellement générés par la composition de proche en proche des NHs calculés avec DT.*

Preuve :

La procédure de validation impose deux contraintes pour la formation d'une route $\mathfrak{R} = (r_1, ..., r_i, ..., r_m) \in R_m(s,d), \forall\ s,d \in N, \forall\ m < |N|$.

Chaque routeur $r_i \in \mathfrak{R}$ satisfait les deux propriétés suivantes ($\forall i \in [0,m]$) :

(a) Toutes les compositions, sur le DT-voisinage à p sauts au plus, entre r_i et la destination d (il s'agit des chemins testés : P), en ignorant les compositions de DT-voisins générant un code SKIP, génèrent un code VALID. Un routeur terminal d'une composition (celui permettant la génération d'un code VALID) $r_{i+q} \in P, 2 \leq q \leq p+1$, vérifie : $C_1(r_{i+q}, d) < C_1(r_i, d)$ et son prédecesseur satisfait $C_1(r_{i+q-1}, d) \leq C_1(r_i, d)$. Aucune composition de DT-voisins (aucun chemin P) ne génére un résultat LOOP pour le couple (r_i, d).

(b) $NH(r_i, r_{i+1}, d) \subset NH(r_{i+1}, r_{i+1}, d)$. Une ligne de routage activée sur un routeur r_{i+1} pour le trafic en transit provenant de r_i est nécessairement activée pour le trafic local de r_{i+1}.

La propriété (a) est récursivement combinée à la propriété (b) : si $NH_k(r_{i+q}, d) = r_{i+j}$, avec $1 \leq j < q \leq p$, alors le prochain saut $NH_k(r_{i+q}, d)$ correspond à un code SKIP. Ce code est ignoré au profit d'un code VALID si la propriété (a) est satisfaite ou d'un code LOOP si $\exists q \leq p \mid r_{i+q} = r_i$.

La propriété (a) découle directement de la procédure de validation DT(p) : d'une part, aucune composition de chemins à $p+1$ sauts ne peut explorer le routeur r_i, d'autre part, une ligne de routage est activé seulement si l'ensemble des chemins de compositions génèrent un code VALID vers r_{i+1} et une $response(d)$ vers r_i. La propriété (b) permet d'ignorer les codes SKIP : les compositions de chemins contenant un circuit dont s ne fait pas partie seront élaguées par les routeurs en aval. La réception d'un code VALID sur r_{i+1} signifie donc que toutes les compositions de chemins testés sont viables car l'extrémité terminale de la composition propose un coût optimal strictement inférieur à celui de s. De même la réception d'un message $response(d)$ par DT(1) signifie que r_{i+2} vérifie $C_1(r_{i+2}, d) < C_1(s, d)$.

Chaîne de coûts strictement décroissants :

La propriété (a) et la transitivité de l'opérande "<" implique l'existence d'une série de routeurs $r_{q(k)}, {}_{q(k) \leq m}$ disposant d'une route optimale de coût strictement décroissant par rapport au routeur précédent. Pour faire référence aux éléments de cette chaîne, nous utiliserons les notations suivantes :

$$(r_{q(0)} = s, r_{q(1)}, r_{q(2)}, \cdots, r_{q(k)}, \cdots, d = r_{q(n)}),\ 1 \leq (q(k) - q(k-1)) \leq p+1,\ q(k) \leq m$$

Chaque routeur $r_{q(k+1)}$ désigne le premier routeur, visité depuis $r_{q(k)}$ par le processus de validation, dont l'exploration a générer un résultat VALID ou l'émission d'un message $response(d)$ via DT(1). La propriété (a) implique $C_1(r_{q(k+1)}, d) < C_1(r_{q(k)}, d)$ pour $0 \leq k,\ q(k+1) \leq m$ ($r_{q(0)} = s$). La figure 2.5 illustre la structure d'une telle chaîne. L'existence de cette chaîne implique qu'il n'existe aucune

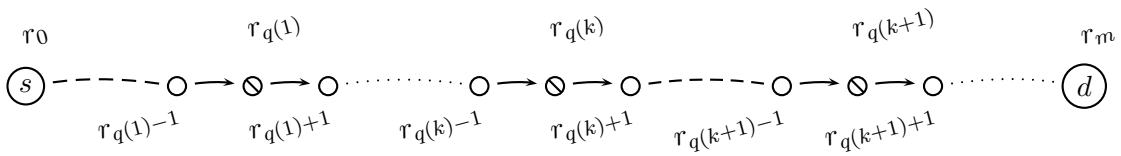


FIG. 2.5 – Chaîne de coûts strictement décroissants

route de coût infini. Or une route de coût fini ne peut pas comporter une boucle de routage au niveau lien. Que ce soit pour le trafic local ou pour le trafic en transit (grâce à la propriété (b)), une route de coût infini ne peut satisfaire un critère de stricte décroissance. A la différence de la règle LFI, la décroissance du meilleur coût est relâchée à $p + 1$ sauts au lieu de 1 seulement, mais le principe reste le même. En revanche, l'absence de boucle au niveau lien ne suffit pas à prouver l'absence de boucle au niveau routeur. La figure 2.6 illustre la forme d'une boucle au niveau routeur sur le nœud s. Une telle boucle peut exister sans qu'une route ne présente un coût infini. Bien que les boucles de routage au niveau routeur ne constituent pas des routes aux coûts infinis, elles peuvent entraîner une surconsommation inutile des ressources. Sur la figure 2.6, on peut observer qu'un flux émis depuis s traversant la boucle $(s, v, \cdots, a, s)$ gâche inutilement les ressources de ce segment de route et que le RTT peut être considérablement augmenté.

La garantie de l'absence de boucle au niveau routeur est assurée par la clause de la propriété (a) : "Le routeur $r_{q(k+1)}$ vérifie $C_1(r_{q(k+1)}, d) < C_1(r_{q(k)}, d)$ et son prédecesseur satisfait la condition : $C_1(r_{q(k+1)-1}, d) \leq C_1(r_{q(k)}, d)$". Cette propriété provient du mécanisme de validation : pour générer l'émission d'un message $response - P$ contenant un code VALID la condition IIC doit être satisfaite sur tous les DT-voisins dans un rayon de $p - q$ sauts. Toute route activée vérifie donc ce critère. Dans la suite, nous nous focaliserons sur l'absence de boucle au niveau routeur. La démonstration qui suit est basée sur un raisonnement par l'absurde et se contente d'envisager le cas où $r_i = r_0 = s$. Ce cas de figure est suffisant pour généraliser la preuve par récurrence grâce à la propriété (b). Ce raisonnement peut être étendu pour tout routeur $s \in N$ car les routeurs activent uniquement les prochains sauts valides pour leur trafic local. Ainsi, si une route de s vers d ne contient pas de circuit, tout routeur

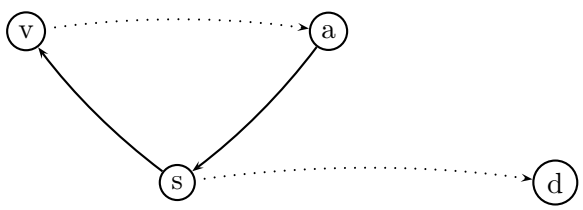


FIG. 2.6 – Une boucle de routage au niveau routeur

en amont de s utilisant ce segment de route ne peut être sujet à la présence de boucle de routage au niveau routeur.
Soit une paire de routeurs (s, d), supposons qu'il existe une route $\mathfrak{R}$ activée via le processus de validation DT(p) contenant un circuit sur le nœud s. Il existe alors un indice $i \geq 1$ tel que $r_i = s$.
Le routeur r_i peut prendre cinq positions en fonction de sa position par rapport à la chaîne de coûts strictement décroissants :

(i) Cas 0 : $i = q(k)$ pour $k \geq 1$
Cette position n'est pas viable dans la mesure où chaque élément de la chaîne vérifie $C_1(r_{q(k)}, d) < C_1(s, d)$. Le routeur s ne peut donc apparaître sur cette chaîne de routeurs.

(ii) Cas 1 : $1 < i < q(1)$
Ce cas est trivial et découle directement de la vague BFS initiée par s et déclenchée en r_1. Le processus de validation DT(p) explore tout le DT-voisinage de s dans un rayon de p sauts au maximum jusqu'à garantir la propriété (a). Cette exploration ne peut visiter s sans générer un résultat LOOP (LOOP>VALID).
Soit r_1 a reçu un code VALID de r_2 via le processus de validation en profondeur, soit r_1 a été validé avec DT(1) ($C_1(r_2, d) < C_1(s, d)$). Nous pouvons donc en conclure que $i > q(1)$.

(iii) Cas 2 : $q(k) + 1 < i < q(k + 1)$ pour $k \geq 1$
Le routeur r_i doit alors garantir dans un voisinage de p sauts au maximum la propriété (a) par rapport au routeur $r_{q(k)}$. Les messages $Query - P$, initiés par $r_{q(k)}$, ne peuvent pas visiter $r_{q(k)}$, sinon un code LOOP est généré. Un message $Query-P$ reçu par $r_i = s$ explorerait nécessairement $r_{q(1)}$: d'une part, celui-ci est dans le DT-voisinage à p sauts de s, d'autre part, aucun routeur en amont de $r_{q(1)}$ ne peut satisfaire la propriété (a) pour le coût $C_1(r_{q(k)}, d) < C_1(s, d)$ sans considérer $r_{q(1)}$. En effet, $r_{q(1)-1}$ (ou éventuellement $r_{q(1)}$ si $q(1) = 1$) est le routeur déclenchant la terminaison de la procédure de validation pour le chemin testé à l'étape (ii) alors que le coût évalué était dans ce cas strictement inférieur.
Par transitivité, on sait que $C_1(r_{q(1)}, d) > C_1(r_{q(2)}, d) > \cdots > C_1(r_{q(k)}, d)$. Récursivement, le routeur r_i a donc exploré $r_{q(2)} \cdots r_{q(k-1)}$ jusqu'à considérer le routeur $r_{q(k)}$ (code LOOP). Le cas (iii) est donc impossible.

(iv) Cas 3 : $i = q(k) + 1$ pour $k > 1$
Le routeur r_i est situé un saut en aval d'un routeur appartenant à la chaîne des $r_{q(k)}$ (voir $r_{q(k)+1}$ dans la figure 2.5). Il s'agit de la seule position où r_i n'est pas contraint de satisfaire la propriété (a).
Par construction, sur un DT-voisin de $r_{q(k)}$, les NH-candidats peuvent être activés un à un via DT(1). Ainsi, la propriété (a) n'est pas nécessairement satisfaite sur le routeur $r_{q(k)+1}$. Considérons le routeur situé en amont de $r_{q(k)}$, c'est-à-dire $r_{q(k)-1}$: celui-ci vérifie $C_1(r_{q(k)} - 1, d) \leq C_1(r_{q(k-1)}, d) < C_1(s, d)$ (propriété (a) par rapport au routeur $r_{q(k-1)}$). Il est impossible que $r_{q(k)-1}$ explore le DT-voisinage de $r_i = r_{q(k)+1}$ sans produire un résultat LOOP. De même qu'à l'étape (iii), $r_{q(k)-1}$ serait contraint de s'explorer lui même pour valider r_i. En effet, le NH de $r_q(k)$ correspondant à $r_i = r_{q(k)+1}$ propose nécessairement un coût strictement supérieur

à $C_1(r_i, d) > C_1(r_{q(k)-1}, d)$ car la valuation est strictement positive.
Le routeur $r_{q(k)-1}$ vérifie la relation $C_1(r_{q(k)-1}, d) = C_1(r_{q(k-1)}, d)$, ce qui est suffisant pour démontrer que $r_{q(k)+1} \neq s \ \forall k > 1$.

(v) Cas 4 : $i = q(1) + 1$
On sait que $r_{q(1)-1}$ vérifie $C_1(r_{q(1)-1}, d) = C_1(s, d)$ pour les mêmes raisons qu'à l'étape (iv), on en déduit alors la relation $C_1(r_{q(1)-1}, d) = C_1(r_{q(1)+1}, d)$. Si s dispose d'un ou plusieurs NH-candidats aux coûts non optimaux, la propriété (a) implique que $r_{q(1)-1}$ s'explore lui-même avec un message *Query-P* : deux sauts séparent $r_{q(1)-1}$ de $r_{q(1)+1}$ or l'exploration du DT-voisinage est exhaustive à partir de DT(2). En revanche, si tous les NH-candidats de s présentent un coût identique, alors cela implique $C_1(r_1, d) < C_1(s, d)$. On sait alors que $C_1(r_{q(1)-1}, d) \leq C_1(r_1, d)$ et donc $r_{q(1)+1} \neq s$ car $r_{q(1)-1}$ ne peut valider le NH de $r_{q(1)}$ passant par s.

Le routeur s ne peut apparaître dans aucune position satisfaisant les propriétés que le processus de validation impose. La relation $NH(a, s, d) \subset NH(s, s, d)$ nous permet de conclure qu'il n'existe pas de boucle de routage sur toute route $\mathfrak{R} \in R_m(s, d),\ \forall s, d \in N,\ \forall m < |N|$ que ce soit pour le trafic local de s ou pour le trafic de transit. En effet, tout routeur a en amont de s empruntant cette route, tel que $s \in NH(a, a, d) \wedge NH(a, s, d) \neq \emptyset$, et récursivement pour les routeurs en amont de a, bénéficient de cette garantie.

Les routeurs appartenant à la chaîne de coûts strictement décroissants jouent un rôle déterminant pour la cohérence du routage. Ils garantissent un acheminement sans boucle vers une destination donnée. Le coût d'acheminement diminue au fur et à mesure de l'aiguillage via ces routeurs. Entre deux éléments de la chaîne, le coût du chemin peut augmenter mais cette relaxation est contrôlée par l'exploration en profondeur vérifiant la propriété (a). La propriété (b) assure de proche en proche la cohérence du routage.

Cette démonstration reste valable si la profondeur d'exploration p choisie par chaque routeur diffère. En effet, la démonstration ne s'appuie pas sur un critère de profondeur global. Dans la partie 2.3, nous étudierons également un mode coopératif pour les routeurs n'implémentant pas DT(p).

La démonstration reste également valide quel que soit l'algorithme de calcul de chemins sous-jacent. Le processus de validation est indépendant de l'algorithme de calcul des NH-candidats. La seule contrainte pour que les critères de validation soient corrects se résume ainsi : les lignes de routages actives sont choisies parmis les NH-candidats évalués par le processus de validation, si les NH-candidats ne sont pas connus, alors seuls les meilleurs chemins sont activés et testés lors de la validation en profondeur.

2.2.3 Optimisation de la validation

Le processus de validation en profondeur peut être optimisé pour, d'une part, réduire le nombre de messages de validation, et d'autre part, accroître la diversité des chemins activés. Cette optimisation est possible grâce à une propriété topologique spécifique : le meilleur chemin du routeur adjacent satisfait la condition IIC. Pour intégrer cette optimisation à notre processus de validation, il suffit de modifier le comportement de DT(1). Deux types de messages de validation doivent coexister selon la destination considérée. Le message $Query' - P$ est une variante simplifiée du message $Query - P$: la condition

IIC est uniquement vérifiée sur le meilleur NH. La relaxation du critère de validation permet également de minimiser la complexité d'exploration. Le meilleur coût d'un DT-voisin est connu localement. La condition IIC est vérifiée en utilisant une vision à deux sauts.
Soit v un DT-voisin d'un routeur s initiant une validation à profondeur p pour la destination d. Si $C_1(v,d) \leq C_1(s,d)$ alors la requête $Query' - P$ est relayée sur les NHs $NH_j(v,d)$ si $C_j(v,d) - w(v, NH_j(v,d)) > c$. Les champs des messages $Query - P$ et $Query' - P$ sont identiques. En pratique, ces deux types de requêtes sont différenciées par un bit indiquant pour une destination donnée si $C_1(v,d) \leq C_1(s,d)$. L'algorithme d'initialisation de $Query' - P$ est donnée dans l'algorithme 7. A la réception d'un message $Query' - P$, seul le meilleur NH-candidat est testé : pour produire un code VALID, il doit garantir un coût strictement inférieur à c.

Algorithme 7 Les messages *Query'-P* : initialisation

Procédure Validation à profondeur 1 (avec $Query' - P$)
($query(d, c, p)$: **message reçu par** v **depuis** s)
Si $C_1(v, d) \leq c$ **Alors**
valider et activer la ligne $(s, NH_1(v, d), d)$
envoyer $response(d)$ à s
Pour j **de** 2 **à** $k^+_{DT}(v, d)$ **faire**
Selon que
$\mathbf{C_j(v, d) - w(v, NH_j(v, d)) \leq C_1(v, d) < c}$:
valider et activer la ligne $(s, NH_j(v, d), d)$
$\mathbf{C_j(v, d) - w(v, NH_j(v, d)) < c}$:
valider et enregistrer la ligne $(s, NH_j(v, d), d)$
$\mathbf{p > 1}$: envoyer *Query'-P(s,d,c,p-1,(v))* vers $NH_j(v, d)$
Fin Selon que
Fin Pour
Sinon
Pour j **de** 1 **à** $k^+_{DT}(v, d)$ **faire**
Si $C_j(v, d) - w(v, NH_j(v, d)) < c$ **Alors**
valider et enregistrer la ligne $(s, NH_j(v, d), d)$
Sinon
Si p > 1 Alors
envoyer *Query-P(s,d,c,p-1,(v))* vers $NH_j(v, d)$
Fin Si
Fin Si
Fin Pour
Fin Si
Fin

Soit une route $\mathfrak{R} = (r_1, \cdots, d)$ activée depuis un routeur s via le processus de validation DT(p). A la réception d'un message $Query' - P$ sur un routeur r_{i+1}, la génération d'un code $VALID$ est favorisée par le test de validation effectué. L'algorithme 8 illustre le comportement d'un routeur r_{i+1} à la réception d'un message $Query' - P$. Le routeur r_{i+1} utilise la majoration $C_1(r_{i+2}, d) \leq C_j(r_{i+1}, d) - w(r_{i+1}, r_{i+2}) \mid NH_j(r_{i+1}, d) = r_{i+2}$ pour connaître le meilleur coût de ses DT-voisins sans avoir à les explorer comme c'est le cas avec les messages $Query - P$. Il s'agit du même mécanisme de validation *avec une vision à*

deux sauts que celui utilisé par DT(1).

Algorithme 8 Le message $Query' - P$ $(C_1(r_1, d) \leq C_1(s, d))$

Procédure DT(p)-Query'-P
($Query' - P(s, d, c, q, P)$: **message reçu sur** r_{i+1} **depuis** r_i)

Lv : **Liste des NHs testés**

Si $C_1(r_{i+1}, d) < c$ **Alors**

r_{i+1} transmet $response - P(s, d, VALID, p - q, P)$ à r_i

Fin Si

Pour j **de** 2 **à** $k^+_{DT}(r_{i+1})$ **faire**

Selon que

$\mathbf{NH_j(r_{i+1}, d)}$ **vérifie** $C_j(r_{i+1}, d) - w(\{r_{i+1}, NH_j(r_{i+1}, d)\}) < c$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = VALID$

$\mathbf{NH_j(r_{i+1}, d)} \in \mathbf{P}$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = SKIP$

$\mathbf{NH_j(r_{i+1}, d)} = \mathbf{s}$:

$Lv(s, d, P, NH_j(r_{i+1}, d)) = LOOP$

envoyer $response - P(s, d, LOOP, p - q, P)$ à r_i

$\mathbf{q > 0}$: envoyer $Query' - P(s, d, c, q - 1, P \cup r_{i+1})$ à $NH_j(r_{i+1}, d)$

$\mathbf{q = 0}$: envoyer $response - P(s, d, LOOP, p - q, P)$ à r_i

Fin Selon que

Fin Pour

Si $|Lv(s, d, P)| = k^+_{DT}(r_{i+1}, d)$ **Alors**

envoyer $response - P(s, d, max(Lv(s, d, P)), p - q, P)$ à r_i

Fin Si

Fin

L'absence de boucle de routage est également vérifiée au niveau routeur avec les messages $Query' - P$. La propriété (a) énoncée dans le paragraphe précédent est étendue. La validation en profondeur peut se contenter de satisfaire le critère suivant, noté propriété (a') : *Si* $C_1(r_{i+1}, d) \leq C_1(r_i, d)$*, alors toutes les compositions de chemins sur le DT-voisinage à* $p - 1$ *sauts de* r_{i+1} *garantissent, en ignorant les compositions générant un code SKIP, un meilleur coût strictement décroissant :* $C_1(r_{i+q}, d) < C_1(s, d)$*. Aucune composition de NH-candidats ne doit générer un résultat LOOP pour le couple* (r_i, d)*.*

La preuve de l'absence de boucle au niveau routeur nécessite alors deux clauses supplémentaires :
Pour le cas (iv), $i = q(k) + 1$, il est possible que la condition $C_1(r_{q(k-1)+1}, d) \leq C_1(r_{q(k-1)}, d)$ suffise à assurer la validation de $r_{q(k)+1}$. Si la propriété (a) n'est pas satifaite, on sait alors que : $\exists l$ $(1 \leq l \leq k^+_{DT}(r_{q(k)-1}, d)$ tel que $C_l(r_{q(k)-1}, d) > C_1(r_{q(k-1)}, d)$. Si $C_1(r_{q(k-1)+2}, d) > C_1(r_{q(k-1)+1}, d)$ alors la propriété (a) garantit qu'il existe un routeur en aval de $r_{q(k-1)+2}$ pour lequel chacun de ses DT-voisins présente un coût inférieur ou égal au meilleur de $r_{q(k)-1}$, et ce, sans qu'aucune composition de chemins ne puisse générer un code LOOP. Cette propriété ne peut être satisfaite par $r_{q(k)}$ si son DT-voisin vérifie $r_{q(k)+1} = s$.
Par induction sur la propriété (b), on en conclut que la série de routeurs $r_{q(k-1)}, \cdots, r_{q(k)-1}$ vérifie :

$$C_1(r_{q(k-1)+1}, d) = C_1(r_{q(k-1)+2}, d) = \cdots = C_1(r_{q(k)-2}, d) = C_1(r_{q(k)-1}, d)$$

Or, soit la propriété (a) implique que $C_1(r_i, d) = C_1(r_{q(k-1)}, d)$. Soit la propriété (a') implique que $C_1(r_i, d) < C_1(r_{q(k-1)}, d)$. Ces deux relations ne peuvent pas être satisfaites car $C_1(s, d) > C_1(r_{q(k-1)}, d)$ pour $k \geq 2$.
Pour le cas (v), la propriété (a') ne peut pas être vérifiée sur $r_{q(1)+1}$ car on sait, pour les mêmes raisons que celles vérifiées dans (iv), que $C_1(s, d) > C_1(r_{q(k-1)}, d)$. Par conséquent, le routeur $r_{q(1)-1}$ serait contraint de s'explorer lui-même via un message *Query-P* et donc de générer un code LOOP.
Les autres cas ne sont pas affectés par le comportement de la procédure de validation optimisée.

2.2.4 Simplification et exemples

Après plusieurs simulations sur nos réseaux d'évaluation[4], il est apparu évident qu'un nombre trop élevé de NH-candidats ne favorise pas la procédure de validation à profondeur $p > 1$. Le nombre de routes est important mais la diversité de ces routes au sens de leur intersection en liens est faible. Plus l'algorithme de calcul de chemins sélectionne de NH-candidats, plus il sera difficile pour le processus de validation en profondeur, de satisfaire la propriété (a).
Il est possible de modifier Dijkstra Transverse de manière à obtenir une couverture plus importante et réduire la complexité en termes de messages et d'espace de mémorisation. Pour cela, DT doit sélectionner les NH-candidats proposant un coût optimal et une seule alternative sous-optimale si elle existe. La complexité d'exécution de DT est réduite car la mise à jour de la matrice Mc est inutile dès lors qu'une alternative existe pour une destination donnée.
Cette pré-sélection garantit, dans la plupart des cas, l'existence d'au moins une alternative de routage. L'ensemble des routes ainsi validées par cette sélection plus fine n'est pas nécessairement un sous-ensemble des routes validées avec l'utilisation de DT sans modification et vice-versa. Sur la figure 2.7, on peut constater que le nombre de prochains sauts calculés sur r_2 conditionne les chances de succès pour la validation du couple de routeurs (s, d). Les valuations représentées sur la figure correspondent aux coûts des chemins via le NH considéré. Le routeur s ne peut valider le NH sur r_1 car celui-ci ne propose pas un coût conforme à la règle IIC. Sur r_2, en revanche, la validation dépend des NHs

[4]L'amélioration est notée DT+(p) dans le chapitre **Outils et résultats de simulations**.

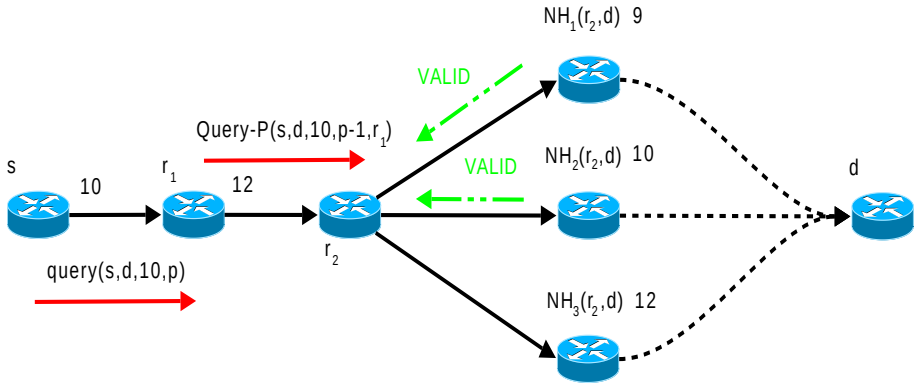


FIG. 2.7 – Validation et NH-candidats

sélectionnés par DT : si seuls les deux meilleurs sont retenus comme NH-candidat, alors la validation à profondeur 2 suffit. Par contre, si le prochain saut candidat $NH_3(r_2, d)$ est enregistré dans $Mc(r_2)$, alors la validation échoue si $p - 1 = 0$ ou continue en profondeur sinon.

Avec cette restriction sélective la probabilité d'activation des NH-candidats est empiriquement plus élevée. Plus particulièrement dans les zones où la topologie est faiblement maillée, la validation est favorisée. De plus, nous avons constaté que la validation en profondeur permet de valider des routes plus disjointes assurant une bande passante cumulative plus importante. L'objectif de cette modification est d'augmenter la probabilité d'existence d'au moins une alternative locale sur chaque routeur. La diversité des chemins est mieux répartie et la couverture est plus importante en termes de protection bien que certains prochains sauts ne soient pas évalués.

Dans la suite, parmi l'ensemble des améliorations et des optimisations proposées, nous avons seulement considéré la simplification de DT pour nos simulations.

Dans la série d'exemples qui suit (voir figure 2.8), basés sur le réseau présenté dans la figure 2.2(a), nous considérons par simplification, la valuation des liens uniforme. Ainsi le coût d'un chemin est déterminé par le nombre de sauts entre la source et la destination.

Pour commencer, analysons les chemins validés par le routeur 4 pour son trafic local en admettant que DT sélectionne (enregistrement dans Mc) toutes les alternatives calculées (sans modification sur DT car inutile pour DT(1)). Dans la première série d'exemples, la profondeur de recherche maximale sera limitée par $p = 1$. Procédons par ordre lexicographique sur l'identifiant de destination pour les trois premières destinations ($d \in \{1, 2, 3\}$) :

($d = 1$) Sur la figure 2.8(a), le routeur 4 dispose de trois NH-candidats, 2, représentant le meilleur chemin calculé, 5 représentant un chemin de coût 3, et 6 pour un chemin de coût surévalué 4 au lieu de 3. Seul le meilleur chemin via 2 est activé d'office, mais DT(1) explore chaque voisin pour valider les NHs alternatifs chez ses voisins pour le trafic en transit. Sur 2, DT(1) valide et active directement le $NH_1(2, 1) = 1$ et $NH_2(2, 1) = 3$ car $C_1(2, 1) = 1 < C_1(4, 1) = 2$ et $C_2(2, 1) - w(2, 3) < C_1(4, 1)$ pour 4 en entrée. Sur 5, DT(1) valide et active directement uniquement $NH_1(5, 1) = 2$ pour le trafic en transit provenant de 4. Sur 6 seul le $NH_1(6, 1) = 3$ est validé et activé directement pour

l'interface 4.

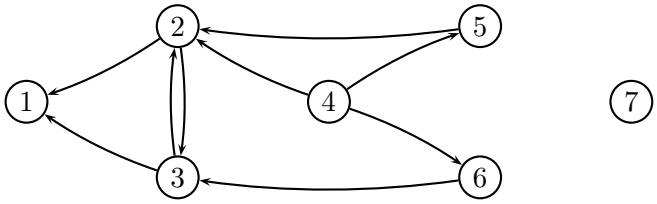
($d = 2$) Sur la figure 2.8(b), le routeur $NH_2(4,2) = 5$ active $NH_1(5,2) = 2$ pour le routeur 4 en entrée. Sur $NH_3(4,2) = 6$, seul $NH_1(6,2) = 3$ pourrait être validé mais $C_1(6,2) = 2 > C_1(4,2) = 1$.

($d = 3$) De même que pour $d = 1$, $NH_1(4,3) = 2$ valide et active automatiquement ses deux NHs $NH_1(2,3) = 3$ et $NH_2(2,3) = 1$ pour 4. Sur 5, seul le $NH_1(5,3) = 2$ est activé pour 4 en entrée. Sur $6 = NH_3(4,3)$, le $NH_1(6,3)$ est activé pour le trafic en transit de 4.

La table de routage sur le routeur 4 et vers la destination 1 prend la forme donnée dans le tableau 2.4 pour le trafic local. Grâce à DT(1), on remarque que 4 peut réajuster son coût vers 1 via 6 comme indiqué dans la troisième ligne de routage. Les lignes de routage activées avec DT(1) sur le voisin 2 prennent la forme représentée par le tableau 2.5 trafic de transit pour $ie = 4$ et vers la destination 1. A titre de comparaison, la condition LFI aurait seulement permis d'utiliser une seule route de 4 vers 1 (la première dans la liste ci dessous), alors que DT(1) permet d'en utiliser six : $R_2(4,6) = \{(2,1)\}$, $R_3(4,6) = \{(2,3,1),(5,2,1),(6,3,1)\}$ et $R_4(4,6) = \{(5,2,3,1),(6,3,2,1)\}$. Ces six routes peuvent aussi être désignées sous la forme d'une suite d'indices correspondant au rang des NHs traversés : $(1,1)$, $(1,2,1)$, $(2,1,1)$, $(3,1,1)$, $(2,2,1,1)$ et $(3,1,2,1)$.
Les figures 2.8(a) et 2.8(b) illustrent la distribution possible des flux avec DT(1), respectivement entre 4 et 1 et entre 4 et 2. Reprenons le même exemple pour illustrer le fonctionnement et la diversité accrue des chemins activés par DT(3). A destination de 2, le routeur 6 n'avait pu assurer un coût inférieur ou égal pour le trafic en transit de 4 car $C_1(6,2) > C_1(4,2)$. Le routeur 6 dispose de deux NH-candidats ($NH_1(6,2) = 3$ et $NH_2(6,2) = 4$, le coût des deux chemins correspondants est égal à 2) pour la destination 2 grâce au calcul de DT.
Le chemin via 4 génère un code $LOOP$ et ne peut être utilisé pour le trafic provenant de 4. 6 déclenche donc une vague d'exploration BFS vers son premier NH, le routeur 3. Celui-ci, selon la procédure $Query - P$ à profondeur 3, peut stocker un code $VALID$ pour $NH_1(3,2) = 2$, un code $SKIP$ pour $NH_3(3,2) = 6 = P[1]$ et doit transmettre un message $Query - P(4,2,1,1,P = (6,3))$ à $1 = NH_2(3,2)$. Le routeur 1 peut lui directement conclure car son premier NH-candidat, $NH_1(1,2)$, vérifie $C_1(1,2) = C_1(4,2) = 1$ (code $VALID$) et son second NH-candidat vérifie $NH_2(1,2) = P[2] = 3$ (code $SKIP$). Or $max(SKIP, VALID) = VALID$, et le routeur 1 peut alors retourner le message *response-P*$(4,2,VALID,2,(6,3,1))$ vers 3.

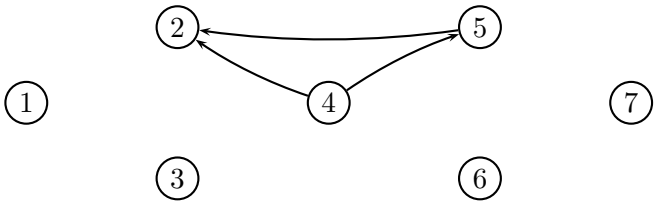
ie	Dest	NH	Coût
4	1	2	2
4	1	5	3
4	1	6	3

TAB. 2.4 – Table de routage du routeur 4



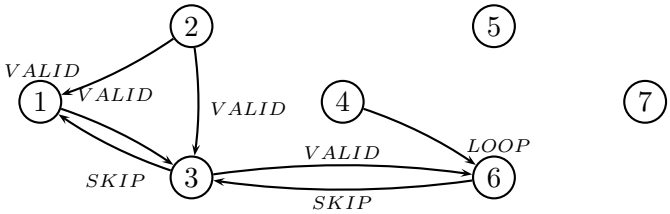
DT(1) : 3 NHs activés et 6 routes

(a) $s = 4, d = 1$



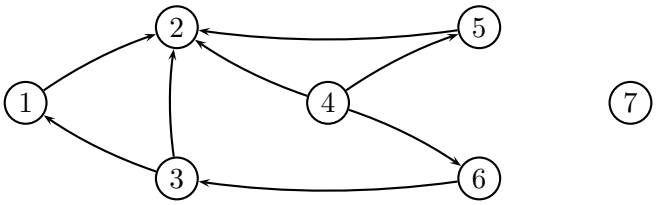
DT(1) : 2 NHs activés et 2 routes

(b) $s = 4, d = 2$



DT(3) : un exemple de vague BFS relayé par 6

(c) $s = 4, d = 2$



DT(3) : 3 NHs activés et 4 routes

(d) $s = 4, d = 2$

FIG. 2.8 – Illustration de DT(p)

ie	Dest	NH	Coût
4	1	1	1
4	1	3	2

TAB. 2.5 – Table de routage du routeur 2

Celui-ci peut alors conclure, en considérant pour simplifier la liste d'états Lv statique : $Lv[4][2][P][2] = VALID$ et $|Lv[4][2][P]| = k^{+}_{DT}(3,2) = 3$, ainsi 3 envoie *response-P*$(4, 2, VALID, 1, (6, 3))$ à 6. La ligne de routage $(4, 3, 2)_6$ y est activée directement car il s'agit du meilleur NH de 6 à destination de 2, et 6 émet un message *response*(2) à 4. A la réception de cette information, celui-ci active la ligne de routage $(4, 6, 2)_4$ pour son trafic local (et éventuellement pour tous les routeurs en amont a, tel qu'il existe une ligne $(a, 3, 2)_4$ enregistrée mais non activé).
La figure 2.8(c) illustre le mécanisme d'exploration BFS avec DT(3) pour le couple $(s, d) = (4, 2)$. La figure 2.8(d) donne une représentation des routes activées entre 4 et 2 grâce à DT(3).

2.3 Implémentations

Cette partie donne des éléments de mise en œuvre pour nos procédures, DT et DT(p), dans un environnement réel. Nous y présenterons les aspects techniques liés à l'implémentation de ces algorithmes ainsi que les difficultés induites par le passage à l'échelle dans le contexte d'un déploiement interdomaine.

2.3.1 Mise en œuvre possible

Comme suggéré dans la partie 2.1, DT peut être utilisé comme un algorithme SPF amélioré. DT ne calcule ni un arbre ni un DAG mais un graphe non acyclique orienté permettant de prendre en compte des chemins multiples aux coûts inégaux entre une racine de calcul et les autres nœuds du graphe. La majoration des meilleurs coûts via chaque voisin contenu dans la matrice de coût peut être utile pour déterminer le meilleur coût des DT-voisins. Dans le contexte d'un déploiement léger, le processus de validation distribué peut s'avérer trop lourd aussi bien en termes de messages qu'en termes de complexité de calculs et de mémorisation d'états.
DT permet, simplement en soustrayant la valuation de l'arc correspondant au prochain saut, de connaître une valeur majorée du meilleur chemin de chaque voisin. Pour une destination d donnée, le meilleur coût d'un DT-voisin $v = NH_j(s, d)$ d'un routeur s vérifie la relation (notée MAJ) :

$$C_1(v, d) \leq C_j(s, d) - w(s, v)$$

Si le meilleur chemin du voisin v passe par s alors il se peut que DT ne l'enregistre pas. C'est le cas si v ne possède qu'un seul chemin primaire. Dans ce cas, par définition on a $C_1(v, d) > C_1(s, d)$. Cette inégalité implique la non satisfaction des règles IIC et LFI.

Grâce à la majoration MAJ (la valeur calculée n'est pas exacte car les arcs internes ne sont pas pris en compte), s est capable d'utiliser localement la condition IIC, LFI ou SPD. Cependant, si la nature du réseau permet une exploration plus profonde, il est plus judicieux d'utiliser DT(1). Notre processus de validation peut en effet sonder les voisins adjacents pour obtenir les coûts réels et donc valide davantage d'alternatives de routage. Si le réseau est capable de supporter une charge en messages et en calcul légèrement plus élevée et que la diversité des chemins activés localement n'est pas satisfaisante, DT(p), pour $p > 1$, peut être déclenché sur les routeurs adjacents, là où DT(1) a échoué.
Dans le pire des cas, le nombre de messages générés ($Query - P$ et $response - P$) par une source s est majoré par la simplification[5] proposée dans le paragraphe 2.2.4 :

$$2 \times \sum_{q=1}^{p} 2^q \times |N| = |N| \times (2^{p+2} - 4)$$

En pratique, on peut réduire cette complexité en agrégeant les messages sur l'ensemble des destinations, au moins sur la vague avant. D'une part, le nombre de destinations pour lesquelles la validation est non satisfaite est considérablement réduit de proche en proche. D'autre part, un seul message peut agréger l'ensemble des destinations pour lesquelles le premier saut déclencheur de l'approfondissement n'a pas été validé par DT(1). Chaque destination d se voit affecté un coût ($C_1(s, d)$) et un bit indiquant la nature de la requête ($Query - P/Query' - P$).
Pour la vague retour, afin d'accélérer la vitesse de convergence du processus de validation, il peut être préférable de ne pas réaliser d'agrégation. Certaines destinations nécessitent une profondeur de recherche plus importante que d'autres, d'où des disparités en termes de temps d'attente selon les destinations. Il faut que l'ensemble des codes d'une entrée (s, d, P) de la structure Lv soient renseignés pour générer un message $response - P$.
En fonction des capacités de chaque routeur et pour un déploiment progressif, la cohabitation entre routeurs implémentant DT(p) (quel que soit p) et des routeurs SPF (ou ECMP) est tout-à-fait possible. Dans le cas d'une coopération SPF-DT, un routeur DT peut utiliser un routeur adjacent SPF en utilisant des critères de validation locaux avec la relation MAJ.
En termes d'implémentation, l'algorithme DT utilise des structures dynamiques. Les dimensions des objets utilisés évoluent avec les notifications topologiques. Les destinations et les voisins connus sont indexés au fur et à mesure de l'apprentissage de la topologie au moyen d'une table d'indirection. En pratique, la matrice Mc est indexée selon ces tables d'indirection. Il est donc nécessaire d'utiliser un ensemble de vecteurs d'indirection pour atteindre l'indice d'indexation souhaité.
Par ailleurs, dans la phase de validation en profondeur pour $p > 1$, le nombre d'états à analyser (Lv est une structure dynamique) peut être réduit grâce à la simplification de DT développée dans le paragraphe précédent. L'utilisation d'une fonction de hachage, pour mémoriser le chemin P testé (comprenant au maximum p sauts), permet à Lv de se contenter d'un seul champ pour renseigner un chemin testé P. Avec une fonction de hachage minimisant les collisions, il est possible d'éviter l'utilisation de p champs dans la liste dynamique Lv. Le champ P correspond à la projection des p éléments du chemin testé.

[5] En considèrant que DT ne sélectionne qu'un seul prochain saut alternatif par destination

En considérant que DT ne sélectionne que deux NHs par destination, une source initie au pire 2^p tests de composition de chemins et sollicite au maximum $2^{p+1} - 4$ routeurs sur le DT-voisinage à p sauts. De cette manière, pour une destination donnée, la complexité en messages n'est pas dépendante du degré des routeurs explorés ni de la taille du réseau.

2.3.2 Stabilisation

L'ensemble des algorithmes, définitions et preuves données dans ce qui précède sont valables si le réseau est dans un état stable.

Definition 13 (**Etat stable**). *Un réseau, ou plus généralement une aire de routage, est dans un état stable si l'ensemble de ses routeurs dispose d'informations topologiques identiques.*

En pratique, chaque routeur obtient, via la diffusion des états des liens, la même série d'annonces LSA avec des numéros de séquence identiques. Cette notion de stabilité est liée à la convergence de la procédure de notification. Un changement topologique est causé par l'apparition ou le retrait d'un lien ou d'un routeur. On peut considérer que la fréquence de ces changements est relativement faible, et que la durée entre deux modifications consécutives est supérieure au temps de convergence. Tous les routeurs obtiennent en un temps limité et dépendant des dimensions du réseau, en particulier son diamètre, les mêmes informations topologiques.
Parvenir à un état stable est donc possible lorsque les modifications topologiques sont relativement rares, ce qui est le cas dans un réseau filaire. La commutation mise en œuvre avec nos procédures doit simplement s'assurer que les messages de signalisation (les LSA et les messages de validations de DT(p)) générés soient transmis sur la même route afin de garantir que les informations soit relayées par ordre de parution. En pratique, ces messages de signalisation seront toujours transmis par le même NH, celui associé à la route de meilleur coût, sur chaque routeur. DT(p) s'assure également que l'information périmée et que les calculs devenus obsolètes soient ignorés et arrêtés grâce à un système de numérotation des messages de validation en fonction de la source émettrice : (s, n), où s est l'identifiant de la source et n, le numéro de séquence. Si un routeur reçoit un message de validation $Query - P$ estampillé $(s, n + 1)$, il peut réinitialiser la structure contenant les NHs testés ($Lv \leftarrow \emptyset$) et ignorer les messages $response - P$ estampillés (s, n') si $n' < n + 1$.
Il faut également vérifier que les tests de validité inter-voisinage soient réalisée sur une base d'information topologique cohérente. Il se peut que le message de validation suivant la réception d'un LSA soit reçu par le routeur voisin avant la fin du calcul de ses NH-candidats (en particulier lorsque les puissances de calcul des différents routeurs sont hétérogènes). Il est nécessaire d'attendre la fin de la phase de calcul avant de vérifier la validité des chemins. Pour cela, le message de validation est mis en file d'attente jusqu'à ce que la phase de calcul de chemins soit terminée.
Il faut également s'assurer que le routeur voisin ne dispose pas d'annonces topologiques supplémentaires lors de la phase de validation. Dans ce cas, les coûts évalués ne sont pas conformes aux règles de validation. Ces différences peuvent aboutir à une vision du réseau incohérente. Un routeur peut activer une ligne de routage pour ses prédécesseurs seulement s'ils disposent de la même vision topologique. Afin

de prévénir les incohérences de routage, les routeurs ignorent les messages de validation ne prenant pas en compte le dernier LSA reçu. Les messages de validation à profondeur 1 contiennent une information de type (o, l) où o désigne le routeur d'origine émetteur de LSA et l, le numéro de séquence associé. Ainsi à la réception d'un message de validation adjacent estampillé (o, l), si le dernier LSA reçu ne correspond pas, alors ce message est ignoré jusqu'à ce que le voisin émetteur prenne note de la nouvelle modification sur o.
De plus, pour répondre à cette problématique de synchronisation, la réception d'un LSA entraîne la suppression de toutes les entrées pour le trafic local correspondant à l'ancienne table de commutation dès lors que les nouvelles entrées locales sont à jour. Durant la période de convergence, les prochains sauts affectés par une panne sont désactivés au profit des prochains sauts alternatifs si ils en existent. La réception d'un message de validation depuis un routeur adjacent supprime les entrées correspondant à l'interface par laquelle le message est reçu (trafic de transit).
Si la table de routage ne contient pas d'entrée pour un routeur en amont donné, alors que celui-ci lui envoie néanmoins du trafic, le commutation utilise les meilleurs NHs jusqu'à ce que le routeur en amont positionne ses nouvelles entrées. Le routage sera cohérent et multichemins dès lors que le routeur en amont réceptionnera le dernier LSA et transmettra au routeur concerné son message de validation.
La formation de boucles de routage transitoires est néanmoins possible de la même manière qu'avec un routage monochemin usuel. Pour la validation à un saut, la stabilisation est basée sur les numéros de séquence des LSA émis par le routeur déclencheur, c'est-à-dire le routeur constatant le changement topologique. Pour la validation en profondeur, les messages *Query-P* sont estampillés par un numéro de séquence associé à la source.

2.3.3 Extensibilité et routage interdomaine

Les algorithmes présentés jusqu'ici concernaient le routage intradomaine. Nous avons considéré que nos procédures présentaient une complexité en temps et en mémoire liée au nombre de destinations du domaine. Le routage interdomaine pose d'autre questions en termes d'extensibilité.
Pour un passage à l'échelle de l'Internet, nos procédures, et les tables de commutation produites, ne sont pas dépendantes du nombre de destinations possibles dans l'Internet. Après filtrage, les routeurs de bordure redistribuent les annonces de préfixes externes en interne. Ainsi, s'il existe plusieurs routeurs de bordure (*Gateway Router*, GWR) qui, pour un préfixe donné, proposent plusieurs meilleurs routes en terme de nombre d'AS traversés (pour éviter le risque de bouclage entre domaine), multiroutage inter et intradomaine peuvent être combinés.

2.3.3.1 Unicité de la correspondance : un seul routeur de sortie

Un protocole de routage comme OSPF met en place sa table de commutation globale en trois étapes successives selon le type de routes considéré : routage intra-zone, inter-zones et routage interdomaines. Pour simplifier, si l'on néglige le routage inter-zones, alors la combinaison entre routes internes et externes utilise une *correspondance* d'adressage sur les routeurs de bordures de domaine. Cette correspondance se caractérise par l'ajout de liens reliant, virtuellement et directement, les routeurs de

bordures aux préfixes annoncés, lors de la mise en place du routage interdomaine sur les routeurs internes.
Nos procédures considèrent uniquement les destinations internes appartenant au domaine. Par conséquent, le routage ne peut bénéficier, en l'état, que de la diversité des chemins en interne. Pour tirer partie des plusieurs routes externes vers un préfixe donné, il faudrait que celui-ci soit traité comme une destination interne dans la phase de validation. Cette approche n'est pas envisagée dans la mesure où cela nous confronte à un problème d'extensibilité lié au nombre de préfixes annoncés.
En pratique, de la même manière qu'avec OSPF, les destinations externes sont perçues en interne comme directement attachées par des liens fictifs aux routeurs de bordures. Nous proposons d'utiliser un principe analogue, assimilable à une table de correspondance entre les préfixes annoncés et les routeurs de bordure redistribuant les annonces.
La correspondance *préfixe→routeur de bordure* est obtenue par redistribution du routage BGP en interne. La table de correspondance doit être globale, tel que chaque routeur détermine, avec un ordre lexicographique et la métrique utilisée, une seule et même association {*préfixe, routeur de bordure*} possible. Ainsi, nos procédures ne dépendent, en termes de complexité, que de la taille du domaine. Le choix de la correspondance doit cependant être unique. L'exemple 2.6 illustre la nécessité d'une correspondance vérifiant ce critère d'unicité. Sur cet exemple, la composition multiroutage inter/intradomaine vers le préfixe Pr_2 peut provoquer la formation de boucle de routage. En effet, si l'association {*préfixe,* GWR} n'est pas unique et globale à l'ensemble du réseau, comme cela pourrait être possible si deux routeurs de bordure redistribuent en interne le même préfixe (sur l'exemple, Pr_2), les routeurs disposant de plusieurs prochain sauts vers les routeurs de bordures peuvent induire un routage incohérent. Il suffit, sur cet exemple, que le routeur NH_3 choisisse une association {*préfixe,* GWR} différente de celle d'un de ses routeurs en amont pour qu'une boucle de routage apparaîsse. En effet, si le routeur de bordure choisie est accessible via le routeur en amont, alors la commutation est incohérente, et une boucle de routage se crée sur NH_3.
Cette solution présente l'avantage d'être simple et de profiter pleinement de la diversité des routes intradomaine. En revanche, pour un préfixe donné, seul un unique routeur de sortie peut être utilisé par l'ensemble du domaine : la diversité des chemins inter-AS est négligée.

2.3.3.2 Partitionnement du domaine : plusieurs routeurs de sortie

Néanmoins, pour exploiter les multiroutes de longueur égale entre domaines, on peut envisager un processus différent. La condition à satisfaire porte sur les routeurs internes : pour un préfixe donné, ceux-ci sont partitionnés en autant de groupes que de routeurs de bordures ayant redistribué l'annonce dans l'IGP. Chaque partition correspond à une correspondance unique entre préfixe et GWR. En revanche, pour un préfixe donné, les routeurs de bordures peuvent faire le choix du routeur de sortie de domaine. Pour cela, il suffit de s'assurer que les routeurs de cœurs fassent le même choix jusqu'à la sortie du domaine. Pour tirer partie de la diversité inter-AS en bordure, il suffit donc de vérifier que les routeurs internes réalisent une commutation cohérente. Pour cela, un processus de validation à un saut comme DT(1), permet de vérifier que le choix de la correspondance *préfixe*→ GWR est identique entre voisins.

Préfixe IP	Adresse du routeur de bordure
Pr_1	A_1
Pr_2	A_2, A_3

Table de routage pour les préfixes externes.

Destination (GWR)	Prochain saut
A_1	NH_1, NH_2
A_2	NH_3
A_3	NH_4, NH_5

Table de routage pour les destinations internes.

TAB. 2.6 – Correspondance entre *préfixe* et *GWR*

Entre routeurs de bordure il faut s'assurer que le choix par défaut est le même.
Par ailleurs, en cas de conflit (plusieurs sorties équidistantes), il faut utiliser un ordre lexicographique pour départager le choix du GWR de sortie. La figure 2.9 illustre le mécanisme de sélection du routeur de sortie pour un préfixe donné Pr (le coût des liens est unitaire). Sur cet exemple le préfixe Pr est annoncé par deux routeurs de bordures *out* GWR1 et *out* GWR2, chacun proposant un AS Path de même longueur : trois sauts interdomaine. Une fois redistribuée en interne cette double annonce partitionne sans intersection possible les routeurs du domaine en deux. Pour un préfixe donné, il y a donc autant de partitions que de routeurs de sorties resdistribuant ce préfixe en IGP.
Soit le meilleur chemin par défaut est sans ambiguïté, un des routeurs *out*, GWR1 ou GWR2, est le routeur le plus proche au sens de la métrique, soit des chemins de coût égaux existent. Dans ce cas, comme l'indique la figure, lorsqu'un routeur est situé sur *une ligne de partition*, celui-ci sélectionne par défaut le routeur de plus petit identifiant selon un ordre lexicographique propre au domaine. Cela permet de garantir que l'intersection entre partitions obtenues pour un préfixe donné soit vide. Une fois ce choix par défaut strictement déterminé, seuls les routeurs de bordures, par exemple *in* GWR, peuvent disposer d'un choix dans la commutation des paquets provenant d'un autre domaine en utilisant par exemple explicitement l'interface d'entrée pour différencier le trafic. Les routeurs d'entrée du domaine peuvent profiter d'un choix entre plusieurs GWR de sortie et bénéficient aussi de la diversité des chemins générés par le protocole de multiroutage interne sous-jacent si plusieurs routes ont été calculées vers le GWR de sortie choisie. En revanche, une fois le choix du routeur de sortie effectué à l'entrée du paquet, ce choix ne doit pas être remis en question par les routeurs de cœur ni par les routeurs de bordure ultérieurement traversés par le paquet jusqu'à la sortie.
Pour que le routage reste cohérent, les routeurs de bordures doivent s'assurer que leur choix de sortie correspond à celui de leur prédécesseur dans le domaine. Les routeurs de cœur font un choix unique et le trafic ne peut pas être réparti sur plusieurs routeurs de sortie. Ce problème ne se pose qu'entre routeurs de bordure : les routeurs de cœur n'ont pas besoin de vérifier la provenance du trafic car c'est

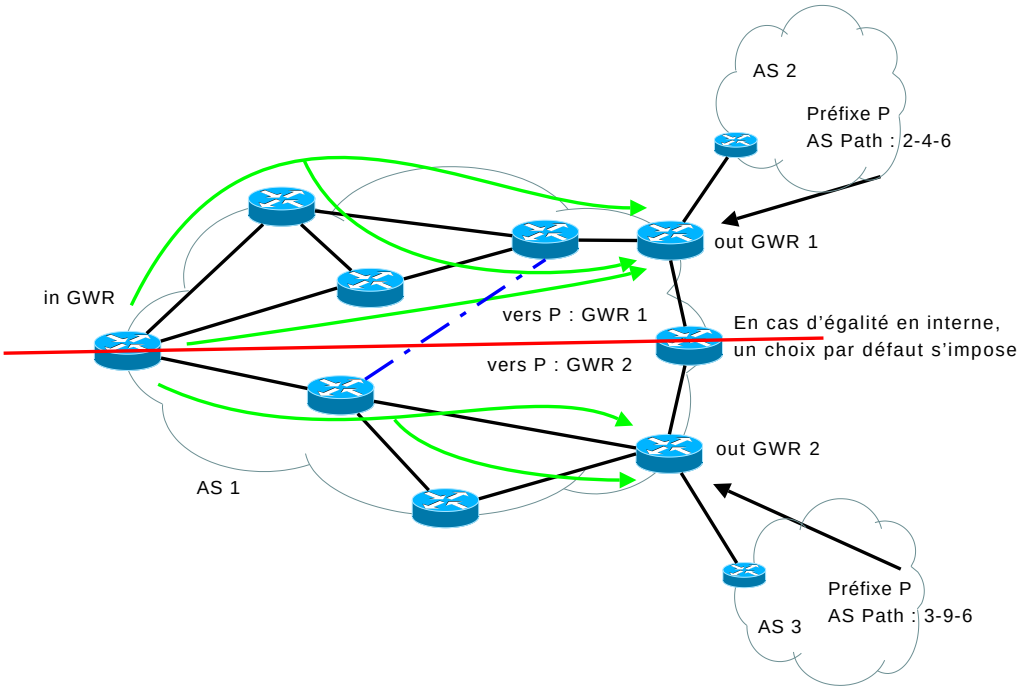


FIG. 2.9 – Partitionnement pour le routage interdomaine

le routeur en amont qui vérifie la cohérence du routage. Les routeurs de bordures disposent du choix de la sortie, mais uniquement à l'entrée du trafic dans le domaine. La règle est la suivante : si le paquet provient d'une interface correspondant à un routeur interne au domaine, alors il doit appliquer son choix par défaut, c'est-à-dire le GWR de sortie par défaut, sinon le choix est piloté par des critères d'ingénierie de trafic.

Tous les routeurs doivent s'assurer que le voisin en aval, choisi pour la commutation vers un préfixe donné, ait fait exactement le même choix du routeur de sortie (celui par défaut entre les GWR). Pour cela, il suffit de s'assurer que le voisin est plus proche, selon la métrique interne, du GWR de sortie. Pour cela, deux solutions sont possibles :

- localement en calculant le SPT de chaque voisin.
- avec un protocole de validation par diffusion comme DT(1).

Nous privilégions la seconde solution si elle est déployée entre routeurs adjacents. La procédure DT(1) est extensible car le nombre de messages dépend seulement des dimensions du domaine et pas du nombre de préfixes. Il suffit que le message $response(d)$ envoyé d'aval en amont soit agrémenté d'un champ contenant le meilleur coût vers la destination d, si celle-ci correspond à un GWR de sortie. Chaque routeur peut ainsi s'assurer que le choix de la commutation via ses voisins en aval soit cohérent. Admettons que le routeur de cœur s ait choisi, pour le routage externe vers le préfixe P, le GWR de sortie dont l'identifiant est d. Pour un préfixe P donné, le critère de cohérence est le suivant : les voisins viables de s sont ceux dont le meilleur coût vers le GWR de sortie d est minimal sur les meilleurs coûts associés à l'ensemble des GWRs de sortie ayant redistribué le préfixe P en interne. Cette vérification

s'effectue localement sur s et le minimum est déterminé avec un ordre lexicographique sur l'adressage IP en cas d'égalité. La complexité du calcul est proportionnel au nombre de préfixes annoncés pour lesquels plusieurs GWR de sortie ont été proposés en interne.
Sur l'exemple 2.9 on observe que *in* GWR peut bénéficier de cinq chemins simultanément vers le préfixe de sortie P. L'arc en pointillé indique que ce lien ne peut être utilisé pour le multiroutage en interne : il connecte deux routeurs appartenant à des partitions différentes.
Le choix de l'affectation du GWR de sortie, réalisée pour un préfixe donné sur les routeurs de bordure, peut s'effectuer sur des échelles de temps indépendantes de celles utilisées pour le partage de charge en interne. Par exemple, les échantillonnages sur de longues périodes de temps seraient associés au trafic interdomaine alors que le *monitoring* en interne pourrait être plus réactif.
Cette proposition permet à un domaine de routage de profiter de la diversité des routes intra et interdomaine. En revanche, la diversité intradomaine peut être restreinte et nécessite des mécanismes de vérification plus complexes que la solution exposée dans le paragraphe 2.3.3.1.

Nos contributions permettent de mettre en œuvre un routage relâché avec une commutation bénéficiant d'une grande variété de prochains sauts dont le choix est dépendant de l'interface entrante des paquets. Contrairement au reroutage monochemin, les critères de validation employés permettent l'utilisation de chemins alternatifs pour dévier partiellement la charge. Les flux peuvent se croiser sans induire la formation de boucles, pour répondre justement aux différentes contraintes définies par les classes de services. La condition IIC permet aux routeurs de s'échanger mutuellement du trafic sous réserve que celui-ci ne soit pas constitué des mêmes flux. De plus, la condition IIC est nettement plus flexible qu'une règle comme LFI sans pour autant autoriser la formation de boucle de routage au niveau routeur comme c'est le cas avec la règle SPD. Le chapitre suivant introduit les notions relatives au problème du partage de charge et à la robustesse d'un réseau IP. Nous y présenterons un état de l'art succinct sur les techniques proposées pour augmenter la fiabilité du réseau en cas de pannes ou bien de congestions. Les contributions que nous développerons dans ce chapitre ont pour objectif d'apporter des éléments de réponse à ces deux problématiques : comment déployer un routage multichemins sans boucles permettant un reroutage rapide et efficace en cas de pannes et de congestions ?
Nos procédures, DT et DT(p), sont adaptées aux deux cas de figure.
Nous étendrons la persective du contournement de congestions pour définir une notion plus globale : l'équilibrage de charge. Cet aspect se définit par l'aptitude d'un réseau à contrôler la répartition des flux sur les routes.

Chapitre 3

Fiabilité et équilibrage de charge

Le routage est une des clés de voûte de la qualité de service. La performance des services proposés sur Internet dépend de la gestion des problèmes physiques et des problèmes induits par une charge mal répartie. Dans ce chapitre, nous étudierons deux objectifs, traditionnellement associés au routage multichemins, et dont le but est d'augmenter la fiabilité et la réactivité du réseau. Nous nous focaliserons sur les problématiques liées au routage intradomaine.
Dans un premier temps, nous traiterons des différents modes de réactions consécutifs aux pannes. Puis nous aborderons le cas d'une déviation de charge partielle en présence de congestion. La protection de liens ou de routeurs peut être vu comme un cas particulier de la déviation de charge avec un routage proportionnel. Les problématiques abordées dans ce chapitre sont liées à la déviation de la charge, totale ou partielle, dans le but de contourner les problèmes, de la congestion transitoire à la panne persistante.
Le premier objectif est de déterminer à quel moment l'utilisation des chemins alternatifs devient nécessaire pour contourner une situation de crise. Le cas extrême, la panne de lien ou de routeur, est traité dans la première partie. La répartition de charge pour l'évitement de congestion, est abordée dans la seconde partie.
Nous décrirons d'abord les principales méthodes de la littérature pour la protection des meilleurs chemins : les protocoles de reroutage rapide. Nous y présenterons notamment les indicateurs de couverture utilisés. La seconde partie introduit un état de l'art des techniques proposées pour le contrôle et la répartition de charge. Pour finir, nous exposerons la solution que nous avons utilisée pour nos simulations.
Dans les deux cas, nous nous focaliserons sur les méthodes de multiroutage au saut par saut. Nous citerons brièvement d'autres solutions, répondant à des problématiques similaires, lorsque la commutation est pilotée par la source.

3.1 Protection et restauration

3.1.1 Généralités

Les techniques de protection et de restauration permettent d'ajouter un niveau supplémentaire de fiabilité, d'intégrité et de disponibilité pour augmenter la robustesse du réseau. La notion de protection se définit généralement comme le positionnement à l'avance d'un chemin de soutien (chemin de secours ou *backup path*). Cela permet une restauration plus rapide que les moyens de reconfiguration traditionnels. En effet, le chemin de secours est déja calculé, mais pas nécessairement activé, avant que la panne ne soit annoncée. Lorsque celle-ci est détectée, le protocole de restauration n'a plus qu'à sélectionner

et activer le chemin alternatif pré-calculé[1].
En revanche, les protocoles de routage IGP, tel qu'OSPF ou IS-IS, sont contraints d'inonder l'ensemble du réseau lorsque la topologie change : les LSA sont diffusés à l'ensemble des routeurs et ceux-ci doivent recalculer le nouveau SPT.
Pour les services et les applications temps-réel tels que la voix ou la télévision sur IP, le temps de réaction est soumis à des contraintes spécifiques de qualité de service. Ces services ne peuvent pas supporter un temps de réaction trop long s'ils veulent garantir une QoS suffisante. La VoIP est, par exemple, particulièrement sensible aux pertes de paquets et aux délais. La vitesse de restauration est cruciale pour ce type de service. Afin de garantir un service non dégradé, un temps de restauration inférieur à 50 millisecondes est souvent pris en référence (BFF05 ; BFF07) pour le routage interdomaine. Avec un tel délai, la couche réseau permet de satisfaire, de manière transparente, les exigences de la couche applicative.
Il existe deux manières simples de réduire le temps nécessaire pour une reconfiguration complète du réseau. La première solution consiste à segmenter le réseau en plusieurs aires si celui est de grande dimension. Cela permet de concentrer l'effort de reconfiguration sur les routeurs appartenant à l'aire concernée. La seconde solution a pour but de minimiser la période déterminant l'émission de deux messages *Hello* consécutifs. Cela permet d'accélérer la phase de détection de la panne.
Un autre domaine de recherche est l'optimisation du temps nécessaire à un nouveau calcul de l'arbre des meilleurs chemins. En effet, il est possible de recalculer seulement la partie de l'arbre concernée par la panne : les branches affectées par la panne. Ces méthodes sont communément désignées sous le terme générique de SPF incrémental (*i-SPF*, voir (MRR78)). Il est également possible d'optimiser le temps de mise à jour de la structure constituant la table de routage : la *FIB*.
Pour obtenir un temps de réaction suffisamment court, il est néanmoins souhaitable que l'ensemble de ces techniques soient combinées à l'utilisation de chemins de secours pré-calculés.
On distingue notamment trois périodes déterminant le temps de convergence avant l'activation de la table de commutation :

1- Détection de la panne : configuration des *timer*, alarmes *SDH* , etc.

2- Notification de la panne : création du message, diffusion de l'états des liens, mise à jour de la base de données topologiques, etc.

3- Reroutage et modification de la commutation : activation du chemin de secours (restauration), mise à jour de la *RIB* et de la *FIB*, etc.

Avant de détailler les méthodes utilisées pour le reroutage rapide sur un domaine IP classique (*Fast Rerouting*, *IPFRR*) [2], nous allons nous intéresser aux caractéristiques des modifications topologiques survenant sur des réseaux réels et les moyens classiques d'accélérer la convergence du protocole de routage utilisé.

[1] Certains protocoles de restauration calculent le chemin de soutien à la volée
[2] A contrario des méthodes existantes sur un domaine MPLS.

3.1.2 Description des pannes et accélération de la convergence

Les analyses de traces issues du réseau Sprint présentées dans (ICBD04) permettent de décrire, par spéculation sur la durée de réparation, différentes classes de problèmes. Par exemple, on observe que seules 10% des pannes durent plus de trois quarts d'heure, alors que plus de 80% des pannes durent moins de 10 minutes. Plus précisement, on distingue quatre catégories d'évènements :

- panne matérielle (10% des cas avec une durée supérieure à 45 minutes),
- panne logicielle (40% avec une durée comprise entre une et 15 minutes),
- maintenance (moins de 5% avec une durée comprise entre 15 et 45 minutes),
- mauvaise interprétation liée par exemple à la surcharge d'un lien (46% avec une durée inférieure à la minute).

Par ailleurs, d'autres résultats issus de Sprint (MIB$^+$04) fournissent une forme de décomposition différente. Seulement 20% des pannes semblent être dues à une intervention planifiée. Parmi les 80% de pannes inattendues, près de 30% concernent des SRLGs alors que le reste affecte uniquement un lien *isolé*.

Bien que les réseaux optiques synchrones (SONET) soient en principe contraints de satisfaire un temps de restauration inférieur à 50 ms, un temps de réaction inférieur à la seconde est considéré comme suffisant. La dégradation des services et applications telles que la VoIP est légère et peu perceptible. Les auteurs de (ICBD04) mettent en avant la dépendance de l'ensemble des mécanismes de restauration avec le jeu de *timers* implémentés à plusieurs niveaux dans les routeurs usuels. Ces timers sont néanmoins garants de la stabilité du routage.

L'article (FFEB05) fournit une description détaillée des différents composants caractérisant le temps total de restauration avec le protocole IGP IS-IS. Les auteurs proposent un ensemble de mesures permettant d'assurer une convergence du protocole de routage intradomaine sous le seuil de la seconde sans compromettre la stabilité de la convergence.

Pour commencer les auteurs estiment que le temps de détection d'une panne est de l'ordre de 20 ms que ce soit avec des mécanismes d'alarmes physiquement intégrés comme sur les réseaux SDH ou SONET, ou avec les messages *Hello* de la couche liaison. Le temps de création du premier message d'avertissement concernant la modification de l'état du lien peut également être très rapide, de l'ordre de 10 ms. Pour cela il est nécessaire de mettre en place des timers dynamiques. La réactivité est alors dépendante de l'état du réseau.

Le temps d'inondation dépend principalement des dimensions du réseau (diamètre, délais de propagations) et ne souffre pas a priori de la mise en tampon dans la mesure où les messages de notification sont prioritaires. Des mécanismes désignés sous le terme générique de *fast flooding* permettent d'accélérer la période de notification notamment en modifiant le *pacing timer*. Leurs tests suggèrent que le temps de diffusion par saut peut donc être relativement insignifiant.

Le temps de calcul du SPT dépend directement du nombre de nœuds appartenant au réseau. Avec des algorithmes incrémentaux, ce temps peut avoisiner une dizaine de millisecondes sur des réseaux composés de plusieurs centaines de nœuds.

Le temps de mise à jour de la RIB/FIB dépend linéairement du nombre de préfixes concernés par

la mise à jour. Les auteurs proposent de considérer en priorité les préfixes les plus importants. Cet ordonnancement est par défaut caractérisé par la largeur du préfixe (annoncé ou non par BGP) mais peut aussi prendre en compte le type de service (VoIP par exemple).
Pour finir, les auteurs analysent également le délai de distribution des calculs effectués sur l'unité centrale vers les cartes réseaux. Le temps entre la mise à jour d'un préfixe par le CPU et sa redistribution vers les cartes peut être accéléré. Pour cela, calcul et distribution doivent être parallelisés et le préfixe à jour doit être transmis en mode *multicast* du CPU vers les différentes cartes.
Les analyses expérimentales présentées dans (SG01) proposent une décomposition similaire avec le protocole OSPF. Leurs mesures proviennent d'équipements *Cisco* et ont été obtenues avec une méthodologie dite *Black-box*[3]. La durée de chaque tâche interne au routeur est analysée et estimée avec des observations externes. En particulier, les auteurs utilisent pour leurs mesures une caractéristique spécifiée par OSPF : un LSA dupliqué est acquitté automatiquement. Cela leur permet de déterminer différents intervalles de temps caractérisant le temps global de convergence : traitement d'un LSA, émission d'un LSA, calcul du SPT et mise à jour de la FIB.

3.1.3 Chemin de secours sur IP : FRR

Dans ce paragraphe nous allons rapidement décrire les méthodes existantes pour un reroutage rapide et local sur des réseaux IP classiques. Il s'agit de protocoles ne nécessitant pas une modification du paradigme classique du routage : la commutation d'un paquet est réalisée en fonction de la destination de celui-ci. Le terme IP-FRR fait référence à un groupe de travail de l'IETF.
Dans le premier chapitre, nous avons déjà introduit les techniques LFA et UTURN. Cependant, ces deux modèles sont dépendants de la topologie et ne permettent pas d'obtenir une couverture complète. Le dernier paragraphe de cette partie fournit une définition précise de la notion de *couverture locale*. Plusieurs contributions ont été proposées pour permettre d'obtenir une couverture locale optimale.
Les tunnels IPFRR décrit dans (SB08) sont dans une certaine mesure une extension de la solution SPF-EE proposée par Wang et Crowcroft. Cependant, les paquets sont encapsulés au lieu d'être commutés en mode *source routing*. En pratique, sur un routeur s, pour la protection d'une panne sur un lien sortant $\{s, v\}$, il faut mettre en place un tunnel lui permettant de contourner ce lien critique. Ce tunnel est un lien virtuel déployé entre le routeur s et un routeur r, appelé *tunnel endpoint*. Le routeur de sortie, r, doit permettre d'atteindre l'ensemble des destinations sans utiliser le lien à protéger.
En général, on considère que ces tunnels sont positionnés lorsque les critères de protection tel que LFA ne suffisent pas. Les paquets déviés sont dirigés vers r en les encapsulant avec une en-tête IP dont le champ destination est r. A la sortie du tunnel, à partir du *tunnel endpoint*, les paquets sont décapsulés et leur commutation suit simplement le meilleur chemin jusqu'à la destination. Pour déterminer les *tunnels endpoint* viables, il faut que s calcule l'intersection entre l'ensemble des routeurs que s peut atteindre sans utiliser le lien $\{s, v\}$ (noté *P-space*) et l'ensemble des routeurs n'utilisant pas le lien $\{s, v\}$ (noté *Q-space*).

[3]Les résultats provenant de (FFEB05) sont issus d'une méthodologie *White-box* dans le sens ou il s'agit de mesures internes.

De plus, les auteurs définissent la notion de *release point* pour forcer la commutation à la sortie du tunnel. Si l'intersection des deux ensembles précédemment cités est vide, il est possible d'utiliser le mode *directed tunnel*. Si r, un *tunnel endpoint*, appartient à l'ensemble *P-space*, et qu'un routeur adjacent à r noté o appartient à l'ensemble *Q-space*, alors les paquets décapsulés à la sortie du tunnel sont dirigés vers o. En d'autres termes, r force leur commutation : il n'utilise pas sa FIB mais force les paquets à passer par o.
Déterminer l'ensemble Q-space requiert le calcul d'un SPT inversé vers la cible : l'extrémité sortante de l'interface à protéger ou le routeur à contourner. Pour un lien $\{s, v\}$, il s'agit de calculer le SPT inversé enraciné sur le nœud v. La complexité de calcul de cette méthode dépend donc du nombre d'interfaces à protéger.
Dans certains cas, il se peut qu'aucun tunnel ne puisse être mis en place pour la protection de certaines interfaces. Plusieurs raisons sont citées dans (SB08), par exemple une forte asymétrie dans la valuation des liens. Néanmoins l'évaluation de protocoles IPFRR réalisée dans (FB05) semble démontrer sur l'ensemble des topologies testées que le mode *directed tunnel* suffit à générer une couverture complète.

Une technique de protection plus récente, intitulée *NotVia address* (BSP06), étend le principe des tunnels pour atteindre une couverture complète. Une adresse *NotVia* est une adresse de secours spécifique attribuée à chaque interface à protéger. La sémantique d'une telle adresse est : *un paquet acheminé à destination d'une adresse NotVia doit être commuté à destination de cette adresse sans être acheminé via le lien auquel cette adresse est associée.*
Pour chaque interface à protéger, deux adresses lui sont attribuées, l'adresse IP classique et l'adresse NotVia. Lorsqu'une panne est détectée, les paquets sont encapsulés à destination de cette adresse de secours. Cette technique requiert la coopération de chaque routeur appartenant au chemin de secours. En effet, chaque routeur associe à l'adresse NotVia un routage spécifique construit sur la base d'un SPT calculé sur le graphe privé de l'interface à protéger. En pratique, chaque routeur est contraint de calculer autant de SPT que d'interfaces à protéger. Par exemple, les adresses NotVia pour la protection de routeurs nécessitent $N-1$ calculs. Ces adresses sont utilisées prioritairement car elles sont plus sûres que celles protégeant les liens, bien que les chemins associés puissent être plus long.
D'une part, étant donnée la complexité engendrée par cette méthode, le groupe de travail IPFRR s'accorde à dire que NotVia doit être utilisé en complément de méthodes plus simples telles que LFA. D'autre part, il est possible de simplifier la complexité de calcul des SPTs grâce à des procédures incrémentales.
Les auteurs de (LFY07) proposent deux idées permettant de réduire la complexité inhérente aux adresses NotVia. Pour commencer, lorsque le lien incriminé n'est pas sur la branche du SPT portant un sous-ensemble de destinations, le meilleur prochain saut et l'adresse NotVia peuvent être confondus. Cela permet d'économiser en complexité pour la FIB. Leur seconde idée permet de réduire le temps nécessaire à la convergence des adresses NotVia prioritaires après un changement topologique. Il s'agit d'ordonnancer la mise à jour des adresses NotVia. Intuitivement, cet ordre peut être déterminé en

fonction de la distance en nombre de sauts entre l'interface à protéger et le routeur effectuant la mise à jour consécutive au changement topologique.
Parallèlement, les auteurs proposent un algorithme, *rNotVia*, de complexité équivalente à celle de *NotVia* pour permettre de connaître le nombre d'adresses NotVia qu'un routeur protège. Cette information peut s'avérer utile pour l'administration du réseau. En effet, leur proposition permet de récolter un ensemble de données utiles pour évaluer l'état du réseau lorsqu'il subit une panne : l'opérateur peut par exemple modifier le comportement du reroutage si une congestion a lieu sur un routeur appartenant au chemin de secours.

Les travaux présentés dans (NLY+07) proposent une technique de protection basée sur une commutation spécifique à l'interface entrante (leur proposition n'est pas liée au groupe de travail IP-FRR). En ce sens, leur contribution est analogue à la nôtre bien que les chemins définis ne soient pas utilisables simultanément. Chaque routeur dispose de deux tables de commutation spécifiques dédiées au routage de secours : une table de routage basée sur l'interface d'entrée et une table pour le reroutage local. Lorsqu'une panne est détectée sur un routeur s, celui-ci ne transmet pas les annonces LSA avant l'expiration d'un timer. Ce timer permet de distinguer une panne transitoire d'une panne longue. Si la panne est résolue avant l'expiration du délai, alors la commutation rebascule sur un mode de commutation classique via le meilleur chemin. Pendant une panne, le reroutage peut être sous-optimal mais il est nécessairement sans boucle au niveau lien durant la reconfiguration complète du réseau. Lorsqu'un routeur détecte une panne sur l'un de ses liens sortants, les paquets sont acheminés en fonction de la table de secours. La table de routage par interface entrante est utilisée pour le trafic en transit lorsque la panne n'est pas locale ou lorsque la panne n'affecte pas la ligne de routage concernée.
Plus formellement, les auteurs définissent la notion de *keylinks* pour désigner l'ensemble des liens pouvant provoquer un basculement vers un routage de secours. Soit un routeur s voisin d'un routeur v, en considérant une destination d : un lien $\{o, l\}$ est contenu dans l'ensemble des keylinks noté $K^d_{v\rightarrow s}$ pour le triplet (d, v, s) si $v = NH_k(s, d)|C_k(s, d) = C_1(s, d)$ et si sur le graphe G réduit de $\{o, l\}$, le lien $\{v, s\}$ est inclus dans $P_1(o, d)$. Leurs algorithmes fonctionnent aussi avec ECMP. Les auteurs font l'hypothèse que la valuation des liens est symétrique.
Une fois l'ensemble des keylinks déterminé pour toutes les destinations possibles, un routeur peut calculer sa table de routage par interface d'entrée et sa table de reroutage local. Pour une destination donnée, une ligne de routage sur un routeur s pour le trafic en transit provenant de l'interface v est calculé à partir du SPT obtenu sur le graphe réduit de l'ensemble des arcs $e \in K^d_{v\rightarrow s}$. Pour obtenir une ligne de routage sur s vers d lorsque le lien $\{s, v\}$ subit une panne, la table de secours pour le reroutage local est calculée à partir du SPT obtenu sur le graphe réduit de l'ensemble des arcs $e \in K^d_{s\rightarrow v} \bigcup \{s, v\}$. En effet, si le lien $\{s, v\}$ est en panne alors que s utilisait v comme NH cela signifie que les liens appartenant à $K^d_{s\rightarrow v}$ sont aussi en panne.
Lorsqu'une panne est détectée localement sur un lien $\{s, v\}$, alors les paquets provenant d'une interface r à destination de d sont commutés selon :

– la ligne de routage $(r, n, d)_s$ appartenant à la table de routage pour le trafic en transit si $n \neq v$.

– la ligne appartenant à la table de routage de secours sinon.

Il est possible de réduire la complexité des algorithmes de recherche de *keylinks* et de construction des tables associées. Lors de la réception d'une notification de modification topologique, il suffit d'utiliser des mécanismes incrémentaux pour conserver les étapes intermédaires obtenues lors des calculs précédents.

3.1.4 Boucles transitoires

Après une modification topologique, la convergence du protocole de routage peut provoquer la formation de boucles de routage transitoires. Ces boucles transitoires peuvent survenir pour deux raisons : une panne inatendue ou une opération de maintenance planifiée. Dans les deux cas, il est possible d'éviter de perdre des paquets ou de créer des boucles de routage. Lorsque la modification topologique n'est pas planifiée, cela suppose que le réseau dispose d'un protocole de reroutage rapide pour contourner la panne durant la période de reconfiguration.

Dans la partie 1.2.3.3, nous avons cité une condition permettant d'éviter la formation de boucles à tout moment, néanmoins celle-ci utilise une règle topologique à vérifier *par destination*. La commutation des paquets est réalisée sans mettre automatiquement à jour la FIB, et les auteurs ne précisent pas comment s'opère le passage à l'échelle vers des milliers de préfixes IP.

Les auteurs de (FB07) proposent une solution d'ordonnancement efficace. Dans le cas d'une opération planifiée, il suffit de déterminer l'ordre dans lequel les routeurs doivent mettre à jour leur FIB. Pour les pannes, par définition non planifiées, il est nécessaire que le routage sous-jacent dispose d'un protocole de reroutage rapide fiable pour supporter la durée de planification de mise à jour de la FIB de chaque routeur. En tâche de fond, la reconfiguration complète est organisée pendant que les flux sont redirigés vers les issues de secours précalculées avec le protocole IPFRR choisi. Pour planifier la mise à jour des FIBs, il est nécéssaire de définir un ordonnancement sans ambiguïté. Cette ordre dépend de la nature de la modification topologigue.

Si la panne a lieu sur un lien $\{s, v\}$, ou si la valuation de cet arc augmente, chaque routeur r doit calculer un SPT inversé vers v. Ce SPT est construit sur la base de la topologie avant modification et ne concerne que les meilleurs chemins affectés par la panne. Chaque routeur r peut alors déterminer son rang : il s'agit de la distance maximale en nombre de sauts d'un nœud ancêtre sur le SPT inversé vers r. La distance maximale intervient pour les chemins de coût égaux avec l'extension ECMP. Un routeur r peut alors mettre sa FIB à jour seulement après que chaque routeur ancêtre, et donc de rang inférieur, l'ait fait aussi. Pour cela il faut définir un temps maximal, $TMFIB$, de mise à jour de la FIB selon le nombre de préfixes annoncés dans le domaine.

S'il s'agit de l'arrivée d'un nouveau lien $\{s, v\}$, ou que la valuation de l'arc $\{s, v\}$ décroit, chaque routeur r doit calculer le nouveau SPT du graphe. Ainsi, il peut déterminer son rang en fonction du nombre de sauts maximal le séparant de v. De même que précédemment, un routeur r met à jour sa FIB dès lors que son *timer* a expiré après avoir été notifié de la modification. La durée de ce timer est le produit de son rang et de $TMFIB$.

Les résultats présentés analysent également l'ordonnancement de la mise à jour lorsque les evènements sont liés à la modification d'un lien appartenant à un SRLG ou lorsqu'il s'agit du retrait ou de l'ap-

parition d'un routeur. Afin d'accélérer le temps de convergence, qui dépend directement du paramètre $TMFIB$, les auteurs introduisent plusieurs mécanismes : des messages pour signifier la fin de la mise à jour et une technique pour tronquer le rang lorsque certains nouveaux NH ne sont pas affectés par la panne.
Ces techniques ne peuvent pas s'appliquer à un protocole de multiroutage saut par saut tel que DT(p). En effet, bien que les distances - en nombre de sauts - sur le graphe des chemins multiples (DT) soient connues, les routes ne le sont pas. Un chemin et le prochain saut associé représentent un coût faisable et peuvent correspondre à de multiples chemins activés de proche en proche : les routes.

3.1.5 Approches alternatives

Cette partie présente succinctement les plans de protection proposant des approches différentes des techniques IPFRR. Dans un premier paragraphe, nous citerons quelques propositions pouvant être mises en place sur un domaine MPLS. Le second paragraphe introduira une autre catégorie de solutions où il s'agit de modifier le poids accordé à chaque lien du domaine, pour permettre au protocole IGP de prendre en compte la notion de protection.

3.1.5.1 Protection par liaison logique

L'architecture MPLS facilite la mise en place de liaisons de secours en cas de panne sur le LSP primaire. Le RFC 3469 (SH03) fournit un ensemble d'opérations requises pour la protection sur un domaine MPLS. Typiquement, il s'agit de déployer un ensemble de tunnels aux propriétés équivalentes entre plusieurs couples de routeurs ingress/egress. La complexité induite par ce type de méthode dépend du nombre de segments protégés.
Les travaux présentés par Kodialam et al. dans (KLS04) mettent en avant le problème de la complexité des procédures engagées sur le routeur d'entrée du domaine. Les auteurs proposent un algorithme d'approximation pour réduire la complexité de calcul du ingress LSR à un facteur polynomial.
Par ailleurs, d'autres travaux se focalisent sur la protection de LSP interdomaine. Nous avons déjà cité dans le paragraphe 1.4.2.2 les travaux de Pelsser et Bonaventure : il s'agit d'un ensemble d'extensions permettant à MPLS d'être généralisé sur plusieurs AS sans compromettre la confidentialité topologique de chacun des AS.
Les travaux de Farrel et Vasseur présentés dans le RFC 4726 (FVA06), bien qu'orientés ingénierie de trafic en interdomaine, introduisent un certain nombre d'outils de mesure utiles pour la protection. Ils y présentent notamment la notion de serveur dédié et placé à l'entrée de chaque système autonome.
Les auteurs de (WYL$^+$07) proposent une approche originale : la protection des liaisons appartenant à un domaine donné est assuré par le routage interdomaine. Cela implique que les AS se couvrent entre eux pour accroître la redondance des liaisons à faible coût. Leur protocole *reliability as an interdomain service* (REIN) est basé sur quatre axes :

- Mise en place d'un modèle économique entre les AS.
- Signalisation via des annonces BGP spécifiques pour partager des chemins de secours interdomaine.

– Mise en œuvre de LSP interdomaine permettant aux données de sortir puis d'entrer à nouveau dans le même domaine sans risque de boucles de routage.
– Augmentation de la fiabilité en sélectionnant les chemins interdomaine les plus adaptés.

Ce dernier point est décomposé en trois étapes. Il s'agit, d'une part, de résoudre un problème d'optimisation basé sur la formulation énoncée dans (WXQ+06) prenant en compte les différents niveaux de services et en vérifiant les contraintes correspondant aux pires cas possibles. D'autre part, il s'agit de transformer les proportions obtenues sous forme de fraction de flot à chaque saut, en routes exploitables avec MPLS. Sur domaine MPLS, a priori seul l'ingress LSR détermine les proportions de trafic à acheminer sur chacun des chemins vers l'egress LSR. Pour finir, les auteurs proposent une technique de pré-sélection des chemins interdomaine. Cette méthode détermine d'abord un sous-ensemble de chemins de secours au coût minimal, puis cet ensemble est augmenté séquentiellement jusqu'à atteindre l'objectif de protection souhaité.

3.1.5.2 Métrique et valuation

De manière générale, pour optimiser le routage IGP lorsque la matrice des demandes est connue ou estimable par des techniques de métrologie, une approche connue consiste à modifier la valuation de chacun des liens du réseau. Dans le contexte du routage saut par saut, si les demandes entre chaque paire de routeurs sont suffisamment stables, il est possible d'accorder un poids spécifique à chaque arc afin que les délais d'acheminement des demandes soient minimisés. Cette minimisation est réalisée en fonction de prédictions obtenues par des mesures de trafic antérieur.
Fortz et Thorup (FT02) ont démontré que ce problème d'optimisation est NP-complet. Néanmoins, avec la mise en œuvre d'heuristiques d'attribution de poids, il est possible d'obtenir des performances comparables à celles que les réseaux d'overlay peuvent en théorie apportées (en utilisant autant de routes que nécessaire). La référence (WWZ01) démontre justement que les protocoles de routage IGP classiques utilisant le jeu de poids approprié, avec un acheminenemt uniquement réalisé sur les meilleurs chemins, peuvent répondre au même problème d'optimisation que lorsque la commutation est multichemins avec MPLS.
Par rapport à la métrique traditionnelle qui consiste à valuer un arc en fonction de l'inverse de sa capacité ou proportionnellement à son délai de propagation, Fortz et Thorup ont montré par simulation que leur algorithme de recherche locale semble supporter un excès de demande de l'ordre de 50 à 110% sans que les capacités des liens ne soient saturés. L'heuristique proposée, *HeurOSPF*, utilise deux tables avec fonctions de hachage pour d'une part guider la recherche en évitant les visites cycliques de minimum locaux et d'autre part forcer la diversification de la recherche de solutions. Comparé aux techniques de routage dans le pire des cas[4] (*oblivious routing*), *HeurOSPF* permet d'obtenir des résultats nettement plus proche de la solution optimale.
Les travaux présentés par Nucci et al. (NBTD07) proposent une approche similaire dont la spécificité est de prendre en compte les pannes de liens. Leur méthode permet de différencier les niveaux de service

[4]Il s'agit d'une vision pessimiste où l'analyse considère le pire des cas en termes de performance par rapport à l'ensemble des demandes. Ces aspects seront développés dans la seconde partie.

(*Service level agreement*, SLA). Pour équilibrer dynamiquement la charge induite après l'occurence d'une panne, l'approche dynamique proposée par Fortz et Thorup (FT00) consiste à réassigner le poids d'un petit nombre de liens. Cependant Nucci et al. ont constaté qu'une telle convergence est relativement lente par rapport à la nature transitoire de la plupart des pannes. Leur but est donc de minimiser l'impact des pannes transitoires sur le trafic. Pour cela, les auteurs intègrent dans la formulation du problème d'assignement optimal à la fois les limites liées aux SLA et l'impact de la panne de chacun des liens. Leur fonction objective prend en compte l'ensemble de ces éléments pour former une métrique composite configurable pour établir un compromis entre les deux problèmes. L'algorithme proposé est une heuristique de recherche *Tabu* basée sur la méthodologie présentée dans (GL93). Le principe fondamental est l'enregistrement des solutions visitées antérieurement dans une liste *Tabu*. Cette liste permet de vérifier l'absence de cycles de recherche. Le pas itératif peut être accéléré par une réattribution simultanée du poids de plusieurs liens pour permettre une diversification de la recherche.

3.1.6 Couverture

Cette partie introduit les notions de couverture locale et globale pour le multiroutage saut par saut. Les notations utilisées ici sont généralisées pour prendre en compte l'activation des prochains sauts en fonction de l'interface d'entrée. Avec le processus de validation introduit dans le chapitre précédent, les ensembles de prochains sauts activés pour les routeurs en amont sont inclus dans l'ensemble des NHs activés pour le trafic local. D'un routeur en amont à l'autre, les ensembles de NH activés sur un voisin s vers une destination d donnée sont différents.
Nous avons envisagé la notion de protection globale pour mesurer la proportion de routeur pouvant détourner les paquets issus de leur trafic local alors qu'ils sont incapables de dévier le trafic en transit pour certaines interfaces entrantes. Un routeur peut dans ce cas avertir les nœuds en amont pour lesquels il ne dispose d'aucun NH alternatif activé pour les destinations concernées. Le paragraphe 3.1.7 fournit des éléments de réponse pour mettre en place une telle notification. Le but de ces mécanismes n'est pas de se substituer aux annonces LSA mais d'agir en parallèle pour accélérer le reroutage. Dans cette partie et dans les résultats présentés dans le chapitre suivant, nous avons seulement évalué la protection de lien.
Afin de formellement introduire les mesures effectuées, nous définissons ici la métrique utilisée pour mesurer la couverture. La qualité de couverture est caractérisée par une valeur comprise dans l'intervalle $[0, 1]$, 1 désigne une couverture complète et 0 désigne l'absence de couverture. Ces notions sont relatives aux routes activées par le processus de validation choisi.
La couverture d'un lien $\{r_{i-1}, r_i\}$ appartenant à une route $\mathfrak{R} = (r_1, r_2, \cdots, r_i, r_{i+1} \cdots, r_m) \in R_m(s, d)$ peut se définir de deux manières. La notation $R(s, d)$ désigne l'ensemble des routes activées de proche en proche entre s et d quelle que soit leurs longueurs en nombre de sauts. Les fonctions $\gamma(i)$ et $\Phi(i)$ représentent respectivement la protection locale et globale d'un lien $\{r_i, r_{i+1}\}$[5]. La protection locale

[5]Pour rappel, $r_0 = s$ et $r_m = d$

d'un lien se définit ainsi :

$$\gamma(i) = \begin{cases} 1, \text{ si } |NH(r_{i-1}, r_i, d)| > 1 \ _{1 \leq i < m} \text{ ou si } |NH(s, s, d)| > 1 \text{ pour } i = 0 \\ 0, \text{sinon.} \end{cases}$$

La protection globale d'un lien est caractérisée par ($\Phi(0) = \gamma(0)$) :

$$\Phi(i) = \begin{cases} 1, \text{ si } \exists (r'_1, ..., r'_j, ..., r'_k) \in R(s, d) | r'_j = r_i \Rightarrow r'_{j+1} \neq r_{i+1} \ _{1 \leq i < m, 1 \leq j < k} \\ 0, \text{sinon.} \end{cases}$$

Soit une route $\mathfrak{R} = (r_1, r_2, \cdots, r_i, r_{i+1} \cdots, r_m) \in R(s, d)$, la couverture **locale** de cette route $T_{loc}(\mathfrak{R})$ est égale à $\frac{\sum_{i=0}^{m} \gamma(i)}{m}$ alors que la couverture **globale** de cette même route $T_{glb}(\mathfrak{R})$ se définit par $\frac{\sum_{i=1}^{m} \Phi(i)}{m}$.
Si la couverture globale $T_{glb}(\mathfrak{R})$ d'une route n'est pas complète ($T_{glb} < 1$), cela signifie qu'il existe au moins un lien sur la route ne pouvant être protégé, ni localement ni au moyen d'une notification en amont signalant le perte de connectivité vers la destination. En d'autres termes, toutes les routes calculées entre s et d utilisent ce lien.
Respectivement, la couverture locale T_{loc} et la couverture globale T_{glb} de l'ensemble des routes activées sur le réseau s'expriment alors ainsi :

$$T_{loc} = \sum_{\forall s,d \in N} \frac{\sum_{\forall \mathfrak{R} \in R(s,d)} T_{loc}(\mathfrak{R})}{|R(s, d)|} \tag{3.1}$$

$$T_{glb} = \sum_{\forall s,d \in N} \frac{\sum_{\forall \mathfrak{R} \in R(s,d)} T(\mathfrak{R})}{|R(s, d)|} \tag{3.2}$$

Ces deux définitions mesurent un niveau de protection sur toutes les routes activées via le processus de validation choisie pour le multiroutage. Les techniques IPFRR ne considèrent que la protection des meilleurs chemins. Pour une comparaison *relativement* équitable du niveau de couverture, seules les routes correspondant à des meilleurs chemins de coût égaux doivent être analysées. Chaque route optimale $(r_1, ..., r_i, r_{i+1}, ...)$ est caractérisée par la relation récursive : $r_{i+1} \in \{NH_j(r_i, d)\}$ seulement si $C_j(r_i, d) = C_1(r_i, d)$. En effet, la notion de couverture n'a pas le même sens lorsque le routage est multichemins, les routes alternatives peuvent être utilisées pour d'autres considérations que le contournement de panne. Le niveau de couverture peut alors se définir comme le taux de protection de l'ensemble des routes primaires et alternatives.
La figure 3.1 permet d'illustrer ces différentes notions. Admettons que le routeur s ait activé trois NHs pour son trafic local vers d : $r_1, v_0, v_1 \in NH(s, s, d)$. Si s désire protéger le lien $\{s, r_1\}$, il dispose donc de deux alternatives locales via v_0 et v_1. Si on considère qu'il existe sur s une route via $(r_1, r_2, r_3, ..., d)$ de m sauts pour atteindre d, alors $\gamma(0) = 1$ et cette route est protégée *au moins* à un niveau de $\frac{1}{m}$.
Admettons que pour le trafic provenant de p_0, seul le NH via r_1 soit activé sur s, c'est-à-dire qu'il

existe une seule ligne de routage $(p_0, r_1, d)_s$ activée pour l'interface d'entrée p_0. Sur la route $\mathfrak{R} = (s, r_1, r_2, r_3, ..., d) \in R_{m+1}(p_0, d) =$, $\gamma(1) = 0$, et la protection locale du lien $\{s, r_1\}$ est nulle pour le trafic provenant de p_0. Néanmoins, la protection globale du lien $\{s, r_1\}$ pour la route $\mathfrak{R}$ est non nulle car il existe un chemin appartenant à $R(p_0, d)$ ne passant pas par ce lien : $(v_0, r_3, ..., d) \in R_{m-2}(p_0, d)$. Il suffit que s notifie p_0 pour lui signaler que la destination d n'est plus accessible via le lien $\{p_0, s\}$. Par ailleurs, pour la route $(r_1, r_2, r_3, ..., d) \in R_m(s, d)$, on remarque que le lien $\{s, v_0\}$ permet de protéger les trois premiers liens de cette route, alors que le NH de s via v_1 n'en protège que les deux premiers liens. Cette observation illustre l'intérêt d'utiliser une métrique pénalisant la valuation des arcs en commun sur les multichemins calculés avec DT. Par exemple en favorisant la pré-sélection, via notre algorithme de calcul, des chemins transverses simples au détriment des chemins transverses avant, il est possible de valider davantage de chemins disjoints.
Considérons le routeur p_1 en amont de s, si l'ensemble $NH(p_1, s, d)$ vérifie $|NH(p_1, s, d)| = 2$ avec, par exemple, $NH(p_1, s, d) = \{v_0 = NH_1(s, d), r_1 = NH_2(s, d)\}$ alors les deux routes de p_1 via s garantissent réciproquement la protection des liens $\{s, r_1\}$,$\{r_1, r_2\}$,$\{r_2, r_3\}$ et $\{s, v_0\}$, $\{v_0, r_3\}$. Considérons le cas moins favorable où p_2 et v_1 sont confondus ($p_2 = v_1$) et tel que $NH(p_2, s, d) = \{NH_1(s, d) = v_0\}$. Dans ce cas, le reroutage local sur s des paquets provenant de p_2 est impossible. Néanmoins, avec une notification adaptée il est possible de protéger l'ensemble des liens entre p_2 et r_3 utilisés par les deux routes de l'ensemble $R(p_2, d)$. Seuls les paquets émis par p_2 vers d via s émis durant le laps de temps entre le moment où la panne survient et le moment où $_2$ désactive sa ligne de routage $(p_2, s, d)_{p_2}$ sont perdus.

Par nature, la protection mise en place avec un multiroutage au saut par saut est uniquement locale. Si une alternative au prochain saut primaire existe précisément à l'endroit où la panne est détectée, il suffit de s'assurer que le chemin alternatif ne contient pas le, ou les éléments faisant défaut. En pratique, selon le type de panne détectée (liens, routeurs ou SRLG), déterminer la capacité de reroutage du chemin alternatif n'est pas trivial localement. Dans la mesure où un chemin calculé ne correspond pas nécessairement à la route empruntée par les paquets, le reroutage peut être un problème complexe : seul la commutation sur le premier saut est contrôlable, la panne de lien est un type de

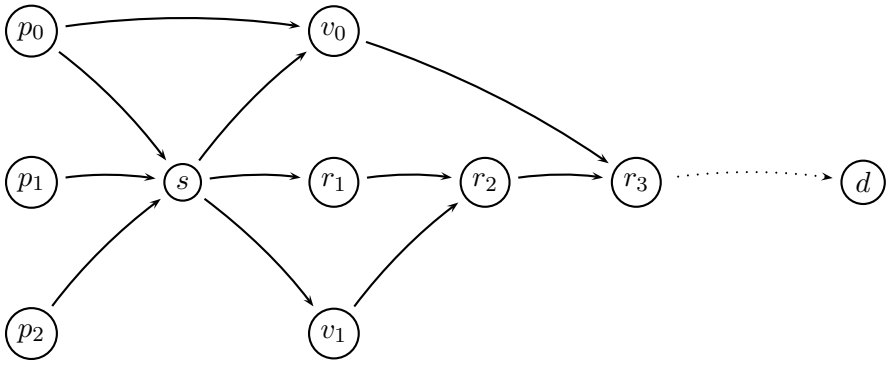


FIG. 3.1 – Couverture multichemins

panne facilement contournable localement.
Par ailleurs, bien que le routage multichemins semble particulièrement bien adapté aux problèmes de protection et de restauration, il est nécessaire d'insister sur la difficulté de mettre en place un niveau de couverture élevé lorsque plusieurs chemins aux coût inégaux peuvent être utilisés simultanément. Les protocoles de restauration rapide ne sont pas soumis aux mêmes hypothèses, et leurs règles de validation bénéficient de cet avantage.
Pour pallier cette difficulté, nous avons envisagé l'utilisation d'un protocole de notification distribué plus léger que la diffusion de LSA.

3.1.7 Modèle de notification pour une protection globale

Le routage multichemins permet de bénéficier d'une solution efficace et temporaire de reroutage rapide pendant que le changement topologique est diffusé via les LSA et que les routes optimales sont recalculées. Cependant il est possible qu'il n'y ait aucune solution locale pour rediriger le trafic. Par ailleurs, il se peut que la redirection proposée ne supporte pas l'excès de charge induit par la déviation de trafic consécutive à la panne. Une possibilité intuitive est d'alerter l'ensemble des routeurs en amont concernés par la panne. Soit s un routeur détectant une panne sur un lien $\{s, v\}$ concernant un sous ensemble de destinations $D = \{d_1, d_2, ..., d_i, ..., d_n\}$ qu'il ne peut rerouter localement. Soit s ne dispose pas de chemin alternatif vers l'une des destinations d_i, soit le prochain saut alternatif de s ne supporte pas la charge de reroutage induite. Ce second cas est généralisé dans la partie suivante dans le cadre de l'équilibrage de charge. Pour simplifier, nous ne considèrerons pas le cas d'une panne concernant les liens groupés de type SRLG. Dans la suite du paragraphe, la figure 3.2 est utilisée pour décrire le mécanisme de remontée d'informations en amont.
Dès lors que l'absence d'alternative est constatée, le routeur s en informe ses routeurs en amont (u et p sur l'exemple). Soit $d = d_i$, une des destinations concernées ; s doit prévenir ses routeurs adjacents s'ils sont présents dans sa FIB tel que, p est un routeur en amont si : $\exists p \mid NH(p, s, d) \neq \emptyset$. Par définition $\{p, s\}$ est un lien entrant vers la destination d. Le routeur détecteur s notifie l'ensemble de ses routeurs en amont dans le but que ceux-ci arrêtent de lui envoyer du trafic à destination de d.
Pour les routeurs en amont (par exemple p sur la figure), la notification est récursive. Si la FIB de p contient un routeur d'entrée p' tel que $s \in NH(p', p, d)$, et si p n'est pas capable de localement rediriger les paquets provenant de p' via un NH alternatif ($\mid NH(p', p, d) \mid \geq 2$), il transmet alors la notification à p'.
Initialement, le premier message de notification émis par le routeur détectant la panne contient la liste des destinations affectées. Chaque routeur en amont reçoit une liste de destinations en fonction des lignes de routage qu'il a activées sur le routeur émettant le message de notification. A la réception d'un tel message depuis un routeur s, un routeur p, pour chaque destination d_i contenue dans la liste émise, désactive les lignes de routage correspondant au lien par lequel le message a été reçu.
Dans la FIB, toutes les lignes de routage de la forme $(p', s, d_i)_p$ sont concernées. S'il n'existe plus aucun prochain saut actif pour une destination d et une interface d'entrée p', alors p ajoute d dans la liste des destinations affectées dans le message qu'il enverra à p'. Lorsque l'ensemble des destinations de la

liste reçue depuis s ont été testées, p envoie à p' un message de notification contenant l'ensemble des destinations pour lesquelles la protection locale est non satisfaite pour le trafic en transit de p'. Sur la figure 3.2, admettons que s détecte une coupure sur le lien $\{s, v\}$ affectant la destination d. Considérons dans un premier temps que le prochain saut de secours $\{s, n\}$ n'existe pas. Le routeur s informe alors l'ensemble de ses prédécesseurs pour lesquels il existe une ligne de routage vers d utilisant le prochain saut v. Sur l'exemple, il s'agit des routeurs p et u. Le routeur u ne dispose d'aucune possibilité locale de reroutage vers d pour le trafic en transit provenant de p, il transmet alors directement l'alerte à celui-ci et désactive la ligne de routage (p, s, d). En revanche, u dispose d'un prochain saut alternatif via p pour son trafic local, il se contente alors de désactiver l'entrée $(u, s, d)_u$ dans sa FIB. Le routeur p reçoit alors l'annonce du routeur s ainsi que l'annonce transmise par u. Il désactive donc les lignes de routage $(p', s, d)_p$, $(p, s, d)_p$ et $(p', u, d)_p$, $(p, u, d)_p$. Dès lors, p utilise pour son trafic local et le trafic de transit uniquement le chemin de secours représenté sur la figure.

Considérons à présent le cas plus complexe d'une panne de routeur, admettons que le routeur v soit hors service. Dans ce cas, on observe que le chemin alternatif de s via n est inutile pour la destination d. Cependant le principe général reste le même que pour la panne de lien. En effet, le routeur n émettra une annonce vers s pour l'avertir qu'il est incapable de rediriger son trafic. Ainsi, s désactive les deux entrées de sa FIB vers d et retransmet l'alerte vers p. S'il existe une route alternative ne passant pas par v, le routeur p ne recevra pas d'alerte via le premier lien de celui-ci.

Une fois qu'un routeur a évalué toutes les lignes de routage pour les destinations internes concernées par la panne, la translation vers les préfixes externes au domaine doit tenir compte des entrées désactivées. Pour cela, il faut que la FIB finale contenant les destinations associées aux préfixes externes considère le même ensemble de NHs que la FIB dédiée aux destinations internes. En effet, si l'ensemble des NHs actifs vers un routeur de bordure est différent de l'ensemble des NHs actifs vers les préfixes annoncés par ce routeur, alors la notification est incohérente pour les adresses externes.

En pratique, pour que le mécanisme décrit soit utile, le temps de notification global des routeurs en amont doit être inférieur à la période de reconfiguration des meilleures routes avec les LSA. En effet, il est inutile de transmettre l'annonce de la panne à un routeur en amont éloigné si le temps nécessaire pour l'alerter est équivalent au temps de convergence d'un protocole de routage classique (tel que OSPF ou IS-IS) lorsque la topologie évolue. Les annonces correspondant à la notification d'une panne ne sont plus retransmises lorsque tous les routeurs notifiés sont capables d'utiliser une route contournant la panne.

On peut aussi profiter d'un tel mécanisme de reroutage pour retarder les LSA si la panne est temporaire. Pour cela, il faut utiliser un *timer* pour distinguer les pannes transitoires des pannes persistantes. Dans ce cas, le protocole de notification permet d'économiser le temps de calcul du *SPT* et la mise à jour de la *FIB*. Pour la FIB, il s'agit seulement de désactiver certains NHs : la proportion de trafic qui leur est associée est réduite à zéro. Si le timer a expiré, ces opérations sont réalisées par la suite. De cette manière, le temps de restauration dépend seulement de la période nécessaire à la détection de la panne et du délai de notification. Le temps de restauration est donc proportionnel à la distance entre la panne et les points de reroutage. Ces distances sont entièrement dépendantes de la topologie et de la diversité

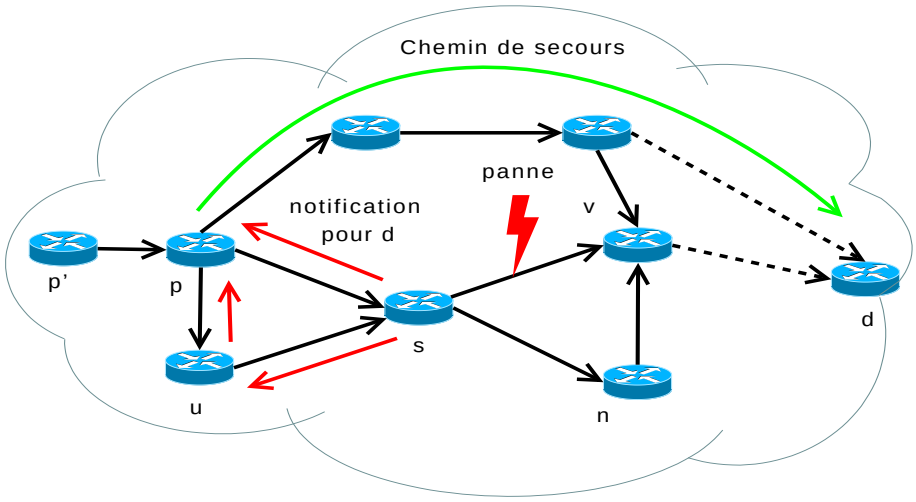


FIG. 3.2 – Vague de notification en amont

des chemins générés par le protocole de routage multichemins sous-jacent.

3.2 Equilibrage de charge

Le partage de charge sur des routes explicites avec une commutation MPLS est un domaine de recherche prolifique. Les solutions proposées dans la littérature utilisent généralement une formulation adaptée à la résolution d'un problème d'optimisation. Selon la variabilité du trafic et les objectifs d'optimisation (cas moyen, pire des cas), le partage de charge peut se faire à partir de connaissances sur le trafic a priori pour dimensionner le réseau. Les mesures utilisées peuvent être réalisées en temps-réel ou à l'avance (*online*/*offline*). Lorsque les mesures sont obtenues *online*, la répartition est réactive, alors que des mesures *offline* sont par nature associées aux modèles proactifs permettant un dimensionnement par anticipation.
Le dimensionnement du réseau, au sens positionnement des routes, peut prendre deux formes : soit le pire des cas est considéré (*oblivious routing*) pour anticiper des modifications de trafic importantes, soit le routage est adapté au cas moyen, c'est-à-dire lorsque le trafic est relativement stable. Les critères d'optimisation peuvent prendre diverses formes selon les objectifs que nous expliciterons par la suite.
Pour éviter les problèmes liés au caractère distribué des protocoles de routage au saut par saut, les boucles de routage par exemple, le contexte technologique le plus adéquat est la commutation d'étiquettes sur un domaine MPLS. L'utilisation d'une signalisation explicite des routes avec les protocoles de signalisation CR-LDP ou RSVP-TE permet d'établir un partage de charge sur le routeur d'entrée du domaine. L'équilibrage de charge dans un contexte distribué et réactif est un problème nettement plus ouvert où les solutions présentées sont généralement incrémentales.
Pour commencer, nous étudierons succinctement les formulations généralement utilisées pour modéliser le problème du partage de charge. Nous détaillerons ensuite les solutions proposées dans la littérature dans un cadre distribué. Pour finir, nous présenterons notre proposition réactive et distribuée.

3.2.1 Contrôle et répartition de la charge

Le partage de charge piloté à la source sur des chemins activés de bout en bout peut s'exprimer selon plusieurs points de vue. Avant de présenter les problèmes d'optimisation classiques pour le routage proportionnel, nous allons rapidement présenter le problème de contrôle du débit.

3.2.1.1 Contrôle du débit

Pour formuler le problème du contrôle de débit par session sur de multiples routes, nous pouvons utiliser la formulation de Kelly et al. (KMT98). Soit $Z = R(s, d)$ l'ensemble de routes reliant une source s et une destination d. Le débit x_Z représente alors la somme des débits $x_{\mathfrak{R}}$ sur chacune des routes $\mathfrak{R}$ appartenant à Z.
Le terme $Z(l)$ désigne l'ensemble des routes de Z utilisant le lien l. Le terme L_Z désigne l'ensemble des liens utilisés par les routes appartenant à Z. La fonction V représente l'objectif de maximisation, il s'agit de la fonction d'utilité qu'il faut appliquer à l'ensemble des routes entre s et d : $V_Z(x_Z)$. La fonction V doit être une fonction continue strictement concave pour $x_Z \geq 0$.
Afin de maximiser l'ensemble des débits $x_{\mathfrak{R}}$ sur le réseau, il faut résoudre le problème d'optimisation suivant :

$$\max_{x_Z \geq 0} \sum_{\forall Z} V_Z(\sum_{\forall \mathfrak{R} \in Z} x_{\mathfrak{R}})$$

sujet à la contrainte

$$\forall l \in E \sum_{\mathfrak{R} \in Z(l)} x_{\mathfrak{R}} \leq c(l)$$

Bien que ce problème puisse être résolu de manière centralisée, cela implique que la fonction V soit connue du réseau. Pour contourner cette hypothèse, Kelly et al. proposent de décomposer le problème en deux sous problèmes. Côté utilisateur (en pratique assimilable à un couple source/destination), la fonction d'utilité est connue, et côté réseau celle-ci n'intervient plus dans la formulation globale du problème d'optimisation. Chaque utilisateur reçoit un débit proportionnel à la somme qu'il est prêt à payer. Les auteurs présentent deux classes d'algorithmes décentralisées permettant de résoudre le problème global et son dual via un protocole de signalisation global.
Les travaux de Roberts et al. présentés dans (OR07) sont également basés sur un contrôle du débit par source. Leur analyse est en partie fondée sur la notion de probabilité de blocage pour un flux. Lorsque le réseau est peu ou moyennement chargé, le routage multichemins permet d'augmenter le débit. En revanche lorsque la charge globale est importante et composé de flux élastiques, le routage multichemins s'avère problématique. L'utilisation d'un nombre élevé de ressources est critique et augmente la probabilité de blocage pour les flux naissants. Les auteurs semblent privilégier un routage monochemin en cas de charge élevée sur l'ensemble du réseau.

3.2.1.2 Répartition de la charge et formulation générale

Pour introduire le problème du partage de charge sur des chemins multiples, quelques notations sont nécessaires. Selon le mode d'équilibrage de charge considéré, les prérequis sont différents. Pour le mode prédictif, il faut disposer d'une série de matrices de trafic. Une matrice de trafic est notée $D = \{u(s,d)|s,d \in N\}$, chaque élement $u(s,d)$ représente la demande entre la paire de routeurs s et d. La première formulation des contraintes proposée ici est basée sur une vision par couple (source, destination), ainsi le terme $NHE(s,r,d)$ désigne l'ensemble des prochains sauts activés sur le routeur r pour le couple (s,d). Un élément de cet ensemble, noté $NHE_i(s,r,d)$, correspond à un prochain saut de r utilisé par le couple (s,d) (l'indice i permet de les différencier). Sur un domaine MPLS, les routeurs s et d désigneraient respectivement l'*ingress LSR* et l'*egress LSR*.
Notons $y_i^r(s,d)$ la fraction du flot entre s et d acheminée sur le lien $l \mid l.y = NHE_i(s,r,d) \in NHE(s,r,d)$ sortant du routeur r. Notons que la demande $u(s,d)$ attribuée à la paire (s,d) contribue à hauteur d'un débit de $u(s,d) \times y_i^r(s,d)$ sur le lien l du routeur r. Les contraintes sur les fractions de flux $\{y_j^r(s,d)\},\ \forall NHE_j(s,r,d) \in NHE(s,r,d)$ sont la conservation des flots et la non négativité, tel que :

$$\forall s \neq d, \forall r \in N \mid r \neq s,\ d : \sum_{\forall p \in N}^{\text{si } \exists i \mid NHE_i(s,p,d)=r} y_i^p(s,d) - \sum_{\forall\ NHE_j(s,r,d)} y_j^r(s,d) = 0 \tag{3.3}$$

$$\forall s \neq d : \sum_{\forall p \in N}^{\text{si } \exists i \mid NHE_i(s,p,d)=r} y_i^p(s,d) - \sum_{\forall\ NHE_j(s,r,d)} y_j^r(s,d) = \begin{cases} -1 \text{ si } r = s, \\ 1 \text{ si } r = d. \end{cases} \tag{3.4}$$

$$\forall r \in N, \forall\ NHE_j(s,r,d) \in NHE(s,r,d) : y_j^r(s,d) \geq 0 \tag{3.5}$$

L'équation (3.3) signifie simplement que chaque routeur r du réseau émet autant de trafic qu'il en reçoit pour toute paire (s,d). Si on intègre les demandes $u(s,d)$ dans l'équation (3.4), celle-ci signifie que d reçoit autant de trafic que s en a émis.
Une fois les contraintes définies, plusieurs objectifs d'optimisation sont possibles, par exemple : minimiser les délais, maximiser les débits ou minimiser l'utilisation maximum des liens, etc. Pour cela, il suffit de modéliser le critère représentant l'objectif. Par exemple si la métrique porte sur le ratio $r(y,D)$ maximum d'utilisation d'un lien (*Maximum Link Utilization*, MLU) :

$$r(y,D) = max_{\forall l \in E} \frac{\sum_{s,d \in N}^{l.y=NHE_i(s,r,d) \in NHE(s,r,d)} u(s,d) \times y_i^r(s,d)}{c(l)}$$

La connaissance de la matrice de trafic D permet par exemple d'adapter le routage en modulant la valuation des liens comme introduit dans le paragraphe 3.1.5.2. Le but est de minimiser le ratio $r(y,D)$.

Dans la référence (SGD05), Sridharan et al. proposent une approche ne nécessitant pas de modifications spécifiques sur le mécanisme de commutation traditionnel. En effet, la solution proposée agit en amont et permet de contrôler l'attribution d'un meilleur chemin à un préfixe IP. En d'autres termes, leur proposition se place dans la perspective de distribuer le trafic par une correspondance entre NH et préfixe. Les auteurs reformulent le problème du routage optimal sur les meilleurs chemins donné dans (WWZ01) pour une commutation par destination. Le dual de ce problème permet de déterminer la valuation à attribuer à chaque lien si la matrice de trafic est connue.
Wang et al. (WWZ01) proposent une méthode permettant de mettre en œuvre de l'ingénierie de trafic sans déployer un réseau overlay full mesh. Leur approche est dite *intégrée* : il s'agit de manipuler la valuation des liens du réseau pour adapter le routage aux demandes. Ils démontrent que leur modèle permet en théorie d'obtenir des résultats identiques à ceux envisageables avec des réseaux d'overlay full-mesh sans s'exposer au problème du passage à l'échelle. Ils prouvent, en utilisant la dualité de la formulation d'un problème de programmation linéaire, que le choix d'une valuation des arcs optimale est un problème équivalent. Ainsi, il est possible de résoudre un problème d'optimisation minimisant l'utilisation du lien le plus chargé en utilisant seulement les meilleurs chemins de coûts égaux.
Cependant, la résolution de ce problème implique que le partage de charge ne soit pas équitable comme avec ECMP. Par ailleurs, bien que le principe de sous-optimalité permette de ne pas considérer la source du trafic, le problème et les contraintes énoncées doivent être reformulés pour être compatibles avec une commutation par destination et au saut par saut. Sridharan et al. (SGD05) proposent une formulation adéquate permettant en outre de réduire la complexité de calcul. En revanche, le problème du partage de charge équitable entre routes de même coût est plus complexe à résoudre. Les auteurs profitent de l'hypothèse suivante : si de nombreux préfixes IP sont associés au même routeur de bordure, l'ABR de sortie, alors statistiquement la distribution de la charge peut être contrôlée en associant un sous-ensemble spécifique de NHs aux coût égaux à chaque préfixe.
Ce mécanisme nécessite la connaissance du trafic par préfixe, la granularité est donc plus fine que pour une approche par routeur de sortie. Par ailleurs, pour réduire la complexité de calcul, il est préférable de sélectionner uniquement un sous-ensemble restreint de préfixes. Les auteurs démontrent que le problème de la correspondance entre un préfixe et le sous-ensemble de NH permettant une distribution optimale est NP-complet. Ils proposent une heuristique de correspondance de type "min-max". Celle-ci fonctionne en deux étapes, d'une part les préfixes sont ordonnés puis traités par volume de demande décroissant, d'autre part, à chaque itération, un sous-ensemble de NH est attribué à chaque préfixe traité dans la perpective de minimiser une métrique donnée. Pour un NH donné, la métrique choisie représente le ratio maximum entre la portion de trafic attribuée et l'allocation de trafic optimale sur l'ensemble des NHs possibles entre la source et la destination correspondant au préfixe. Dans ce contexte itératif, le trafic alloué comprend le trafic déja attribué lors des itérations précédentes.
D'autres travaux, comme les recherches initiées par Gallager (BG87a) et Bertsekas (BG87b), introduisent une classe de problèmes d'optimisation orientée délais d'acheminement. Les auteurs formulent le problème du routage à délais optimaux. Cependant, la résolution de ce type de problème dépend du pas d'itération (une forme de constante globale), et ne s'avère réellement efficace que lorsque le trafic

est quasi stationnaire [6]. Nous étudierons dans la partie 3.2.2 les travaux de Vutukury et Garcia-Luna-Aceves (VGLA99) proposés pour répondre à ces problèmes.
Avant de présenter les méthodes de partage de charge distribuée dont la réactivité dépend de mesures *online*, nous allons présenter une catégorie de propositions dont l'approche est orthogonale. Le paragraphe suivant fournit un aperçu des contributions proposées dans les réseaux MPLS. Le RFC 2702 (AMA$^+$99) énonce un certain nombre de règles pour parvenir à mettre en place de l'ingénierie de trafic efficace.

Si l'on considère le paradigme classique du routage par destination, la formulation du problème est plus simple. Dans le cas où les ensembles de NHs utilisés pour la commutation du trafic en transit et du trafic local sont identiques, $NH(p,r,d) = NH(r,r,d)\ \forall p \in pred(r)$, la formulation du partage de charge peut s'exprimer de la manière suivante.
Sur un routeur r disposant de $n = |NH(r,r,d)|$ NHs actifs pour atteindre la destination d :

$$\{x_1^d, x_2^d, \cdots, x_j^d, \cdots, x_n^d\}_r$$

désigne le vecteur des proportions globales (pour le trafic de transit et local) associé à la destination d sur le routeur r. Le rang j indique le classement du chemin associé au j^e NH en fonction de la métrique C. Les proportions globales vérifient les contraintes de conservation et de non négativité :

$$\sum_{j=1}^{n} x_j^d = 1 \tag{3.6}$$

$$\forall j \in [1,n]\ x_j^d \geq 0 \tag{3.7}$$

La granularité de cette formulation est suffisante pour un routage où la commutation n'est pas dépendante de la source ou de l'interface d'entrée.

3.2.1.3 Partage de charge piloté par la source

Nous pouvons classifier les travaux d'ingénierie de trafic par équilibrage de charge en trois grandes catégories :

- Prédire les demandes moyennes (*predicted*) : l'ingénierie de trafic est mise en place pour optimiser les demandes représentatives sur un ensemble de matrices de trafic $\{D\}$ collectées dans le temps.
- Prévenir le pire cas (*oblivious*) : l'ingénierie de trafic prend en compte le pire des cas sur l'ensemble des matrices de trafic.
- Réagir en fonction des demandes instantanées (*online*) : l'ingénierie de trafic est réalisée à partir de mesures temps-réel obtenues via des sondes évaluant la qualité des routes.

Dans le cas où le réseau dispose d'outils de métrologie permettant de maintenir un historique des matrices de trafic évoluant dans le temps, il est possible d'estimer la charge à venir sur l'intervalle de temps

[6] Le lecteur est invité à lire la référence (Vut01) pour une description plus précise.

suivant. Les travaux de Zhang et al. (ZKT+05 ; ZLG+05) sont un exemple de ce type d'approche. La thèse de Casellas (Cas02) utilise par exemple la notion de bande passante effective. Lorsque la charge du réseau est relativement stable, en conditions normales, cette catégorie d'algorithmes permet d'obtenir des résultats efficaces. Cependant, lorsque les changements de charge sont brusques et importants, la seconde catégorie, le routage *oblivious*, permet d'anticiper les variations de charge ne correspondant pas au trafic moyen mesuré sur les périodes antérieures. Ces cas sont relativement fréquents, en particulier lors de l'occurence d'une panne car la charge induite par la redirection de trafic est imprévisible. Les propositions avancées par Applegate et Cohen (ABC04 ; AC03) vont précisement dans ce sens. En revanche, cette approche n'est pas optimale en condition normale lorsque la charge correspond au cas moyen.
D'autres protocoles relativement récents tel que *MPLS Adaptative Trafic Engineering* (MATE, (EJLW01)) et TeXCP (KKDC05) utilisent des sondes permettant d'estimer en temps-réel l'utilisation d'un LSP. A la différence de MATE, TeXCP est un protocole distribué ne nécessitant pas la présence d'un *oracle* pour permettre à chaque source de connaître l'état du réseau. Les sources sont informées par les routeurs de cœur de l'utilisation des liens par un *feedback* similaire à XCP (KHR02). Par ailleurs, TeXCP privilégie l'utilisation des chemins les plus courts car la métrique utilisée a tendance à allonger les délais d'acheminement contrairement à MATE qui a pour objectif de minimiser les délais.
Le protocole *Optimized Multipath for MPLS* (OMP-MPLS, (Vil99a)) propose une démarche analogue sans garantie sur la stabilisation de la procédure distribuée.
Une proposition plus récente, *Common-case Optimization with Penalty Envelope* (COPE, (WXQ+06)), affirme que les protocoles réactifs tels que TeXCP peuvent, durant de larges périodes transitoires, être pénalisés par des changements de charge brusques et importants. Leur démarche est un compromis entre les deux premières catégories énoncées dans ce chapitre. Les auteurs utilisent à la fois la méthodologie classique par prédiction via une séquence de matrices de trafic obtenues sur des intervalles de temps consécutifs (*common case*), et proposent une nouvelle approche basée sur le concept d'enveloppe couvrante (*penalty envelope*). Le principe est de restreindre l'espace des solutions tel que les proportions $x_i^r(s,d)$ soient optimales à l'intérieur de cette enveloppe. En pratique, il s'agit d'une contrainte supplémentaire permettant d'inclure dans l'analyse les pires cas possibles. Les dimensions de l'enveloppe sont déterminées en fonction de toutes les matrices de trafic possibles, dont celles obtenues pour la prédiction. Le but est de minimiser la fonction objective sur les matrices de prédiction, en vérifiant que la contrainte liée à l'enveloppe couvrante soit satisfaite quel que soit le trafic.

3.2.2 Partage de charge au saut par saut

Dans cette partie, nous nous focaliserons sur le problème du partage de charge avec un routage IP saut par saut avec une correspondance de commutation par destination. La répartition des flux est effectuée à chaque saut. Chaque routeur est potentiellement capable, si plusieurs prochains sauts sont activés vers une destination, de distribuer le trafic. Pour une destination donnée, la commutation d'un paquet sur un NH actif dépend de la proportion qui lui a été attribuée. Les paquets sont acheminés selon leurs destinations et les proportions établies sur chaque routeur. Le choix de l'aiguillage est calculé

à chaque saut.
La figure 3.3 fournit un aperçu des différentes problématiques liées au partage de charge. Le protocole ECMP est un exemple de protocole de routage proportionnel statique : pour une destination d donnée, les bornes caractérisant les proportions sont déterminées selon un mode équitable sur l'ensemble des meilleurs prochains sauts. Soit $\{x_1^d, ..., x_j^d, ..., x_n^d\}$ le vecteur de proportions utilisé par un routeur s ayant activé n prochains sauts vers la destination d. Un partage équitable signifie simplement que les proportions sont égales : $\forall i, j \in (1, ..., n)$, $x_i^d = x_j^d$. Cette caractéristique assure la stabilité de la commutation. ECMP ne provoque pas d'oscillation de charge car il ne prend pas en compte la charge des liens.
Les algorithmes dynamiques utilisent des indicateurs pour évaluer l'offre résiduelle du réseau. Il peut s'agir d'une batterie de mesures telle que la bande passante disponible, le taux de perte, le taux d'utilisation des files d'attente, la puissance de calcul disponible, etc. La granularité de ces mesures peut prendre en compte l'origine et la destination du trafic ou d'autres critères associés à la qualité de service.
L'ensemble des procédures dynamiques présentées dans ce paragraphe utilise la règle LFI. L'ensemble des chemins multiples activés vérifient une condition de stricte décroissance du meilleur coût sur les routeurs adjacents. Cette règle permet de relâcher le critère d'égalité des coûts utilisé avec ECMP.
Les techniques de répartition de charge au saut par saut étudiées ici ne sont pas dépendantes de la règle définissant la condition suffisante utilisée pour vérifier l'absence de boucles de routage.

3.2.2.1 Indicateurs et mesures

Plusieurs indicateurs peuvent être utilisés pour mesurer la qualité d'un chemin. Les performances d'un chemin en termes de latence, probabilité de blocage ou de perte, dépendent directement du lien *offrant* la qualité la plus dégradée de ce chemin. Une métrique de type convexe ou concave sur l'ensemble des liens correspondant au chemin permet d'associer une valeur indicative de la performance à chaque prochain saut. Cette valuation reflète généralement l'état du lien en termes de bande passante utilisée. Dans la suite, nous noterons D_l le débit mesuré sur un lien l en fonction de l'unité de temps t choisie tel que $D_l = \frac{q(l)}{t}$. La valeur $q(l)$ désigne la quantité de trafic acheminée via le lien l durant la période de temps t.
Villamizar propose dans (Vil99b) une technique intitulée *OSPF-OMP*. Ses travaux utilisent une mesure de bande passante *Equivalent load*, notée $\rho(l)$, adaptée aux liens chargés subissant des pertes. L'auteur propose d'intégrer dans ses mesures de débit la probabilité de perte P sur chaque lien $l \in E$. En pratique, la probabilité de perte représente le taux de perte enregistré sur la période de temps écoulée. Pour évaluer la bande passante réelle, Villamizar utilise une approximation de l'estimation par dérivation de Mathis et al. (MSM97). Le débit BW d'un flux TCP est fonction des caractéristiques de ce flux : le *Round Trip Time* (RTT), la taille du segment maximal (MSS) et une constante C caractérisant le mécanisme d'acquittement utilisé :

$$BW = \frac{MSS \times C}{RTT \times \sqrt{P}}$$

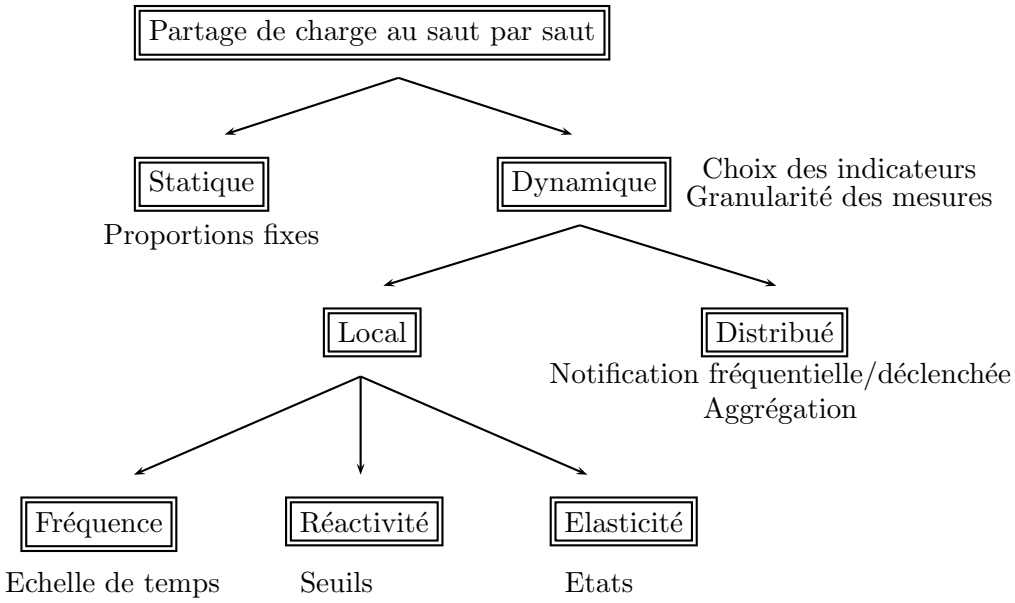


FIG. 3.3 – Le partage de charge au saut par saut : méthodologies

Ainsi, Villamizar définit une métrique permettant de déterminer le lien le plus chargé :

$$\rho(l) = max(\frac{D_l}{c(l)}, \frac{D_l}{c(l)} \times K \times \sqrt{P})$$

Le scalaire K permet de déterminer le seuil de probabilité de perte auquel la mesure *Equivalent load* devient sensible. Si $K > \frac{1}{\sqrt{P}}$, alors $\rho(l) > \frac{D_l}{c(l)}$. Ainsi, avec $K = 10$, $\rho(l)$ est supérieur à la mesure classique du débit sur le lien l si la probabilté de perte dépasse 1%.
Un autre indicateur possible est le délai marginal $T'_l(f_l)$ associé à un lien l et à une charge u_l le traversant, celui-ci peut être obtenu en dérivant la fonction suivante :

$$T_l(u_l) = \frac{u_l}{c(l) - u_l} + u_l \times d(l)$$

Cet indicateur est néanmoins instable si le lien l est très chargé ($c_l - u_l < \epsilon$). Des techniques telles que celles proposées dans (CAT90) permettent néanmoins d'estimer le délai marginal avec des mesures actives en temps-réel. Cet indicateur est utilisé par Vutukury et Gallager dans (VGLA99). Le paragraphe 3.2.2.3 décrit l'heuristique de partage associé à cet indicateur. La distribution de l'estimation du délai marginal est assurée par la diffusion de vecteurs de délai LSU décrit dans le paragraphe 1.2.3.3. Le délai marginal d'un chemin est calculé avec une métrique additive sur l'ensemble des liens le constituant.
Un autre type de méthode consiste à prédire le trafic dans l'intervalle à suivre grâce à des mesures récoltées précédemment. Pour cela, la moyenne mobile pondérée de façon exponentielle (*exponentially weighted moving average*, EWMA) notée P_l est souvent utilisée.

$$P_l(t) = \beta \times D_l(t) + (1 - \beta) \times P_l(t - 1)$$

Selon la valeur du scalaire β, la prédiction tient plus ou moins compte des mesures récentes. Par exemple pour $\beta = 1$, seule la dernière mesure compte.

3.2.2.2 Distribution de l'information et déclencheur

Afin de distribuer l'état de charge de chaque lien à l'ensemble du réseau, un protocole de notification doit être mis en place. L'état d'un lien signifie ici un ensemble de mesures s'y reportant : pertes, bande passante, etc.
La diffusion de cette information permet d'associer à chaque NH une mesure correspondant à l'état du chemin auquel il est associé. Le mécanisme d'inondation d'*OSPF-OMP* utilise les options des *Opaque LSA*, *OLSA*. Le format exact est décrit dans le RFC 2370 (Col98). Plus précisement, un message OLSA contiendra un champ *LSA-OMP-LINK-LOAD* ou *LSA-OMP-PATH-LOAD* selon la nature interne ou externe des routes. En intradomaine, ce champ est caractérisé par trois indicateurs : le taux d'utilisation du lien $\frac{D_l}{c(l)}$, le taux de perte P et la capacité du lien $c(l)$. En interdomaine, si disponible, les informations contenues dans le *LSA-OMP-PATH-LOAD* sont dépendantes du chemin et sont notifiées par les routeurs de bordure (ABR) : il s'agit du taux d'utilisation du lien le plus chargé (métrique convexe : max), du taux de perte avec une métrique multiplicative et du lien de plus petite capacité (métrique concave : min). La fréquence de l'innondation de l'état des liens avec les messages OLSA dépend de la différence enregistrée avec les mesures précédemment effectuées : le delta de variation. En pratique, il s'agit d'un filtre relatif aux mesures précédentes et au temps écoulé depuis la dernière inondation.
Chaque routeur associe une structure *next hop structure* à chaque destination, un ensemble de chemins. Pour les routes interdomaine, une seule structure par ABR est nécessaire. Cette structure contient l'ensemble des informations relatives aux chemins : l'ensemble des liens les composant, les NH associés, et surtout le segment de chemin le plus chargé (*critically loaded segment*). Ce segment représente le lien [7] le plus chargé selon la mesure *Equivalent load* reçue ou enregistrée localement. Ce segment critique permet de choisir, lorsque plusieurs NHs sont disponibles, le prochain saut à *décharger*. Le paragraphe suivant décrit la méthodolgie proposée par Villamizar.
Les auteurs de (GZR03) proposent une approche similaire en termes d'indicateurs de débit, ils utilisent la mesure *Equivalent Load*. Néanmoins, ils mettent en avant deux défauts majeurs de *OSPF-OMP* : la *next hop structure* est coûteuse en terme d'espace de mémorisation et l'inondation via les OLSA peut être également un facteur important de complexité.
A la différence d'OMP, la technique de multiroutage adaptative présentée dans (GZR03) utilise des structures de données simplifiées, car locales. Les auteurs proposent une vague de dissémination récursive intitulée *backpressure* pour informer le voisinage en amont. Le principe et les mécanismes d'agrégation des messages *backpressure* sont les suivants :
pour chaque lien sortant $\{s, v_0\}$, la procédure proposée nécessite que le routeur s dispose de deux informations : la charge *Equivalent load* du lien, $\rho(\{s, v_0\})$, et la *part de responsabilité* du trafic émis via ce lien dans la charge des liens à un saut $\{v_0, v_i\}$ tel que $v_i \in succ(v_0)\ \forall i | 0 < i < k^+(v_0)$ et plus généralement p sauts en aval. La première mesure est entièrement locale, la seconde est obtenue via le

[7] Il peut s'agir de plusieurs liens, un segment de chemin

mécanisme de signalisation. Un message *backpressure* $B_{v_0 \to s}$ transmis de v_0 vers s prend cette forme :

$$B_{v_0 \to s} = f(\rho(\{v_0, v_1\}), ..., \rho(\{v_0, v_{k^+(v_0)}\}, B_{v_1 \to v_0}, ..., B_{v_{k^+(v_0)} \to v_0}))$$

La fonction f a pour objectif d'intégrer les $2k^+(v_0)$ informations de charge en une seule valeur. Soit $g_i = max(\rho(\{v_0, v_i\}), B_{v_i \to v_0}),\ \forall 0 < i < k^+(v_0)$ la fonction agrégeant la charge locale et la charge en aval d'un lien sortant $\{v_0, v_i\}$, permettant de réduire l'espace à $k^+(v_0)$ paramètres. Les auteurs proposent d'utiliser une fonction additive pondérée telle que :

$$B_{v_0 \to s} = \sum_{v_i \in succ(v_0)\ s} \frac{\beta_i(s)}{\beta_i} \times g_i$$

avec $\beta_i(s)$ désignant la charge totale émise par s via le lien $\{v_0, v_i\}$ et β_i désignant la charge totale transitant via le lien $\{v_0, v_i\}$. Un tel calcul nécessite une granularité de mesure liée à l'interface d'entrée du trafic. Le routeur v_0 doit disposer d'une matrice de mesure de trafic correspondant au $\beta_i(s)$. Cette matrice carrée de dimension $k^+(v_0)^2$ contient, à l'indice (s, i), le trafic provenant de l'interface d'entrée s transitant via le lien $\{v_0, v_i\}$ tel que $\sum_{s \in pred(v_0)} \beta_i(s) = \beta_i$ désigne tout le trafic émis via le lien $\{v_0, v_i\}$: la somme d'une colonne de la matrice. La figure 3.4 résume la procédure de notification. Le routeur v_0 a 4 voisins v_1, v_2, v_3 et $v_4 = s$, ceux-ci sont connectés à la fois par des liens entrants et sortants.

Le message $B_{v_0 \to s}$ est un condensé des informations de charge collectées localement par v_0 et des informations obtenues via les messages *backpressure* des voisins $v_i, i = \{1, 2, 3, 4\}$. Ces routeurs ayant eux-mêmes opéré la même procédure de calcul distribué. Le mécanisme est récursif, si bien que s dispose d'une quantification par lien sortant lui permettant de réguler son trafic.

Les auteurs proposent d'utiliser deux échelles de temps T_C et T_B pour respectivement contrôler la répartition de charge et déterminer la fréquence de signalisation. Ils recommandent d'utiliser une relation linéaire tel que, si on note D le diamètre du réseau, la fréquence de contrôle dépende linéairement de la fréquence de signalisation :

$$T_C \cong D \times T_B$$

Ce choix permet d'assurer que les informations de congestion distantes soient prises en compte au plus tard dans la boucle de contrôle suivante.

Les travaux d'évaluation présentés dans (MZK01) analysent différentes techniques déterminant la fréquence de rafraîchissement de l'état des liens. Trois approches sont évaluées : rafraîchissement périodique (*time based*), rafraîchissement sur seuil et par classe (*threshold and class based*), et rafraîchissement hybride. Par ailleurs, la périodicité de la notification, bien que similaire en méthodologie à la réactivité des modifications de commutation, est généralement découplée des mécanismes algorithmiques de changement de proportions.

Le protocole *OSPF-OMP* utilise par exemple un ajustement dont le pas d'itération est plus *agressif* en termes de variations de charge qu'en termes d'inondation d'états des liens. Autrement dit, la cadence de modification des proportions est plus élevée que celle de la distribution de l'état des liens. Lorsque

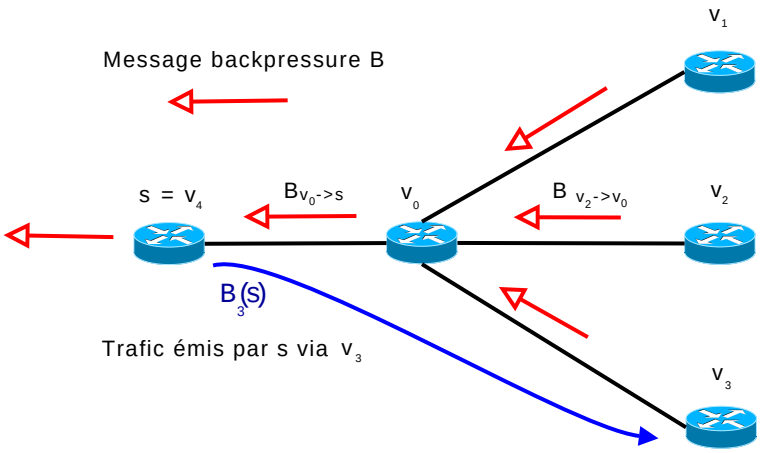


FIG. 3.4 – Vague de notification en amont

le réseau est peu chargé, cela permet d'éviter de continuellement l'inonder d'OLSA inutiles.

3.2.2.3 Algorithmique de décision

Ce paragraphe présente des exemples de mécanismes algorithmiques utilisés pour la répartition de charge. Pour commencer, nous décrirons la procédure d'ajustement de charge proposée dans *OSPF-OMP*.

L'ajustement de débit doit être limité pour assurer la stabilité de la commutation. Villamizar utilise un algorithme incrémental basé sur le calcul du *critically loaded segment*. La vitesse d'ajustement dépend de quatre facteurs :

- la différence en termes de charge entre le chemin le moins chargé et le segment critique,
- la mesure Equivalent load du segment critique,
- le type de destination (interne ou externe),
- le temps écoulé depuis le dernier ajustement.

Pour chaque chemin contenu dans la *next hop structure*, l'algorithme fait intervenir trois variables : la valeur courante de partage (*trafic share*), la quantité déviée vers ce chemin (*move increment*) et un compteur du nombre de changements dans la même *direction* (*move count*). Typiquement, la variable *move increment* est initialement fixée à 1%. Lorsque le segment critique change, les chemins contenant ce segment sont examinés en priorité car il faut déterminer le nombre de chemins affectés. Les chemins affectés ne sont pas concernés par un changement sur la variable *move increment*. En revanche, pour les chemins ne contenant ni le segment critique courant, ni le segment critique de l'itération précédente, la variable *move increment* est augmentée récursivement [8] et *move count* est incrémenté. Pour les chemins ne contenant pas le segment critique courant, mais qui contenaient un tel segment à l'itération précédente, la variable *move increment* est remplacée par la plus petite valeur *move increment* des chemins contenant le segment critique courant, à moins que celle-ci soit déjà plus petite et que *move count* soit

[8] Se référer à l'annexe du document (Vil99b) pour les détails techniques

positionné à zéro. Dès lors que l'ajustement des variables *move increment* et *move count* est réalisé, il suffit de prendre en compte les changements enregistrés pour ajuster la variable de distribution *trafic share*. Si le chemin contient le segment critique courant, alors la valeur courante de partage est réduite en fonction de la somme des variables *move increment* des chemins ne contenant pas le segment pour une destination donnée. La stabilité de leur algorithme de décision est assurée par le caractère incrémental du processus : les changements de proportions sont relativement lents au début et s'accélèrent si la variable *move count* augmente. Cette variable permet d'accélèrer le processus lorsque le changement de proportion se confirme d'une itération à l'autre.

L'algorithme d'équilibrage de charge proposé dans (GZR03) diffère d'OMP dans la mesure où la notion de chemin est réduite à une vision strictement locale : les liens sortants. Ainsi, l'objectif de la fonction d'équilibrage est d'harmoniser les valeurs $g_i = \max(\rho(l), B_{v \to s})$ calculées pour chaque lien sortant $l = \{s, v\}$ d'un routeur s. La distribution de charge est effectuée de telle sorte que la fonction g_i recherche un point d'équilibre entre les liens sortants. De même que pour *OSPF-OMP* la répartition de charge est réalisée par destination avec une fonction de hachage CRC 16 (CWZ00). De plus, la régulation du trafic est progressive pour garantir la stabilité du processus. Initialement, le changement de proportion est lent mais croît exponentiellement si le changement de direction (*move count*) persiste. D'une itération à la suivante, lorsque le lien l de plus grande valeur g_i change, la variation de proportion retourne à son état initial.

Nous allons à présent décrire de manière concise et formelle la solution de répartition proposée dans (VGLA99). Il s'agit d'une heuristique distribuée utilisant la diffusion de l'estimation des délais marginaux.

Initialement, chaque élément x_j^d du vecteur de proportion, calculé vers d sur un routeur s, correspondant à un j^{eme} NH tel que $v = NH_j(s, d)$, satisfait l'équation suivante :

$$x_j^d = \frac{1 - \frac{C_1(v,d)+w(s,v)}{\sum_{v' \in NH(s,s,d)} C_1(v', d) + w(s, v')}}{|NH(s, s, d)| - 1}$$

La procédure d'assignement de charge dynamique procède en quatre étapes :

- Déterminer avec la métrique dynamique le délai marginal le plus court $C_1(s, d)$ et le NH correspondant $NH_1(s, d) = v$.
- Pour tous les voisins $v' = NH_j(s, d) \in NH(s, s, d)$ calculer $a_j(v') \leftarrow C_1(v', d) + w(s, v') - C_1(s, d)$
- Calculer la variation du changement de proportion $\Delta \leftarrow \frac{1}{2} min(\frac{x_j^d}{a_j(v')} | v' \in NH(s, s, d) \wedge a_j(v') \neq 0$
- Pour tous les voisins $v' = NH_j(s, d) \in NH(s, s, d)$, $x_j^d \leftarrow x_j^d - \Delta \times a_j(v')$
- Pour $v = NH_1(s, d)$, $x_1^d \leftarrow x_1^d + \sum_{j=2}^{j=|NH(s,s,d|} \Delta \times a_j(v')$

Cette heuristique vérifie la sémantique : *Plus un chemin présente un délai court, plus la proportion qui lui est attribuée augmente.* Cet algorithme incrémental évolue selon un raisonnement complémentaire aux deux approches précédemment évoquées. Plutôt que de relâcher une proportion de charge sur le

chemin le plus chargé, cet approche favorise le chemin de meilleur délai.

3.2.3 Interactions avec TCP

En pratique, l'équilibrage de charge dynamique nécessite l'utilisation d'un module de partage permettant de répartir le trafic selon la granularité souhaitée. Pour cela, le trafic peut être partagé à plusieurs niveaux.
La partage de charge au niveau paquet, par exemple avec un ordonnancement de type *round robin*, fréquentiel ou probabiliste, est pratique pour assigner avec précision la proportion de charge sur chaque prochain saut. Malheuresement, ce type d'ordonnancement ne convient pas aux flux TCP.
En effet, lorsque les chemins sont hétérogènes en termes de délais, une telle granularité peut déséquencer un nombre élevé de paquets. Une série de paquets consécutifs, appartenant à une même fenêtre TCP (rafale ou *burst*), peut être désordonnée sur chaque routeur capable de répartir ces paquets sur plusieurs interfaces de sortie pour une destination donnée. Une connexion TCP interprète ce désordre comme une perte, c'est-à-dire comme un signe de congestion, et réduit alors la fenêtre de congestion. Cette confusion peut dégrader les performances des flux TCP en termes de débit et de délai d'acheminement.
Le RFC 2992 (Hop00) fournit une excellente description de ce problème avec ECMP. Bien que ECMP soit implémenté avec un partage de charge par flux, lorsque les proportions évoluent, il est possible que les paquets, appartenant à une fenêtre TCP donnée, soient soumis à ce problème. En effet, si ces paquets sont commutés lors de l'activation d'un changement de proportions dans la FIB, ils peuvent être acheminés sur des routes différentes.
Le partage de charge par flux commute chaque flux ou agrégat de flux sur un unique chemin spécifique à la classe de flux concernée. Un flux peut se définir à différentes échelles de précision : adresse source, adresse destination, port, protocole, etc. Ce type de commutation utilise généralement une identification par fonction de hachage et permet alors d'éviter le problème du déséquencement des paquets. Par exemple, avec hachage CRC 16 générant une valeur comprise entre 0 et 65355 pour *OMP-OSPF*.
La référence (CWZ00) présente une évaluation de plusieurs méthodes de hachage dans le contexte du partage par flux. La précision du partage de ce type de méthode est inférieure aux méthodes *par paquet*, particulièrement lorsque, statistiquement, le nombre de classes de flux est faible. Par ailleurs, dans ce cas l'équilibrage de charge peut être relativement lent et ne pas suffire pour réagir à de fortes ou rapides variations de charge.
La technique FLARE introduite dans (KKSB07) constitue un compromis entre ces deux niveaux de granularité. Ce procédé analyse le délai marginal entre les chemins parallèles et le temps entre deux rafales TCP consécutives. Le partage de charge est réalisé par burst plutôt qu'en utilisant un procédé d'identification par flux. Cette méthode présente donc l'avantage d'être relativement précise en terme de granularité et, dans la plupart des cas, permet d'éviter les problèmes liés aux déséquencements des paquets.
Nous proposons un procédé différent pour un partage au niveau flux. Notre approche permet d'identifier un flux en lui attribuant une étiquette. Celle-ci peut prendre différentes significations.
Nous considérons que le nombre de flux, ou le nombre de classes de flux traversant un routeur est

statistiquement suffisante pour obtenir une granularité permettant un partage de charge précis. La précision escomptée dépend largement de l'intervalle de temps considéré. Nous nous focalisons sur des réactions basées sur une échelle de temps relativement large, de l'ordre de la seconde.
Dans un environnement réel l'étiquette pourrait être positionnée sur les routeurs d'accès ou à l'entrée du domaine. En pratique, dans nos simulations, tous les paquets appartenant à un flux donné sont estampillés (via un *tag*) à la source avec un nombre aléatoire $n \in [0, N]$. Ce *tag* est alors similairement exploité par chaque routeur offrant un choix entre divers NHs pour la destination donnée. Les routeurs commutent le paquet vers le prochain saut dont les bornes relatives aux proportions contiennent la valeur de l'étiquette. Sur un routeur r et pour une destination donnée d, si la source émettrice du paquet a estampillé le paquet avec la valeur θ, alors le paquet est commuté sur l'interface de sortie $NH_j(r, d)$ tel que :

$$N \sum_{i=1}^{i=j-1} x_i^d \leq \theta < N \times [\sum_{i=1}^{i=j-1} x_i^d + x_j^d]$$

Ce tag pourrait aussi être recalculé ou utilisé différemment sur chaque routeur pour uniformiser le choix du rang de l'interface de sortie si le nombre de sauts est suffisamment grand. Dans nos simulations, le tag conditionne directement le coût de la route utilisée : plus le tag d'un paquet est proche de sa borne maximale N, plus celui-ci a de chances d'être systématiquement dévié de la route optimale.
Techniquement, l'en-tête IP du paquet doit contenir une valeur comprise dans un champ de $\lceil log_2 N \rceil$ bits. Avec IPv6 le champ *flow label* pourrait contenir un tel tag alors que le champ *ToS/DSCP* pourrait être utilisé avec IPv4.
Il est à noter que le tag pourrait être utilisé pour implémenter une forme de routage à qualité de service. En effet, si ce tag est associé à une classe d'application spécifique, et que chaque routeur utilise une commutation cohérente, les prochains sauts vérifiant certaines contraintes de QoS (ou par induction les routes) peuvent être systématiquement associés à un type de service. Par exemple, les micro-flux seraient systématiquement commutés sur les chemins les plus courts alors que les flux longs avec une fenêtre d'émission agressive seraient acheminés sur des routes non optimales. Dans (SRS99), les auteurs proposent une analyse détaillée du problème du routage avec des flux *éléphants*. Ils préconisent d'utiliser les routes non minimales pour ce type de flux alors que les flux *souris* seraient inutilement perturbés sur ces routes non optimal.

3.2.4 Déviation de la charge et congestion

Cette partie présente nos propositions pour mettre en œuvre un équilibrage de charge réactif. Nous présenterons un algorithme de routage proportionnel basé sur une analyse locale de la bande passante utilisée. Les travaux introduits dans cette partie se fondent sur les hypothèses suivantes :

- la valuation attribuée à chaque arc est choisie pour optimiser le routage dans le cas moyen, c'est-à-dire lorsque le trafic est normal. En d'autres termes, le routage et les proportions de partage sont initialement dépendants des demandes habituelles.
- le changement de proportion n'intervient que dans le cas où la situation devient critique, c'est-à-

dire lorsque qu'un lien est excessivement chargé.
L'objectif principal de nos contributions est de mettre en avant l'intérêt de la diversité des chemins pour faciliter la redirection de charge en cas de congestion.

3.2.4.1 Introduction et moniteur

Le choix de l'échelle de temps (notée t en secondes) est capitale pour l'interprétation de nos mesures. Ce choix est d'autant plus important que nos mesures sont des débits dont le calcul est relatif à un temps d'analyse donné. L'analyse est locale à chaque routeur mais l'information peut éventuellement être relayée aux routeurs en amont. Les décisions consécutives aux mesures de débit conditionnent la répartition de charge. Néanmoins, il faut bien différencier les mesures et le *monitoring* à proprement parler. En effet, les mesures peuvent concerner de multiples indicateurs à une certaine échelle de temps t_1 alors que les actions qui en découlent sont prises à une échelle de temps t_2 plus large. Les mesures sont généralement moins coûteuses en consommation CPU que les actions consécutives, d'où l'intérêt de découpler ces deux tâches pour obtenir par exemple des moyennes soumises à certains traitements statistiques adéquats.
La granularité de l'analyse est également un critère prépondérant. L'analyse des débits peut tenir compte d'un ensemble de paramètres tels que : l'interface d'entrée, la destination, la source, le type de flux, etc.
Les deux premiers paramètres retiennent notre attention dans la mesure où ils sont directement liés à la structuration des tables de routage que nous avons définies. Ainsi sur chaque lien $l \in E$, la mesure du débit enregistre la valeur suivante en octets par seconde :

$$D_l(d,p) = \frac{q_d^p(l)}{t}$$

où $q_d^p(l)$ désigne la quantité de trafic (en octets), transitant sur le lien l, et provenant de l'interface p à destination de d sur le dernier intervalle de temps de mesure t.
Les débits $D_l(d,p)$ nous permettent de connaître précisément les couples {destination, interface d'entrée} générant les volumes de trafic les plus importants. De cette manière, il est possible de déterminer sur quel ligne de routage il faut agir dans les tables construites via le processus de validation DT(p).
Chaque routeur dispose d'un ensemble de mesures dont le cardinal dépend du nombre d'interfaces et des dimensions du réseau. Au pire, sur un routeur r donné, on obtient au maximum $k^+(r) \times k^-(r) \times |N-1|$ mesures de débits.

3.2.4.2 Mise en œuvre

Pour la collecte des mesures, l'échelle de temps choisie est soumise à des contraintes spécifiques et influe directement sur le coût processeur. Le degré de précision des mesures est limité pour éviter une sollicitation trop importante de la mémoire ou du processeur. En pratique, dans nos simulations, nous avons observé que ce degré d'analyse est peu utile pour la répartition de charge locale. Cet indicateur est davantage destiné à la notification distribuée d'une congestion.

Dans nos simulations, le *monitoring* ne considére pas les mesures à la granularité du couple (d, p). Le débit D_l d'un lien l s'exprime alors :

$$D_l = \sum_{p \in pred(r),\ d \in N} D_l(d, p)$$

r désigne le routeur duquel sort le lien l. Si ce débit excède une certaine proportion $\alpha \in]0, 1]$ de la capacité $c(l)$ du lien l, le contrôleur de répartition de charge déclenche le changement des proportions [9]. La condition de réaction est la suivante :

$$D_l > \alpha \times c(l)$$

Le choix du scalaire α conditionne le niveau de réactivité. Soit le moniteur de charge attend que le lien soit réellement surchargé (α proche de 1), soit le moniteur de charge tente d'anticiper la congestion ($\alpha << 1$). Le scalaire α est associé à l'intervalle de temps choisi pour réaliser les mesures. Sur un intervalle de temps suffisamment long, par exemple supérieur ou égal à la seconde, α ne doit pas être trop proche de 1. En revanche, pour détecter des micro-congestions ($t << 1s$), α peut être proche de 1. Deux raisons justifient ces choix : les réseaux d'opérateurs sont en général surdimensionnés et le trafic est majoritairement composé de flux TCP. Sur de larges intervalles de temps, le volume de la charge d'un lien haut débit est relativement faible.
Le choix de l'échelle de temps est un problème fondamental. Par exemple, un trafic gigabit commuté sur un lien saturé peut remplir une file d'attente de *600 000* bits (*75* paquets de *1000* octets) en *0.6* milliseconde.
A petite échelle, on constate que les rafales de paquets induites par les flux TCP provoquent des micro-congestions. Pour que le moniteur puisse les détecter, t devrait être de l'ordre de la milliseconde.
Or une période d'analyse si courte ne serait pas réaliste. Cela induirait une charge trop importante pour le processeur. Nos procédures s'appliquent plutôt aux congestions persistantes dont la durée est de l'ordre de la seconde. Au niveau de la couche réseau, les solutions d'ordonnancement des files d'attente par classe de service ou avec des stratégies telles que RED sont plus adaptées à la régulation du trafic sur de petits intervalles de temps.

3.2.4.3 Répartition statique

Dans ce paragraphe, nous présentons deux distributions possibles pour une répartition de charge statique. Nous nous sommes inspirés du mode de répartition le plus simple et pour lequel il existe une mise en œuvre réelle : un partage proportionnel et équitable entre chemins de coûts égaux avec ECMP. Cependant, lnotre processus de validation issue, DT(p), autorise l'utilisation de multichemins aux coûts inégaux. Ainsi, nous avons envisagé la distribution suivante : chaque prochain saut se voit attribuer une proportion relative au coût du chemin qu'il représente. Le terme $m(p) =\mid NH(p, s, d) \mid \leq k^{+}_{DT}(s, d)$ désigne le nombre de prochains sauts validés pour les paquets provenant de l'interface p.

[9] Les actions associées seront détaillées dans le paragraphe 3.2.4.4.

Nous proposons deux formules pour définir la proportion attribuée au j^{eme} prochain saut selon l'interface entrante. Formellement,

$$\{x_1^d(p), x_2^d(p), ..., x_j^d(p), ..., x_n^d(p)\}_s$$

désigne le vecteur des proportions attribuées aux prochains sauts $NH_j(s,d) \in NH(p,s,d)$ activés pour l'interface entrante p vers la destination d sur un routeur s. Le terme $n = m(s)$ désigne le nombre maximal de prochains sauts activés. Notons que si $NH_j(s,d) \notin NH(p,s,d)$ n'est pas un NH actif sur s pour le prédécesseur p, alors la proportion correspondante est nulle.
Sur un routeur s, pour une interface d'entrée p et vers une destination d si $m(p) > 1$:

$$x_j^d(p) = \frac{\sum_{\forall NH_i(s,d) \in NH(p,s,d)} C_i(s,d) - C_j(s,d)}{(m(p)-1) \times \sum_{\forall NH_i(s,d) \in NH(p,s,d)} C_i(s,d)} \tag{3.8}$$

$$x_j^d(p) = \frac{1}{C_j(s,d)} \times \frac{1}{\sum_{\forall NH_i(s,d) \in NH(p,s,d)} 1/C_i(s,d)} \tag{3.9}$$

La formule (3.9) favorise davantage les chemins de meilleurs coûts que la formule (3.8) lorsque *m(p)* est supérieur à 2. Ces deux formules considèrent une distribution vérifiant l'intégrité de la somme des proportions : $\forall p, d \ \sum_{j=1}^{n} x_j^d(p) = 1$.
Les résultats de simulation que nous avons obtenues semblent indiquer que le routage proportionnel statique n'est pas efficace. Néanmoins, il peut être utile de considérer ce calcul de proportions par exemple pour déterminer un état initial. Sur des réseaux dont la valuation des liens est dépendante d'une politique d'ingénierie de trafic, les coûts des chemins ne sont pas nécessairement adaptés à un routage sur des prochains sauts multiples.
Par ailleurs, l'algorithme de calcul de multichemins peut utiliser des métriques basées sur des connaissances a priori du trafic et du dimensionnement du réseau. Le partage de charge sur les diverses routes peut être réalisé selon deux types de mesures : avec des métriques statiques sur de grands intervalles de temps et avec des métriques dynamiques pour un routage proportionnel sur de petits intervalles de temps. Les routes peuvent aussi être précalculées et positionnées de proche en proche sur des informations statiques (délais de propagation, capacités des liens, etc). Pour nos simulations, nous avons considéré le cas d'une métrique additive. Typiquement, la valuation w d'un arc e est une composition linéaire de la bande passante et des délais de propagation de chaque lien : $w(e) = \frac{1}{c_i} \times \alpha + d_i \times \beta$.
La métrique peut également tenir compte, pour une destination donnée, du nombre de liens communs entre chemins primaires et alternatifs. Les chemins transverses avant présentent nécessairement une intersection avec les chemins primaires. Il est donc possible de pénaliser l'utilisation de la composition avant. Par exemple, en modifiant la valuation des arcs incriminés au moyen d'un scalaire λ. Soit e un arc appartenant à la branche d'un arbre des plus courts chemins, il suffit d'utiliser une valuation

spécifique pour la composition avant :

$$w(e) = \lambda \times (\frac{1}{c_i} \times \alpha + d_i \times \beta)$$

3.2.4.4 Répartition dynamique

Cette partie est consacrée aux différentes méthodes de répartition de charge dynamique que nous avons envisagées. Nous nous sommes focalisés sur deux modèles : dans un premier temps, nous décrivons un algorithme de partage de charge uniquement basé sur des informations locales, puis nous introduisons le principe de notification quantitative vers les routeurs en amont.
Dans les deux cas, la problématique générale reste semblable : en fonction des informations issues des mesures, comment modifier dynamiquement les proportions attribuées à chaque prochain saut pour distribuer au mieux l'excès de charge constaté sur un lien ?
La différence entre répartition dynamique *locale* et répartition dynamique distribuée est définie par la nature des informations perçues. Sur un routeur s, si les informations utilisées concernent uniquement le voisinage direct de s (liens et routeurs adjacents), alors la répartition est déterminée en fonction de critères locaux. En revanche si sur s, les proportions évoluent aussi en fonction d'informations reçues des voisins en aval, alors il s'agit d'une répartition distribuée.
Dans la suite du paragraphe, les notations $V_d(p)$ et V_d^T désignent respectivement la charge provenant du routeur p et la charge totale à destination de d. Le terme $k_{DT}^-(s,d)$ désigne l'ensemble des routeurs en amont de s ayant activé une ligne de routage sur s à destination de d.
Ces variables sont sujets aux contraintes suivantes :

$$\forall p \in I \quad \sum_{j=1}^{n} x_j^d(p) = 1 \tag{3.10}$$

$$\forall p \in I,\ \forall j \in [1,n] \quad x_j^d(p) \geq 0 \tag{3.11}$$

$$\forall j \in 1,...,n \quad \sum_{p \in I} \frac{x_j^d(p) \times V_d(p)}{V_d^T} = x_j^d \tag{3.12}$$

Les contraintes (3.8) et (3.10) impliquent la propriété de conservation des proportions globales et par interface entrante, telle que l'intégralité du trafic soit commutée. La contrainte (3.9) indique simplement que les proportions sont des réels positifs. La contrainte (3.10) indique que la somme des proportions reportées à la charge entrante par interface dépend des proportions globales.
Nous définissons la fonction $U(l)$, où l désigne un lien sortant de s, comme le taux d'utilisation du lien l (la charge totale supportée par le lien l divisée par sa capacité $c(l)$) :

$$U(l) = \sum_{p \in I, d \in N}^{\{j | NH_j(s,d) = l.y\}} \frac{x_j^d(p) \times V_d(p)}{c(l)} \tag{3.13}$$

Pour minimiser la charge maximale du réseau nous utilisons comme heuristique l'objectif global suivant :

$$\min \max_{l \in L} U(l) \tag{3.14}$$

L'objectif de ce problème d'optimisation est d'anticiper la création des congestions en diminuant la charge relative des liens les plus utilisés. L'idée intuitive est la suivante : si les liens du réseau sont faiblement chargés, alors le temps de réponse du réseau est globalement meilleur. La référence (KKDC05) formule un problème analogue de minimisation lorsque le routage est piloté par la source sur un domaine avec une commutation de type RSVP-TE.
La résolution de ce problème d'optimisation globale implique l'utilisation de la programmation linéaire. Ce type de méthode n'est pas adapté pour produire des décisions rapides lorsque le trafic évolue rapidement.
Le contrôleur de charge que nous allons décrire dans le paragraphe suivant favorise l'utilisation des chemins optimaux jusqu'à ce qu'un problème significatif déclenche la réaction du moniteur en fonction du seuil défini précédemment.

3.2.4.5 Répartition locale

Le routage doit privilégier une consommation non excessive des ressources disponibles. Lorsque le réseau est très chargé, il est souhaitable que les chemins les plus courts soient utilisés en priorité. Pour cela, notre politique de déviation de charge utilise un classement des NHs en fonction de leur coût.
Dans ce paragraphe, nous présentons l'algorithme incrémental que nous avons choisi d'implémenter pour nos simulations. Il utilise une heuristique locale dont le but est de tendre incrémentalement vers les proportions optimales correspondant à l'équation (4.12). Notre heuristique ne présente pas une complexité en calcul importante et permet de réagir rapidement aux fluctutations de charges imprévisibles.
Pour réduire la complexité des mesures, comme suggéré dans la partie 3.2.4.2, chaque routeur s se contente de mesurer la charge de chacun de ses liens l durant l'intervalle de temps t choisi :

$$D_l = U(l) \times c(l)$$

Le contrôleur de charge du routeur s réagit seulement lorsqu'un des liens sortants de s, l par exemple, est saturé en fonction d'un seuil donné : $D_l > \alpha \times c(l)$. Ce routeur doit alors déterminer son lien sortant le plus critique en fonction du taux de charge [10]. Le contrôleur de charge déplace alors l'excès de demande vers un NH alternatif à l'activité suffisamment faible. Un NH est considéré peu chargé si sa charge vérifie $U_l < \beta \times c(l)$ (critère de faible charge).
Pour un couple (destination, interface entrante) donné, (d, p), un lien l correspond à un j^e NH tel que $l.y = NH_j(s, d) \in NH(p, s, d)$ et $x_j^d \geq 0$. Le meilleur lien alternatif sortant de s et satisfaisant le critère

[10] Il est possible d'utiliser des échelles et des seuils différents en fonction de la capacité des liens. Par exemple, une échelle exponentielle permet de considérer prioritairement les liens haut débit.

de faible charge vérifie :

$$min_j(C_j(s,d) \mid D_{\{s,NH_j(s,d)\}} < \beta \times c(l))$$

Le principe incrémental de notre contrôle de charge est le relâchement itératif d'un facteur de proportion. En pratique, lorsqu'un lien sortant est à l'origine du déclenchement de la procédure, il s'agit de calculer la proportion théorique qu'il doit relâcher sur un ou plusieurs NHs alternatifs.
Après chaque période de mesure, correspondant à une itération de notre processus, réalisée à l'échelle de temps t, la nouvelle proportion correspondant au lien $l = NH_j(s,d)$ le plus chargé vérifie :

$$\forall p \in I, d \in N \quad x_j^d(p) \leftarrow x_j^d(p) \times (\frac{\alpha \times c(l)}{U_l}) \tag{3.15}$$

Si une congestion est détectée sur le lien l, le contrôleur de charge attaché à s doit déplacer, pour toute destination et pour chaque interface entrante possible, la proportion relâchée sur $x_j^d(p)$ vers un ou plusieurs NHs alternatifs.
En pratique, il se peut qu'il n'existe aucune alternative locale et peu chargée, pour un ou plusieurs couples (d,p). Dans ce cas, la proportion $x_j^d(p)$ reste identique. Par ailleurs, afin d'ajuster les proportions, il est nécessaire de prendre en compte la capacité des liens et le coût des chemins associés. Une fois la proportion de relâchement calculée, il reste à déterminer comment la distribuer entre les prochains sauts non congestionnés. Si le nombre de flux est peu élevé ou que la répartition du volume n'est pas uniforme sur les intervalles délimités par les proportions, alors les volumes effectivement déviés par le contrôleur de charge ne sont pas nécessairement satisfaisants. Pour cela, l'adaptation de notre heuristique est incrémentale.
La distribution de la proportion relâchée peut s'effectuer selon diverses indicateurs : bande passante résiduelle ($br_l = c(l) - D_l$), coût du chemin alternatif ($C_j(s,d)$), etc. La difficulté est de produire une réaction cohérente, notamment lorsque plusieurs liens sortants d'un même routeur sont surchargés. Le contrôleur de charge du routeur s peut notamment vérifier si l'ensemble de ses liens sortants alternatifs peu chargés peut supporter la charge entrante. La somme des bandes passantes résiduelles de ces liens doit vérifier :

$$D_l - \alpha \times c(l) \leq \sum_{\forall l.y \in NH(p,s,d)}^{D_{\{s,NH_j(s,d)\}} < \beta \times c(l)} br_l$$

Si c'est le cas, le contrôleur de charge peut répartir le surplus de charge sur les NHs dont la somme des bandes passantes résiduelles est capable de supporter cette charge. Si on considère que les NHs sont classés par coût croissant (en fonction du chemin qu'ils représentent), il suffit de parcourir la table des NHs jusqu'à ce que le surplus de charge soit absorbé par les meilleures alternatives possibles.
Dans le cas contraire, l'excès de charge ne peut être réparti sur l'ensemble des prochains sauts. Il devient alors nécessaire de notifier les voisins en amont. Le paragraphe 3.2.4.7 introduit quelques éléments de réponse à ce sujet.

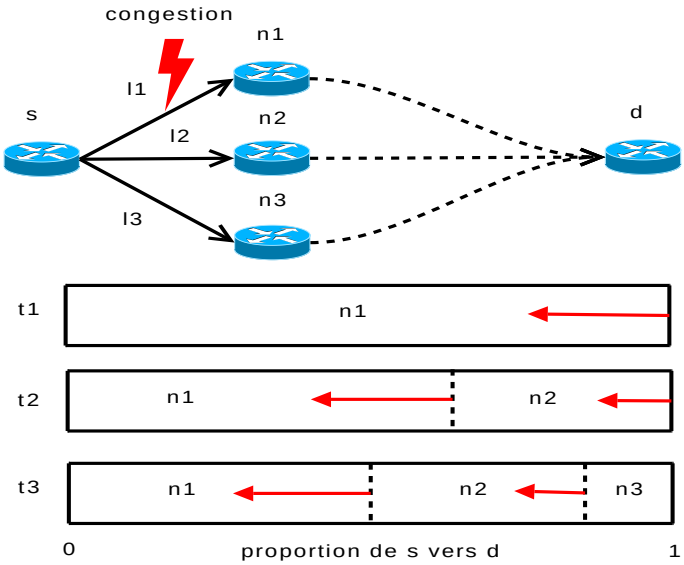


FIG. 3.5 – Proportions glissantes

3.2.4.6 Retour à un état initial

Nous considérons comme hypothèse initiale que la valuation des liens est optimisée en fonction des demandes moyennes observées. Par conséquent, dans un état nominal, le réseau est configuré pour utiliser uniquement les meilleurs chemins comme avec les techniques décrites dans (SGD05) par exemple. Dès lors que la charge d'un lien devient critique, notre contrôleur de charge intervient pour distribuer l'excédent de charge sur des prochains sauts alternatifs. Notre module de répartition est donc exclusivement réactif et se base sur des mesures de charge temps-réel.
Nous avons donc défini un mécanisme pour retourner dans un état normal lorsque la situation critique semble terminée. En pratique, lorsqu'un lien l n'est plus congestionné par une charge importante $D_l < \beta \times c(l)$, les proportions $x_1^d(p)$ (et $x_j^d(p)$ si $C_j(l.x,d) = C_1(l.x,d)$) correspondant aux lignes de routage $(p, NH_1(l.x,d), d)$ $\forall p \in pred(l.x), \forall d \in N$ retournent progressivement à 1 au détriment des autres proportions $x_j^d(p) | C_j(l.x,d) > C_1(l.x,d)$.
Le retour à un état normal peut prendre plusieurs formes. Il peut notamment être linéaire ou exponentiel. Dans nos évaluations, nous avons choisi d'utiliser un pas linéaire tel que $x_1^d(p) \leftarrow x_1^d(p) + \frac{1}{100}$.

3.2.4.7 Répartition distribuée

Dans ce paragraphe, nous décrivons succintement les pistes de recherche envisagées pour notifier les routeurs en amont de la congestion. L'objectif est de permettre à un routeur détectant une congestion de communiquer avec les routeurs générant du trafic passant par lui. Soit un routeur s et un sous-ensemble de destinations pour lesquelles il n'est pas capable de distribuer le surplus de charge constaté sur un lien sortant l. Cette incapacité peut prendre forme via deux phénomènes : soit s ne possède pas de NHs alternatifs pour ce sous-ensemble de destinations, soit le(s) NH(s) alternatifs ne

peu(ven)t supporter la charge induite par la déviation. En pratique, et pour simplifier, on désignera l'un ou l'autre de ces cas par le fait que s n'a pas suffisamment de ressources alternatives pour protéger l. Le mécanisme de restauration a pour objectif de contourner les congestions mais la panne de lien peut être considérée comme un cas particulier.
Une répartition distribuée de la charge nécessite une coopération entre routeurs voisins, et ceci récursivement, éventuellement jusqu'à la source du trafic. Un protocole de notification en amont tel que celui décrit dans la partie 3.1.7 peut s'adapter au cas des congestions. Néanmoins, à ce modèle, il faut ajouter une information de quantification pour désigner l'ordre de grandeur de la congestion. Par ailleurs, une répartition distribuée est plus sensible aux problèmes d'oscillations de charge et donc de stabilité.
La quantification de la congestion, ou plus généralement du résidu de congestion non distribuable localement, peut s'inspirer des modèles étudiés dans ce chapitre. La proportion de charge relâchée, qui n'est pas redistribuable sur des NHs alternatifs, peut servir d'indicateur relatif.
Pour réduire le nombre de messages de notification, nous envisageons d'implémenter un processus d'agrégation similaire à celui décrit dans (GZR03). La proposition avancée dans ces travaux est appropriée à un routage par interface entrante car les mesures sont collectées à cette granularité.
Une autre problématique intéressante est la définition d'une notification sélective et incrémentale : d'une part, un routeur détectant une congestion n'est pas nécessairement contraint d'alerter tous les routeurs en amont présents dans sa table de routage. D'autre part, la notification peut s'adapter en fonction du comportement des routeurs en amont. Selon la quantité de trafic émise par un routeur en amont donné, la notification peut prendre la forme d'un processus incrémental. A chaque itération, seul le voisin émettant le plus de trafic vers les destinations affectées par la panne est prévenu. Ce type de sélection permet de réduire la complexité en termes de messages mais nécessite des mesures dont la granularité est dépendante de l'interface d'entrée.
Les informations remontées en amont peuvent être distribuées selon un mode *soft state* pour définir la période de réaction : tant que la congestion persiste, la notification serait périodiquement réémise et chaque annonce serait associée à un *timer* caractérisant la gravité du problème. A l'expiration de ce timer, le routage pourrait alors tendre progressivement vers son état nominal si les annonces ne sont plus relayées.

Notre proposition de multiroutage, DT(p), bénéficie de propriétés intéressantes aussi bien pour réguler le trafic en cas de congestions anormales et persistantes que pour dévier la charge lors d'une modification topologique impromptue. Dans le chapitre suivant, à travers différentes perpectives, topologies et scénarios, nous nous efforcerons de mettre en avant la diversité de chemins générée avec DT(p) pour mettre en œuvre une couverture importante. La qualité de la couverture présente d'autres avantages qu'une protection efficace. Elle garantit, via l'activation de routes diverses, une bande passante cumulative élevée entre chaque paire de nœuds. Plus les possibilités de redirection locale sont élevées, plus l'évitement de congestion est favorisé. En effet, le nombre de points de reroutage générés par DT(p) permet d'assurer la fiabilité du réseau de manière réactive et distribuée. Cette robustesse se caractérise aussi bien par le contournement de pannes physiques que par une réactivité accrue pour faire face à

des variations de charge ne correspondant pas au trafic nominal observé via des outils de métrologie et d'inférence. Nos contributions sont d'autant plus intéressantes lorsque ces deux aspects sont liés : l'occurence d'une panne est généralement à l'origine de variations de charges rapides et importantes que le réseau doit être capable de supporter.

Chapitre 4

Outils et résultats de simulations

Ce chapitre est consacré à la présentation des outils de simulation et des résultats obtenus. Nous avons utilisé le simulateur *Network Simulator 2* (ns2, (ns2)) pour évaluer les performances de nos contributions. La première partie décrit différentes modifications apportées à ns2. Nous y avons intégré nos algorithmes de routage ainsi que plusieurs protocoles de multiroutage saut par saut et de reroutage rapide sur IP.
Afin de mesurer la diversité de chemins générés par nos propositions et de pouvoir la comparer aux méthodes existantes, nous avons utilisé nos propres outils de cartographie. La seconde partie introduit les principaux outils usuels de cartographie et présente les topologies obtenues par traces. Par ailleurs, pour analyser l'impact de la diversité des chemins pour le partage de charge, nous avons défini un modèle de trafic basé sur des mesures réelles issues du projet *Totem* (tot) et obtenues sur le réseau GEANT (gea).
La dernière partie présente les résultats de simulation. Ces résultats se déclinent en deux catégories : d'une part, les mesures déterministes évaluant la diversité des chemins activés ; d'autre part, les mesures obtenues avec notre modèle de trafic permettant d'analyser la qualité du routage multichemins proposé. Nous évaluerons la diversité des chemins sous trois angles : nombre moyen de prochains sauts activés, nombre de routes entre chaque paire de routeurs et couverture pour la protection du chemin primaire. Ces trois indicateurs permettront de mettre en relief la flexibilité de redirection générée par nos méthodes. Nous analyserons également les bénéfices induits par notre solution de multiroutage en cas de congestion sur un réseau chargé. Ces résultats souligneront l'importance de la diversité des routes pour distribuer l'excès de charge du aux congestions.

4.1 Network Simulator 2 (ns2)

Le développement de protocoles dans ns2 s'effectue dans les langages de programmation *C++* et *OTcl*. L'implémentation de nos contributions implique un certain nombre d'interactions entre ces deux couches de code. Plusieurs modifications ont été nécessaires pour intégrer nos propositions, en particulier pour :

- Le calcul et la validation des chemins,
- La mise en œuvre de la commutation multichemins proportionnelle,
- L'estampillage à la source pour les flux TCP.

Cette partie est structurée selon les modules ns2 où ces modifications ont été importantes. Par ailleurs, durant la phase d'implémentation, nous avons analysé un certain nombre de procédures liées à la mise en œuvre des agents de la couche transport. Ces aspects seront développés dans le dernier paragraphe.

4.1.1 Routage à états des liens (*Linkstate module*)

Les algorithmes de calcul de chemins sont liés au module *linkstate*. Le protocole de routage à états des liens de ns2 et son algorithme de calcul des plus courts chemins ne prend pas en compte les chemins aux coûts inégaux. De plus, ns2 ne comporte pas de processus de validation, car le principe de sous-optimalité des meilleurs chemins est suffisant pour une mettre en œuvre commutation cohérente. C'est à l'intérieur de ce module que nos algorithmes, DT et DT(p), ont été intégrés. L'algorithme de Dijkstra y est modifié pour supporter le calcul de chemins transverses. Ces chemins sont ensuite validés par une fonction spécifique (ECMP, LFI, IIC, etc.).
En pratique, le processus de validation est réalisé via l'échange de messages entre routeurs adjacents. Cette fonctionnalité permet de mettre en œuvre plusieurs tests de validition selon la règle choisie : DT(p), LFI, LFA et UTURN. DT(p) nécessite un ensemble de procédures protocolaires plus élaborées. Si la fonction de validation échoue à un saut, il est possible d'étendre la validation sur le voisinage à p sauts. Les messages de validation $Query - P$ sont transmis vers les DT-voisins non validés dans un rayon de p sauts. Chaque nœud possède une structure dynamique correspondant à Lv pour mémoriser l'état des tests. Nous avons utilisé une fonction de hachage pour projeter le chemin testé P (constitué de 1 à p sauts) sur un unique champ de la structure Lv.
Une fois validés, les prochains sauts sont enregistrés et transmis vers le module de commutation (*classifier*) qui se chargera de les activer si la propriété de couverture est garantie entre routeurs adjacents. De manière générale, quelle que soit la fonction de validation, cette propriété régit la commutation. Ces modifications font intervenir plusieurs modules de programmation depuis les mécanismes d'annonces du routage à états des liens jusqu'au commutateur. Des exemples d'intercations entre les différents modules de ns2 sont donnés dans l'annexe 4.3.3.

4.1.2 Commutateur (*Classifier module*)

C'est au niveau du *classifier* que nous avons implémenté la commutation par interface entrante. Les lignes de routage (*slots*) ont été modifiées pour supporter cette dimension supplémentaire. Dans ns2, l'aiguillage des paquets est réalisé par destination, chaque slot correspond à une destination et un NH donné. Plusieurs slots pour une destination donnée peuvent être mis en place avec l'option *Multipath* de ns2. Nous avons simplement étendu cette caractéristique aux NHs de coûts inégaux. Cependant, avec nos modifications l'aiguillage est dépendant de l'interface d'entrée, en particulier pour DT(p) et UTURN. Pour LFI et LFA, les slots sont identiques quelle que soit l'origine du paquet.
Dès lors qu'il est sollicité pour ajouter un nouvau slot v correspondant à une ligne de routage $(p, v, d)_s$, le classifier d'un routeur s vérifie que l'entrée $(s, v, d)_s$ existe. Si ce n'est pas le cas, il enregistre la requête pour éventuellement l'activer plus tard à condition que la ligne $(s, v, d)_s$ soit ajoutée ultérieurement. A chaque réception d'un LSA, chaque nœud réinitialise le commutateur en supprimant l'ensemble de du ses entrées.
L'algorithme de partage de charge est aussi intégré au module classifier. Dans ns2, seul le partage de charge équitable avec une option de type *round robin* est implémenté. Nous avons intégré deux

mécanismes supplémentaires : les slots sont ordonnés par le coût des chemins/NHs qu'ils représentent et ils sont alors associés à une proportion. Cette proportion est relative à un couple (destination, interface d'entrée), noté (d, p). Techniquement, une proportion est caractérisée par un intervalle dont la taille représente la fraction des flux à destination de d et provenant de p qui seront transmis via le slot v correspondant à la ligne de routage $(p, v, d)_s$. Ces intervalles dynamiques dépendent des informations reportées par le module de *monitoring* de ns2.
Initialement, seul le meilleur NH est utilisé. Dans nos simulations, la borne inférieure associée au slot $NH_1(s, d)$ est 1, et la borne supérieure est N. Tous les autres slots $NH_j(s, d) \mid j > 1$ ont leurs deux bornes égales à N. En fonction des notifications émise par le module de *monitoring*, ces bornes évoluent selon l'algorithme choisi. Le paragraphe suivant décrit comment les flux sont dispersés sur ces intervalles. Les paquets de signalisation utilisés pour les LSA et pour les messages de validation ne sont pas soumis aux changements de proportions, ils sont systématiquement acheminés sur le slot correspondant au meilleur NH. En cas de panne sur le NH primaire, l'ensemble de la signalisation est relayé sur la même ligne de routage alternative si il en existe une.

4.1.3 Flux et couche transport

Dans ns2, les flux sont générés avec le module *agent*. Un agent *source* émet des paquets vers un agent *puits* qui les réceptionne, et dans le cas de TCP, les acquitte. Les agents sont attachés à des nœuds. On peut considérer qu'un couple d'agents (source, puits) est assimilé à un flux. Pour simuler le routage proportionnel à la granularité du flux, la structure des paquets émis par un agent a été modifiée. Chaque paquet est estampillé par son agent émetteur. Cette estampille, ou *tag*, est la même pour tous les paquets appartenant à un même couple d'agents. Ainsi, le classifier d'un nœud traversé par ce flux, pourra commuter tous les paquets appartenant à un flux donné sur le même slot. Pour une destination d donnée, le choix du slot est déterminé par l'appartenance du tag à l'intervalle associé à ce slot. Si les proportions sont stables pendant la durée d'acheminement d'une rafale d'un flux, chacun des paquets appartenant à cette rafale sera acheminé sur une route identique. Cette propriété permet de garantir la cohérence de distribution des paquets appartenant aux rafales TCP et éventuellement de mettre en œuvre du routage QoS.
Cependant, dans nos simulations, l'agent émetteur ne prend pas en compte l'aspect QoS : les estampilles sont attribuées aléatoirement aux agents. Par ailleurs, les commutateurs considérent un ordonnancement global des tags. En d'autres termes, le tag attribué à un paquet conditionne son acheminement de bout en bout car les slots et les proportions qui leurs sont associées sont ordonnés selon le coût du chemin auquel il correspondent. En pratique, l'espace d'estampillage est déterminé par l'intervalle $[1, N]$. Ainsi, un paquet dont le tag est proche de N aura une probabilité de *déviation* plus importante qu'un paquet dont le tag est proche de 1. Le terme *déviation* signifit ici que le paquet est aiguillé vers un chemin non optimal. Le tag choisi peut influer sur le coût de la route correspondant aux NHs choisis de proche en proche pour la commutation. Si l'ordonnancement des proportions choisies par les classifiers n'est pas global, c'est-à-dire si l'association entre les intervalles et les prochains sauts n'est pas uniforme, alors cette propriété n'est plus satisfaite et la commutation n'est pas uniforme de bout en bout par rapport

à l'estampille positionnée par la source.
Pour accélerer la durée des simulations, nous avons utilisé le correctif proposé par Wei et Cao (pat). Ce correctif permet d'analyser par simulation le comportement des implémentations linux pour TCP. D'une part, il offre une variété de modélisation des mécanismes liés à TCP extrêmement intéressante pour le contrôle de congestion (*cubic* (RX05), *highspeed* (Flo03), *reno* (FHG04), etc.). D'autre part, il facilite l'intégration et donc l'évaluation de nouveaux modèles de contrôle de congestion. Pour finir, en corrigeant une erreur présente dans l'implémention de TCP, il permet de réduire considérablement la durée des simulations où le nombre de flux TCP est important. En ce qui nous concerne, ce correctif a permis de diviser par 4 le temps de nos simulations pour un temps simulé équivalent. En pratique, la durée d'une simulation d'un quart d'heure de temps simulé prend, en moyenne avec ce correctif, 7h. Nous avons utilisé des agents sources *reno* et des puits de type *Sack* pour la simulation des flux.

4.2 Cartographie et modèles de trafic

4.2.1 Cartographie

Ce paragraphe présente les topologies obtenues par cartographie.

4.2.1.1 Topologie aléatoire et réaliste

Pour générer aléatoirement des topologies aux caractéristiques paramétrables, des outils tel que *BRITE* (MLMB01) ou *GT-IM* (Zeg96) sont souvent utilisés dans les travaux de recherche scientifiques. Cependant, rien ne garantit que les propriétés des topologies ainsi générées soient réalistes. Les travaux présentés dans (MKFV06) suggèrent d'utiliser une approche plus élaborée : à partir d'un segment restreint de topologie, une partie d'un graphe réel, les algorithmes proposés permettent de générer des graphes plus grands aux caractéristiques similaires.
Pour permettre aux acteurs de la recherche de comparer leurs résultats, en particulier pour les protocoles de routage, un consensus implicite s'est fait sur les topologies obtenues avec l'outil *Rocketfuel* (SMW02). Cet outil permet de collecter par inférence le graphe de l'ISP sondé. La plupart des méthodes similaires utilise l'outil *traceroute*. On peut également citer le projet *Skitter* (ski) qui est également l'un des outils les plus répandus pour sonder les topologies de l'Internet. Néanmoins les auteurs de (TMSV03) ont démontré que les topologies obtenues avec des sondes comme Rocketfuel présentent une diversité de chemins supérieure à la réalité. Certains liens point à point sont virtuels et peuvent créer un biais non négligeable dans l'évaluation topologique. Par exemple, l'interconnexion *full mesh* de routeurs par un switch de niveau 2 peut générer une confusion dans l'analyse des liaisons point à point. D'autres travaux, comme (MFM+06), analysent des modèles d'interconnexions d'AS pour permettre d'évaluer le routage interdomaine.

4.2.1.2 Nos modèles avec mrinfo

Nous avons choisi d'utiliser les topologies obtenues avec nos propres sondes. Ces travaux ont été initiés au sein notre équipe depuis l'étude de Pansiot et Grad (PG98) sur les routes et les arbres multicast dans l'Internet. Cette approche a ensuite été étendue en sondant, avec des techniques de type *traceroute*, la partie IPv6 de l'Internet (MH05).
Plus récemment, J-J. Pansiot a initié un projet analysant la dynamicité des réseaux multicast (mri). Ce projet est basé sur l'outil *mrinfo*. Pour les réseaux natifs multicast ne filtrant pas les requêtes *mrinfo*, cet instrument permet d'obtenir une carte précise des interconnexions entres routeurs (Pan07). Nous avons ainsi obtenu quatre topologies aux caractéristiques variées sur les réseaux Renater, OpenTransit, Global Crossing et Alternet. Le lien (oim) fournit l'ensemble des cartes correspondantes.
Grâce à mrinfo, il est possible de collecter un ensemble de caractéristiques topologiques sous la forme d'une correspondance entre interfaces IP. La base de donnée recueillie est structurée telle que chaque élément de cette base contient deux informations :

- l'adresse IP par laquelle le routeur r a répondu à la requête,
- une liste L_r de ses interfaces sortantes associée à la liste des interfaces connectées.

Une entrée de la liste de correspondance L_r est notée : $L_r(i, j)$. Les indices i et j désignent respectivement l'adresse IP d'une interface sortante de r et l'adresse IP d'une interface entrante d'un routeur v ($i \in r, j \in v$). Typiquement, une entrée prend cette forme (il s'agit d'un routeur situé à Grenoble et appartenant au réseau Renater) :
r : 193.51.179.238 (grenoble-pos1-0.cssi.renater.fr) [version 12.0] : 8
L_r : 193.51.179.238 -> 193.51.179.237 (lyon-pos13-0.cssi.renater.fr) [1/0/pim]
193.51.184.118 -> 193.51.184.117 (amplivia04-grenoble.cssi.renater.fr) [1/0/pim/querier]
193.51.181.94 -> 193.51.181.93 (prim-tigre-grenoble.cssi.renater.fr) [1/0/pim/querier]
193.51.180.33 -> 193.51.180.34 (nice-pos2-0.cssi.renater.fr) [1/0/pim]

Pour translater nos topologies sous la forme de scripts Tcl, nous avons considéré que deux routeurs r et v sont connectés par les interfaces i et j si et seulement si la liaison est symétrique c'est-à-dire : $L_r(i, j) \in L_r \wedge L_v(j, i) \in L_v$. Par ailleurs, nous considérons seulement la partition connexe la plus grande en nombre de routeurs.
Le tableau 4.1 fournit un récapitulatif de leurs principales caractéristiques ainsi que celles de GEANT (voir 4.2.2, les nœuds marqués d'un tiret ne figure pas dans la topologie considérée par le projet *Totem*). La figure 4.2, donne une représentation visuelle de ces quatre réseaux. Pour cela, nous avons utilisé le logiciel de visualisation et d'analyse *Pajek* (paj).

Sur l'ensemble de ces quatre topologies, nous avons considéré la valuation des liens comme uniforme. Ces quatre réseaux présentent des caractéristiques différentes en termes de maillage et de connectivité. La figure 4.1 représente la distribution des degrés de l'ensemble des routeurs appartenant aux différents réseaux. Pour les réseaux Renater et OpenTransit, il semble que la distribution des degrés des routeurs approche une loi de puissance alors que pour les deux autres réseaux, la distribution est plus uniforme.

Nom du réseau	# de nœuds	# de liens	Diamètre	Degré moyen
Alternet	83	334	8	4
Global Crossing	97	370	9	3.8
OpenTransit	74	204	11	2.8
Renater	78	198	9	2.5
GEANT	23	74	5	3.2

Tab. 4.1 – Réseaux d'évaluation

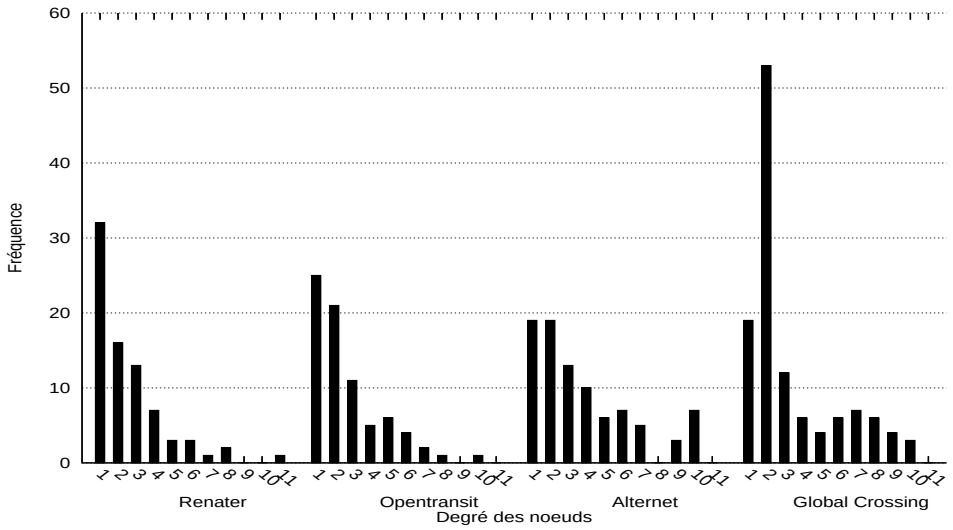


Fig. 4.1 – Distribution des degrés

Ces topologies seront utilisées dans l'évaluation statique permettant de comparer la diversité des chemins générée par différents processus de validation. Les résultats obtenus sur ces quatre réseaux sont déterministes. Aucun trafic de données n'a été simulé dans la première partie de notre évaluation. L'objectif de cette première partie de notre analyse est d'évaluer le nombre de routes et la diversité des prochains sauts activés.

Sur l'ensemble de ces réseaux d'évaluation, nous ne disposions pas d'informations réalistes concernant l'échange de trafic point à point. Pour cela, nous avons utilisé une topologie pour laquelle des mesures précises ont été effectuées via le projet *Totem* : GEANT.

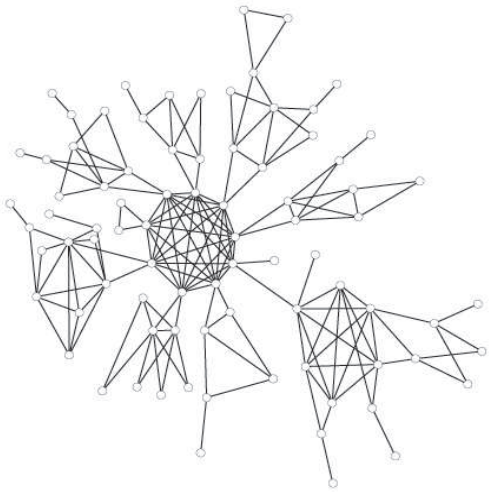
4.2.2 Trafic et flux TCP

Les outils que nous avons utilisés pour simuler l'échange de trafic sur le réseau GEANT sont décrits dans ce paragraphe. Cette topologie, largemement utilisée dans la littérature scientifique, a été l'objet de mesures diverses notammment pour y analyser la dynamicité des flux intra et interdomaine.

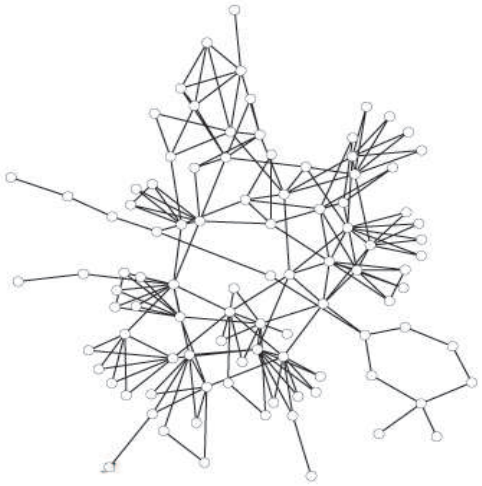
Les liens de GEANT sont valués de manière à optimiser le routage : ces poids correspondent aux données collectées dans les matrices Totem (voir figure 4.3). De cette manière, le routage et la répartition des volumes recueillis avec l'outil Totem sont influencés par cette valuation spécifique : il s'agit d'un routage intégrant de l'ingénierie de trafic pour s'adapter aux charges typiques.
La modélisation du trafic nous a permis d'évaluer l'impact de la diversité de commutation générée par nos contributions dans un contexte dynamique. Notre générateur de flux utilise les ensembles de données fournis via l'outil Totem (UQBL06). Cette base de traces réelles décrit l'activité du réseau sous formes de matrices de trafic. Chaque matrice représente les données collectées sur une période de temps d'un quart d'heure. Une entrée donnée correspond au volume de trafic échangé entre deux routeurs. Ces mesures ont été réalisées avec les informations de routage IGP, des données *Netflow* échantillonées et les informations de routage interdomaine de type BGP. Notre générateur de flux a pour objectif de reproduire, avec la granularité de flux la plus fine possible, les activités de trafic collectées dans ces matrices. En pratique, cette granularité est néanmoins fortement limitée par les contraintes de simulation : une simulation ns2 nécessite un temps de calcul très important lorsque la modélisation du trafic intègre de très nombreux flux TCP. Le paragraphe 4.2.2.1 décrit notre proposition pour la génération de trafic afin d'obtenir un compromis entre simulation réaliste et temps de calcul raisonnable.

4.2.2.1 Modèlisation du trafic

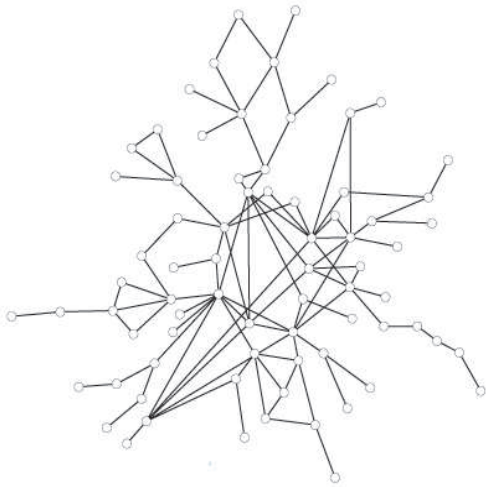
Alors qu'un grand nombre de flux de l'Internet sont de durée très courte, l'écrasante majorité des paquets et des données appartiennent aux flux volumineux dont la durée de vie est relativement longue. L'analyse réalisée expérimentalement dans (LH03) sur des opérateurs de l'épine dorsale d'Internet propose une décomposition *zoologique* des flux. Les auteurs subdivisent notamment le trafic Internet en deux classes de trafic : les *éléphants*, représentant les flux volumineux à longue durée de vie, et les



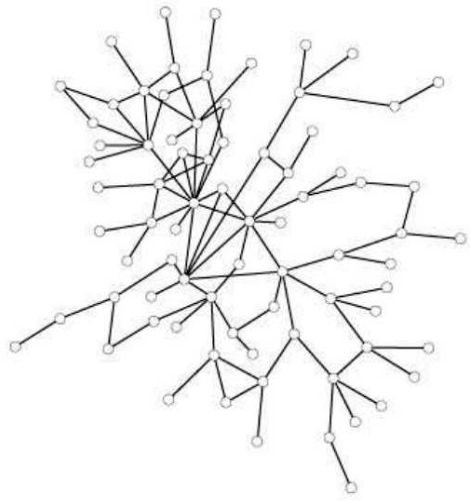
Alternet



Global Crossing



OpenTransit



Renater

FIG. 4.2 – Topologies obtenues avec mrinfo

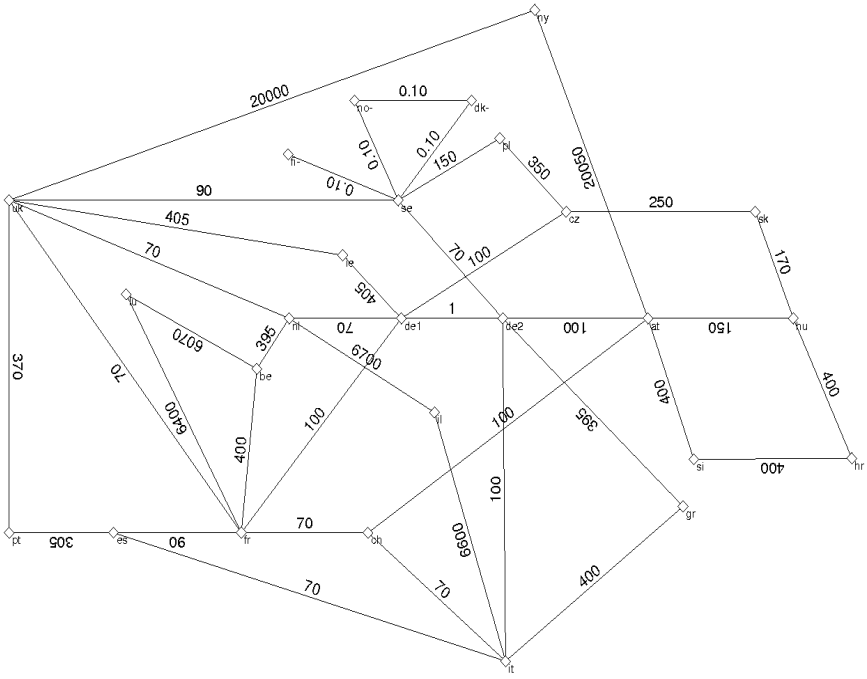


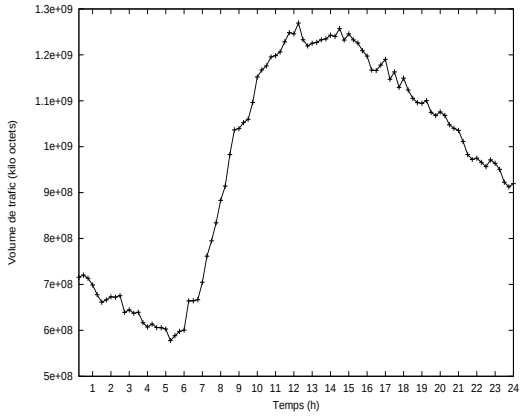
FIG. 4.3 – Le graphe valué de GEANT

souris, micro-flux éphémères (pour plus de détails, voir (GM01) et (PTB+02)).
De manière générale, un très grand nombre de flux, environ 80% de l'ensemble des flux, sont inférieurs en volume à une centaine de kilo-octets. Néanmoins la somme de ces micro-flux représente moins de 20% du volume total.
Pour modéliser cet ensemble de flux *bruit de fond*, les *souris*, il est impossible en pratique (dans ns2), de simuler des millions de micro-flux (TCP ou UDP). Le simulateur ns2 est extrêmement gourmand en ressources pour générér un modèle de flux avec une granularité *par flux*. Nous avons donc considéré les outils de modélisation présents dans ns2 pour la simulation des micro-flux.
Nous avons notamment envisagé d'utiliser le modèle de trafic *ON/OFF* inclus dans ns2 pour simuler le trafic *bruit de fond* sur le réseau GEANT. Le choix de la distribution des durées des périodes actives (*on*) et passives (*off*) et du débit durant les périodes d'activité caractérise ce modèle. Néanmoins dans ns2, les périodes *on* et *off* doivent être supérieures à la milliseconde et ne permettent pas de simuler une granularité temporelle très fine.
Heureusement, le correctif TCP-linux nous a permis d'étendre notre approche "tout-TCP" aux flux relativement petits. Notre modèle de trafic est orienté "tout-TCP" car ce protocole de transport représente près de 90% du trafic global de l'Internet. Les volumes des flux TCP, composant l'ensemble du trafic simulé, sont distribués selon un modèle à *queue lourde* (*heavy tailed distribution*). Le volume total obtenu par couple (source,destination) vérifie, pour un quart d'heure donné, la quantité de trafic donnée dans la matrice Totem correspondante.
Afin de reproduire le trafic de manière réaliste, un générateur de nombres aléatoires est utilisé pour tirer au sort une valeur $m \in]0, 1]$, ensuite, la transformation de *Pareto* $m \rightarrow \frac{x_{min}}{m^k}$ permet de déterminer la taille d'un flux. La fréquence de volume par flux ainsi obtenue suit une loi de puissance. Le terme x_{min} rerésente la taille de flux minimale et k est le paramètre de courbure de Pareto (*Pareto shape*). Pour obtenir un compromis faisable entre génération intensive de flux et temps de simulation acceptable, le paramètre k est fixé à 2. Le terme x_{min} est égal à $20kB$ et nous avons borné la taille maximale d'un flux à $10GB$. Pour cela, si la fonction de transformation génère un flux de taille supérieure, alors l'excédent de volume est retiré et réinjecté à l'ensemble du volume à écouler.
Dans nos simulations, l'influence de la couche applicative est négligée, chaque entrée de la matrice est décomposée en flux TCP associés à des agents reno. Chacun de ces flux génère un trafic élastique dont l'instant de départ est choisi aléatoirement dans l'espace de temps simulé $[1, 900]$ (en secondes). Chaque script de simulation génère au moins une centaine de milliers de flux. Près de 80% de la charge globale est induite par les 20% de flux les plus gros en considérant qu'un flux *éléphant* contient un volume de donnée supérieur à $500kB$. Chaque flux est constitué d'un ensemble de paquets de $1kB$. L'impact des micro-flux n'est pas considéré dans nos simulations, c'est-à-dire les *souris* dont le volume est inférieur à $20kB$.
En pratique, nos scripts utilisent des files d'attente *drop-tail* capables de contenir 75 paquets de $1kB$. La fenêtre d'émission des flux TCP est bornée par une taille maximale de 65 paquets. L'ensemble de ces paramètres a été choisi avec précaution pour que la charge globale se stabilise autour d'une moyenne correspondant à la somme des entrées de la matrice divisée par le temps simulé. La figure 4.4(a) illustre

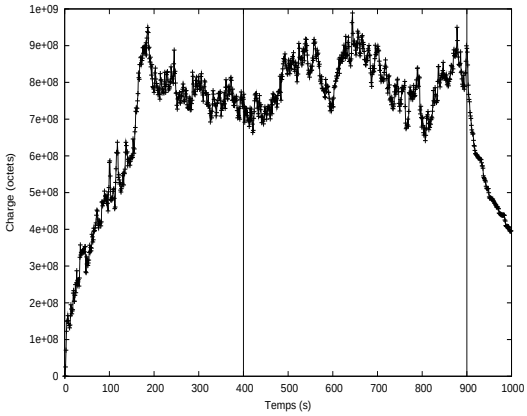
l'évolution de la charge générée lors d'une simulation.

La figure 4.4(b) représente la charge globale sur un quart d'heure simulé. Le trafic est relativement similaire d'un jour à l'autre, nos simulations reproduisent le trafic des quarts d'heures les plus chargés, c'est-à-dire entre midi et quartoze heures. Pour chaque simulation, nous avons retiré la période de *warm up* correspondant à la stabilisation progressive de la charge et uniquement considéré les données collectées durant la période stable (*steady state*). La figure 4.4(b) donne les bornes utilisées pour définir un tel état (entre 400 et 900 secondes). Cette figure correspond à un intervalle de trafic collecté dans la matinée, il s'agit du volume globale de trafic acheminé par seconde.
Les volumes de trafic mesurés ne sont pas suffisants pour *stresser* GEANT qui est a priori surdimensionné comme beaucoup de réseaux d'opérateurs. Le nombre de pertes et l'utilisation des liens est très faible, moins de 10% en moyenne. Nous avons donc décidé d'augmenter certaines entrées dans les matrices utilisées pour déclencher des congestions significatives. Les résultats ont été obtenus via un ensemble de scénarios possibles. Nous avons choisi d'augmenter la charge sur les liens dont l'utilisation est déjà relativement élevée. En pratique, il s'agit des liens de capacité Gigabit dont la charge mesurée à l'échelle de la seconde dépasse 10% de sa capacité sur des périodes de temps supérieures à 100 secondes.
Les figures 4.5(a) et (b) montrent un exemple d'augmentation de la charge sur le lien $de1 \rightarrow de2$ de GEANT (voir figure 4.3). Nous avons défini trois modèles d'augmentation artificielle de la charge : $1 \rightarrow n$, $n \rightarrow 1$ et $1 \rightarrow 1$. Le premier modèle correspond par exemple à la diffusion d'une vidéo depuis un site de partage vers n récepteurs, alors que le second modèle peut représenter l'envoi simultané de n fichiers lors d'une *deadline* sur un serveur de publications. Le dernier cas correspond simplement à un tranfert ponctuel de données volumineuses. Les deux premiers modèles multiplient par un facteur de 5 une ligne ou une colonne de la matrice de trafic. Le dernier modèle augmente par un scalaire configurable une seule entrée de la matrice.
Dans les trois cas, nous avons choisi de générer une seule congestion persistante sur un lien donné avec un routage SPF.

Sur les figures 4.5, le routage est monochemin et il s'agit d'un modèle de congestion $1 \rightarrow n$ vers le routeur $de1$. On observe, sur la figure 4.5(d), que le nombre de pertes est relativement important bien que le lien ne semble chargé qu'à 50% sur l'échelle de temps choisie. Les figures 4.6 illustrent les mesures que nous avons prélevées sur les files d'attente à la granularité des évènements simulés : arrivées et départs des paquets. L'état des files est reporté dès lors que plus de 10 paquets sont présents dans la queue. Sur la figure 4.6(a), l'échelle de temps est de l'ordre de la seconde, et on constate que les files de 75 paquets sont soumises à de fréquentes rafales de paquets. La figure 4.6(b) donne une idée de la vitesse à laquelle une file peut se remplir lorsque le trafic est de type TCP. Moins d'une milliseconde suffit à provoquer une micro-congestion sur une file de petite taille.

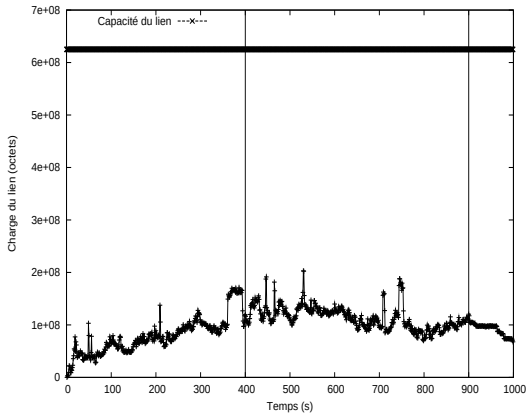


(a) Trafic mesuré sur une journée

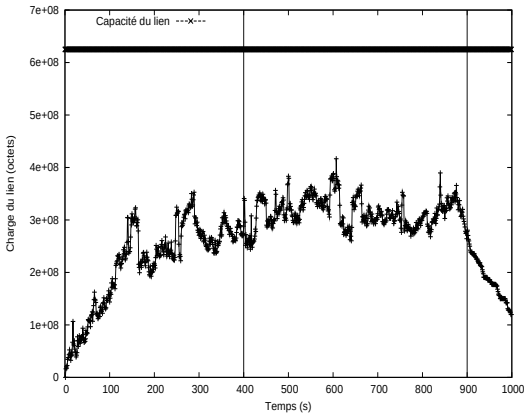


(b) Trafic simulé sur un quart d'heure

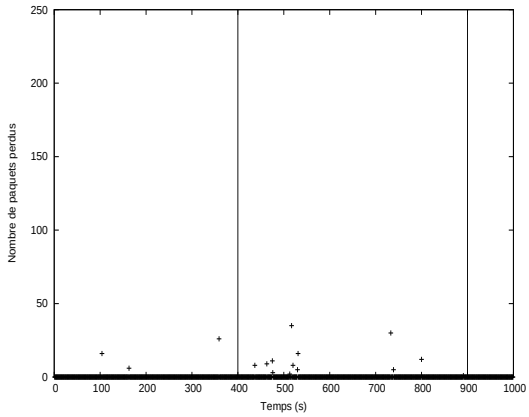
FIG. 4.4 – Etat stable et charge globale



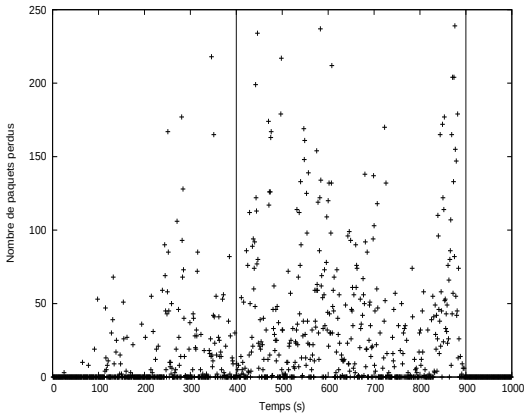
(a) Charge initiale



(b) Charge augmentée

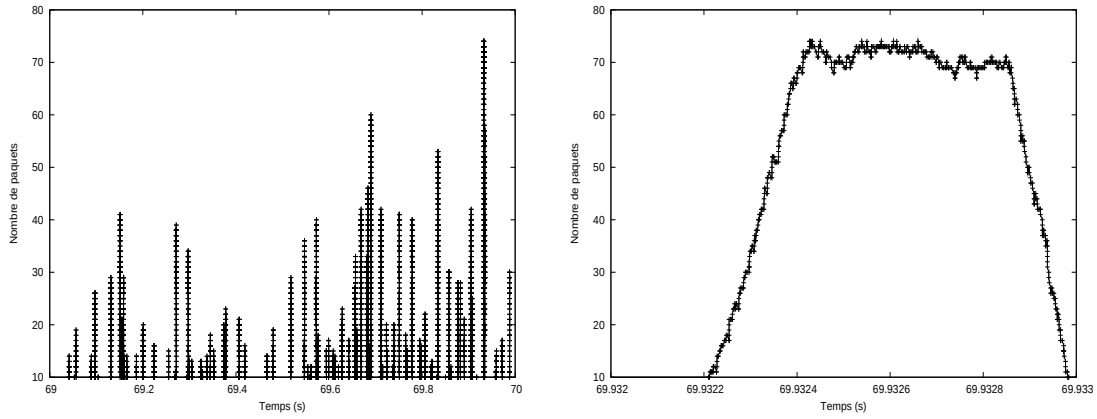


(c) Nombre de pertes initial



(d) Nombre de pertes avec charge augmentée

FIG. 4.5 – Génération de congestion avec un facteur de 5 sur le lien $\{de1, de2\}$



(a) Sur une échelle de temps de 1 s (b) Sur une échelle de temps de 1 ms

FIG. 4.6 – Etat d'une file d'attente

Ce type de micro-congestion est difficilement évitable au niveau de la couche routage, il est préférable d'agir au niveau des stratégies de files avec des mécanismes adaptés (PRIO, SFQ, RED, CBQ, etc). Nous avons choisi d'utiliser une échelle de temps détectant les périodes de congestions durables, caractérisées par une charge moyenne importante sur une ou plusieurs secondes.

4.3 Résultats de simulations

4.3.1 Diversité des chemins

Les résultats présentés dans ce paragraphe mettent en avant la diversité des chemins générés via nos procédures selon plusieurs perspectives. La première analyse (paragraphe 4.3.1.1), calcule le nombre moyen de lignes de routage activées. Pour toutes paires *(interface d'entrée,destination)*, les moyennes sont rassemblées en fonction du degré des routeurs en considérant seulement le trafic en transit. Nous nous sommes focalisés sur le nombre d'entrées dans la table de routage pour le trafic en transit dans le cas de DT(p) afin d'obtenir une comparaison équitable pour LFI.
La seconde étude évalue le nombre de routes activées entre chaque paire de routeurs. L'ensemble des routes alternatives sont regroupées en fonction de leurs longueurs.
Pour finir, nous avons aussi analysé la couverture générée par différentes formes de routage en termes de protection.

4.3.1.1 Prochains sauts validés

Dans cette première analyse de la diversité des chemins, nous avons considéré, pour chaque paire (source,destination), le nombre moyen de prochain sauts activés en fonction du degré des routeurs. Cette évaluation prend en compte une granularité de commutation représentant la moyenne du nombre de prochains sauts pour le trafic en transit. Une telle granularité n'est pas utilisée par la règle LFI alors que DT(p) active moins de NHs pour le trafic en transit que pour le trafic local. Ainsi la comparaison

est à l'avantage de LFI.
La figure 4.7 met en évidence un cas particulier où les routeurs A et D (de degré 3) ne peuvent être utilisés pour le trafic en transit avec la règle LFI. Par exemple, si les routeurs B et C font offices de passerelle vers le domaine de routage, alors aucun d'entre eux ne peut se servir de A ou de B pour rediriger son trafic. Avec une valuation uniforme, aucun routeur ne peut commuter des paquets via A ou D excepté si la destination est l'un de ces routeurs. Ainsi, les liens entrant vers A et D sont sous utilisés.
La condition LFI est restrictive pour la diversité : certains routeurs ne bénéficient d'aucun prochain saut alternatif quelle que soit la destination alors que leur degré est supérieur à 2. Notre procédure de validation DT(p) permet, même à profondeur 1, de profiter systématiquement du degré sortant pour au moins un sous-ensemble de destinations.
Les figures données en 4.8 mettent en avant la supériorité de notre procédure de validation lorsque la valuation des liens est uniforme (les histogrammes sont superposés). En effet, la condition IIC permet de valider nettement plus de NHs, même pour le trafic en transit, que la condition LFI. On notera que LFI et ECMP produisent des résultats identiques lorsque la valuation est uniforme car les deux règles sont alors équivalentes. Pour cette première série d'études préliminaires, les délais de propagation ne sont pas pris en compte dans le choix de la métrique, plus précisément, la valuation des liens est considérée égale sur l'ensemble du réseau.
Nous utiliserons la notation (p, d) pour désigner un couple *(interface d'entrée,destination)*. Sur les figures données en 4.8, l'ordonnée représente la moyenne du nombre total de lignes de routage activées pour l'ensemble des couples (p, d) relatif à un routeur s dont le degré est donné en abscisse ($s \neq p$). Cette moyenne peut, sur un routeur s donné, atteindre au maximum : $k^-(s) \times k^+(s) \times |N|$.

Sur le réseau Renater, on observe que DT(1) permet aux routeurs, dont le degré est égal à 8, de disposer d'en moyenne 1200 lignes de routage au total. Cela signifie qu'un routeur de degré 8 a en moyenne $\frac{1200/8}{77} \approx 2$ lignes de routage activées pour chaque couple (p, d). Un tel calcul prend en compte les couples (p, d) pour lesquels il n'existe aucune ligne de routage activée, ce qui signifie que lorsqu'au moins une ligne de routage est activée, le nombre de prochain sauts est en réalité plus élevé. Avec LFI, on constate qu'en moyenne moins d'un prochain saut est activé par couple (p, d).
Sur les quatre réseaux, le nombre de lignes de routage activées est grossièrement proportionnel au degré de chaque routeur. Nous remarquons aussi que le nombre de lignes de routage activées augmente plus

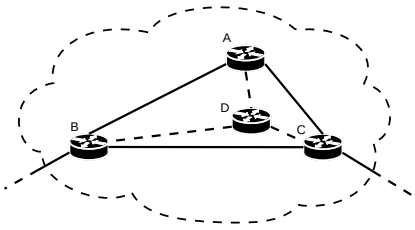


FIG. 4.7 – Configuration pyramidale

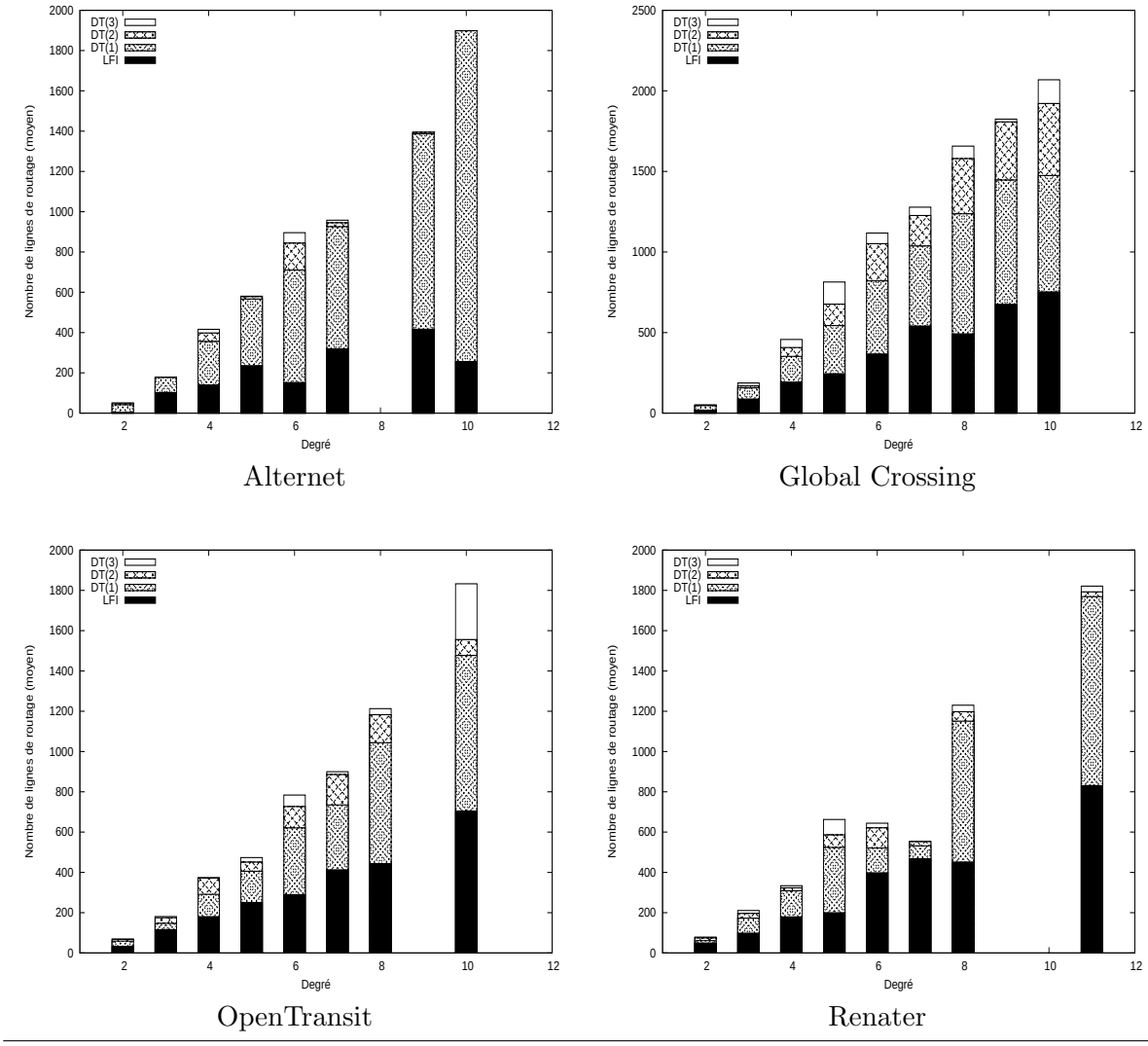


FIG. 4.8 – Lignes de routage activées selon le degré des routeurs

rapidemment avec DT(p) qu'avec LFI en fonction du degré. Cette observation semble suggérer que la condition IIC profite plus du degré des routeurs pour générer une diversité des routes élevée qu'une règle comme LFI.
La procédure DT(1) génére déjà un gain substantiel par rapport à LFI. Sur les réseaux Renater et OpenTransit, les bénéfices semblent d'autant plus tangibles que ces topologies sont relativement peu maillées (en moyenne, moins de 3 liens orientés par routeur).
Sur les topologies plus grandes et plus maillées comme Alternet et Global Crossing (en moyenne, plus de 3 liens orientés par routeur), le bénéfice de la procédure de validation en profondeur pour $p > 1$ est plus marqué, particulièrement pour les routeurs de degré élevé.
Néanmoins, de manière générale, la complexité induite par DT(p), lorsque $p > 1$, ne semble pas, sous cet angle de vue, apporter des avantages considérables en termes de diversité (spécialement pour Renater et Alternet). La seconde partie de l'analyse, portant sur le nombre de routes et la couverture générée, mettra en avant l'importance du processus de validation en profondeur pour $p > 1$.
A ce niveau de l'analyse, on peut conclure que DT(1) augmente de manière significative le nombre de lignes de routage activées par rapport à une condition comme LFI. Cette amélioration permet d'étendre les capacités de redirection en cas de panne ou de congestions. De la même manière, le flot maximal entre chaque paire de nœuds dépend en partie du nombre de redirections possibles de proche en proche, si bien que le débit potentiel maximum de chaque couple est favorisé par nos procédures. DT(p) produit des résultats systématiquement supérieurs à ceux obtenus via ECMP ou LFI.
Le nombre de routes est un indicateur plus pertinent pour évaluer la diversité de redirection engendrée par le multiroutage. Bien que cet indicateur dépend du nombre de NHs activés, il reflète mieux la qualité d'adaptation du routage sous-jacent en cas de problème. En effet, c'est la composition de proche en proche des NHs qui permet au routage d'être flexible dans la commutation. Par exemple sur une topologie comme Renater, DT(3) permet la validation de 17 routes entre Strasbourg et Pau, dont le nombre de sauts varie entre 6 et 11, alors que Strasbourg ne dispose que de deux NHs actifs vers cette destination pour son trafic local.

4.3.1.2 Distribution des routes

Dans cette partie, nous limiterons notre analyse aux réseaux Alternet et OpenTransit. L'objectif est d'évaluer la distribution des routes en fonction de leur longueur en nombre de sauts. La figure 4.9 représente le nombre de routes dépourvues de boucles selon la méthode employée (les histogrammes sont superposés). Sur cette figure, nous avons considéré une implémentation simplifiée d'ECMP qui ne prend en compte que les routes de coût optimal disjointes entre elles. Cette nuance met en avant la différence entre le nombres de routes totalement et partiellement disjointes. En effet, la condition LFI, dans la mesure où la valuation est uniforme, active uniquement les routes alternatives dont le coût est optimal. Celles-ci peuvent être totalement ou partiellement disjointes de la route primaire. Les histogrammes de gauche concernent OpenTransit, tandis que ceux de droite sont relatifs à la distribution des routes sur Alternet. Rapporté à l'ensemble des routes primaires, la proportion de routes alternatives est très faible. Nous pouvons observer que le nombre de routes alternatives activé avec DT(p) est nettement

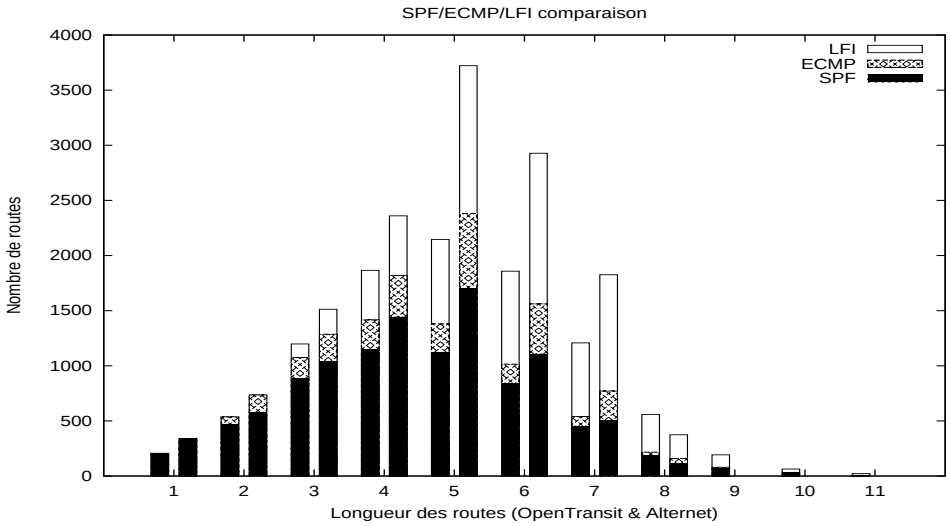


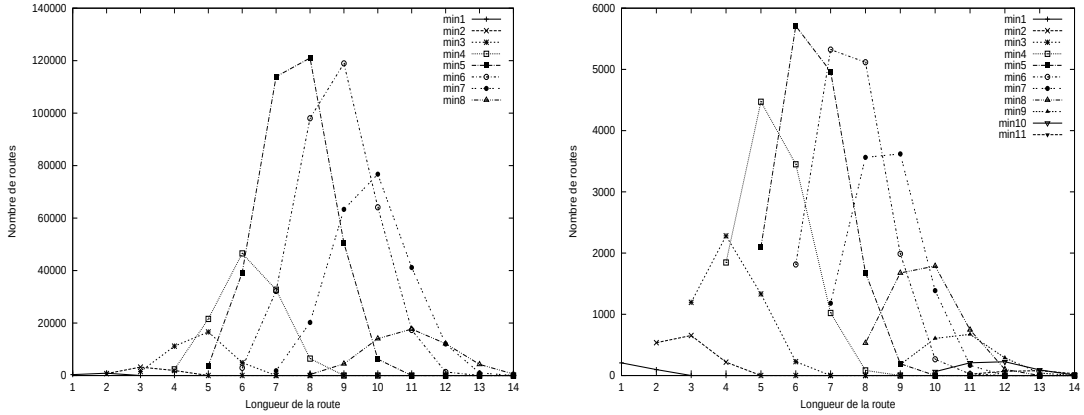
FIG. 4.9 – Nombre de meilleurs routes

supérieur à celui obtenu avec la condition LFI. Dans la mesure où la valuation est uniforme sur nos topologies d'évaluation, la longueur $l(s,d)$ d'une route alternative, liant deux routeurs s et d, vérifie :

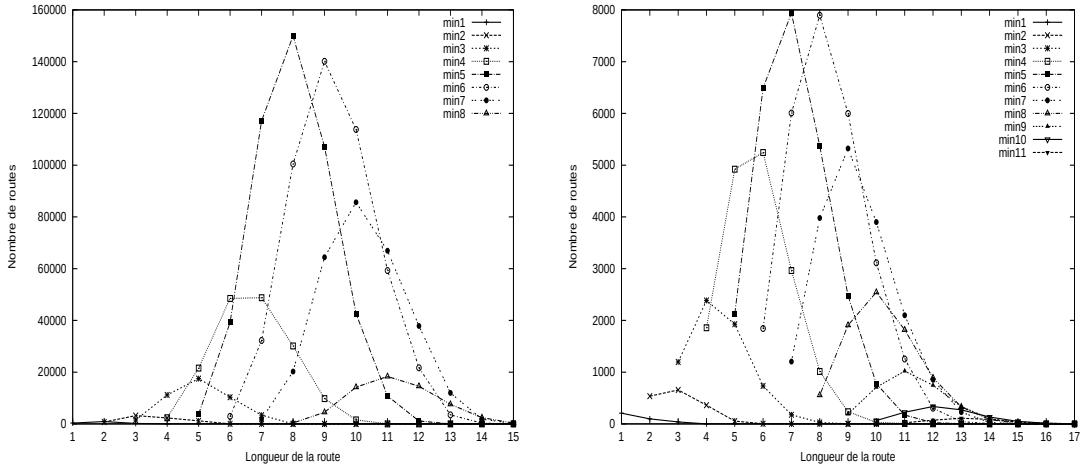
$$l(s,d) \leq C_1(s,d) \times (p+1) \tag{4.1}$$

Les routes alternatives sont regroupées sur une courbe donnée en fonction de la distance minimale de la route primaire de longueur $C_1(s,d)$ reliant un même couple de routeurs. Le premier point d'une courbe $min = 1$ indique le nombre de liens du réseau et le premier point de la dernière courbe, le $max(min)$, correspond au nombre de routes dont la longueur est égale au diamètre du réseau.
La figure 4.9 peut servir de base à l'interprétation des figures données en 4.10 et 4.11. Un histogramme de la figure 4.9, représentant un ensemble de routes de longueur donnée calculé via LFI, correspond au premier point d'une courbe noté $min = l$ sur les figures 4.10 et 4.11 car DT(p) active lui aussi l'intégralité des meilleurs routes. Ces deux figures proposent une décomposition de la distribution du nombre de routes alternatives en fonction de leur longueur et de la profondeur de validation employée. Le processus de validation DT+(p) désigne l'amélioration proposée dans le paragraphe 2.2.4 : l'algorithme DT est modifié de manière à réduire l'espace des DT-voisins aux NHs de meilleurs coûts et, au plus, une alternative supplémentaire.

OpenTransit est un réseau faiblement maillé : en moyenne, à peine plus de $2,75$ liens orientés par routeur. Le nombre de routes activées via notre processus de validation y est donc moins élevé. Cependant, cette caractéristique provient essentiellement de la longueur moyenne des cycles dans le réseau.

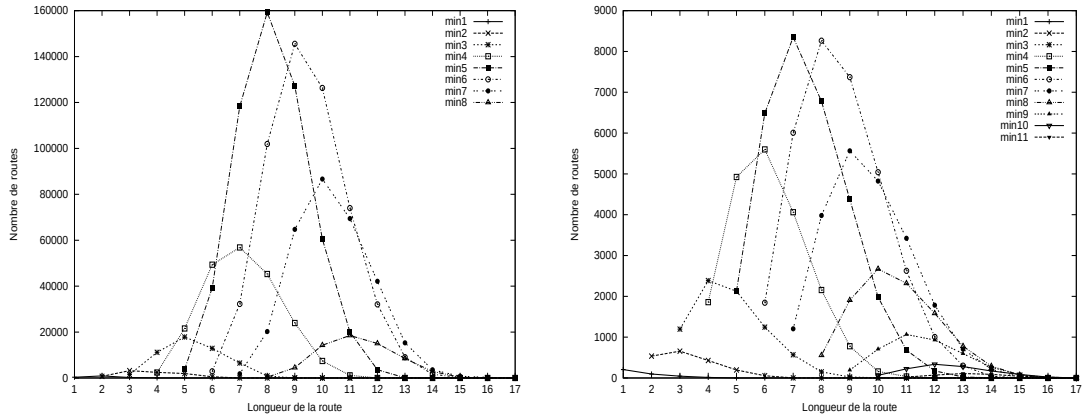


Alternet -DT(1) OpenTransit -DT(1)

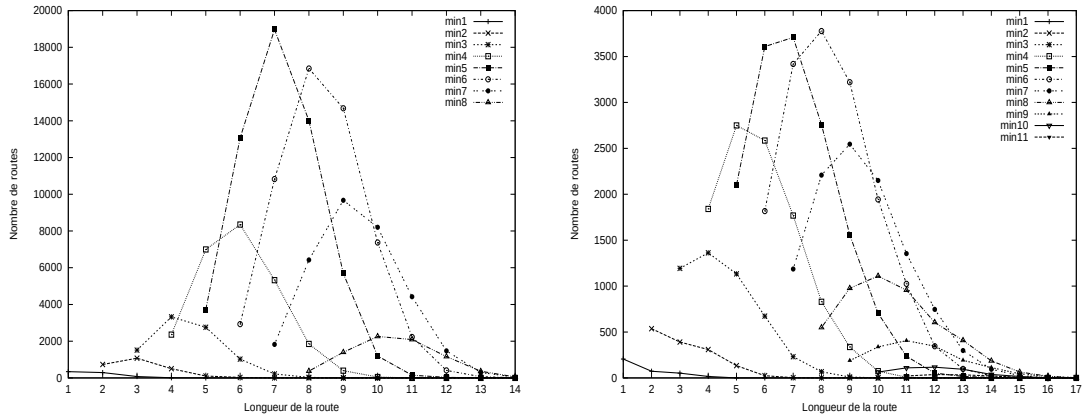


Alternet -DT(2) OpenTransit -DT(2)

FIG. 4.10 – Nombre de routes activées avec DT(1) et DT(2)



Alternet -DT(3) OpenTransit -DT(3)



Alternet -DT+(3) OpenTransit -DT+(3)

FIG. 4.11 – Nombre de routes activées avec DT(3) et DT+(3)

La plupart des cycles sur OpenTransit comportent un nombre de sauts nettement supérieur à la profondeur p utilisée dans nos évaluations de DT(p). En revanche, Alternet est une topologie composée d'un imbriquement de petit cycles. Si l'on considère l'exemple simple d'une topologie en anneau de valuation uniforme, DT(p) est capable de valider une route alternative pour toutes les destinations (si l'anneau comprend au maximum $p + 2$ sauts : un cycle composé de $p + 1$ routeurs).
Sur les deux réseaux, nous pouvons également observer que la longueur des routes alternatives n'atteint quasiment jamais la longueur maximale définie par la relation donnée dans l'équation (4.1). Cette propriété intérêssante signifie que les routes alternatives, prévues en cas de congestion ou de panne, ne sont en moyenne pas beaucoup plus longues que la route primaire.
Comme nous le verrons dans le paragraphe 4.3.2, le nombre de routes est l'un des facteurs clés pour générer une bonne qualité de couverture. Cependant cet indicateur n'est pas le seul caractérisant le taux de protection (ou de déviation pour l'évitement de congestion). La faible intersection entre les routes primaires et alternatives est également un critère essentiel.
On peut observer, sur les deux dernières figures présentées en 4.11, que DT+(3) génère l'activation d'un nombre de routes nettement plus faible qu'avec DT(3). Ceci est du à une sélection de NH-candidats plus restrictive. En revanche, les routes validées grâce à cette amélioration ne sont pas les mêmes : elles sont plus disjointes. Le paragraphe 4.3.2 met en avant cette différence. Lorsqu'aucun NH alternatif n'est validé à profondeur 1, ou que seule une ligne de routage est activée pour une interface d'entrée donnée, cette amélioration permet d'augmenter les chances de succès de la validation en profondeur. DT+(p) permet d'activer des lignes de routages là où les capacités de reroutage sont faibles tout en garantissant aux zones où la diversité des connexions est importante par nature, un minimun de validation suffisant.

Dans le paragraphe suivant, nous allons étudier le temps de convergence nécessaire à la validation des chemins en profondeur. Pour la fiabilité du réseau, il faut interpréter ces résultats selon deux perspectives. D'une part, le temps de convergence global n'influt pas sur la qualité de couverture : les routes opimales sont activées de la même manière qu'avec ECMP et ce temps global n'influt pas sur la période de restauration si une alternative locale existe. D'autre part, après stabilisation des meilleures routes, la période critique, au cas où une panne surviendrait avant la convergence du processus de validation, est déterminée par la validation du premier chemin alternatif.

4.3.1.3 Convergence

Afin d'évaluer le temps de convergence induit par notre processus de validation en profondeur, nous avons mis au point deux scénarios de pannes de liens sur Renater. Pour cela, nous avons estimé les délais de propagation de ce réseau au moyen d'une méthode de calcul orthodromique. Chaque délai est calculé en fonction de la distance physique du lien reliant deux routeurs. Sur Renater, ces délais sont inférieurs à 2ms au pire et inférieur à la milliseconde en moyenne.
Deux types de pannes ont été considérés : une panne de liens bidirectionnels en cœur de réseau, et une panne de lien en périphérie. Les figures 4.13 et 4.14 illustrent[1] la distribution des temps de stabilisation

[1] Le lecteur est invité à consulter l'annexe 4.3.3 pour observer ces deux figures en détails.

selon le type de panne. Pour correctement évaluer les délais d'émissions, les messages de signalisation (LSA, $Query-P/response-P$ et $query/response$) sont dimensionnés en fonction de la taille du réseau. Chaque paquet correspondant à une requête de validation est proportionnel au nombre de destinations concernées. La capacité des liens est considérée comme uniforme et égale à 2.5 Gigabits par seconde. En revanche, nous n'avons pas simulé le temps de calcul induit par l'algorithme DT ou par un algorithme d'un SPT classique. De la même manière, les dimensions du réseau étant modeste, le temps de mise à jour de la FIB a été négligé. L'abscisse des figures 4.13 et 4.14 représente le temps simulé, et chaque motif correspond à un temps de calcul sur un routeur donné. Ces flèches sont ordonnées en fonction de l'instant de réception du premier LSA. Les flèches simples sur ces deux figures montrent le temps nécessaire à la réception de l'ensemble des LSA émis. Les intervalles avec bornes désignent les temps nécessaires pour la validation à profondeur p. Ces mesures prennent en compte les temps d'enregistrement des lignes de routage pour le trafic local et en transit.
Dans le premier scénario la connexion est coupée de manière bidirectionelle sur les liens entre Lyon et Paris à $t = 5s$, tandis que sur le second scénario il s'agit du lien bidirectionel assurant la connexion entre Bordeaux et Toulouse. Le temps de convergence global (jusqu'à la dernière validation) dure moins de 20 ms pour mettre à jour la table de routage de l'ensemble des routeurs sur les deux scénarios.

Le temps de stabilisation global est davantage conditionné par le temps de détection que par les périodes de validation. La période de validation à profondeur 1 est inférieure à 3ms dans tous les cas, à 6ms pour $p = 2$ et à 10ms pour $p = 3$.
La période de temps liée à la validation est dépendante de la profondeur choisie, alors que l'instant de réception de la dernière annonce LSA dépend, au pire, du diamètre du réseau.
La différence visible entre les figures 4.13 et 4.14 provient de la localisation de la panne déclenchée. La distribution des LSA est conditionnée par la situation du ou des liens incriminés, ainsi les délais de réception de ces annonces sont liés aux propriétés topologiques globales alors que DT(p) est seulement dépendant du DT-voisinage à p sauts.

4.3.2 Protection et restauration

4.3.2.1 Local

Dans ce paragraphe, nous nous focaliserons sur la couverture locale générée avec DT(p) comparé aux méthodes LFA et UTURN utilisées pour le reroutage rapide.
Pour chaque lien l, nous avons d'abord calculé le nombre de routes primaires l'utilisant : $y(l)$. La figure 4.15 illustre cette distribution (les liens sont ordonnés en fonction du niveau d'utilisation). Pour les deux réseaux, les distributions semblent similaires. Une des caractéristiques communes est le fait que les réseaux sont composés de liens de deux types : les liens de cœur et les liens de périphérie. Les uns sont très sollicités, les autres ne sont utilisés que par un nombre de routes restreint. La figure 4.16 propose une perspective de la distribution de la couverture en fonction du niveau d'utilisation des arcs.

Sur la figure 4.16, le taux de protection est représenté par le ratio $\frac{y(l)}{x(l)}$ où $x(l)$ est le nombre de routes pour lequel le lien l est protégé. Les histogrammes sont triés selon l'ordre d'utilisation des liens

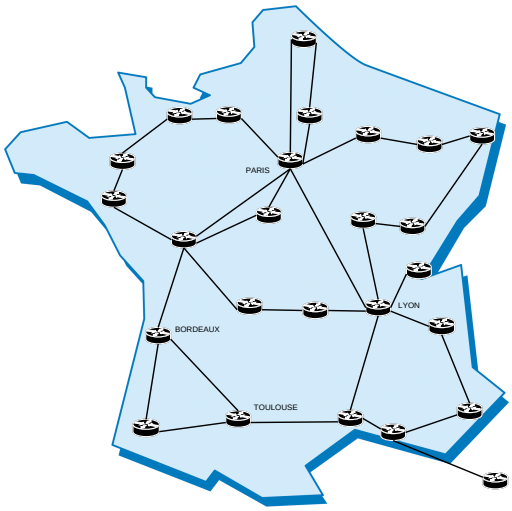


FIG. 4.12 – Topologie simplifiée de Renater

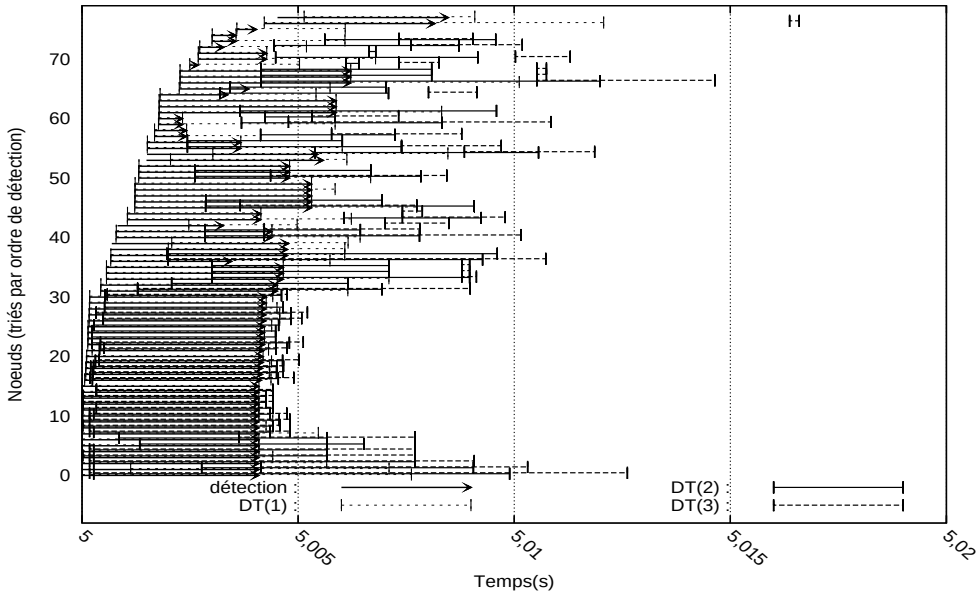


FIG. 4.13 – Temps de convergence - panne du lien entre PARIS et LYON

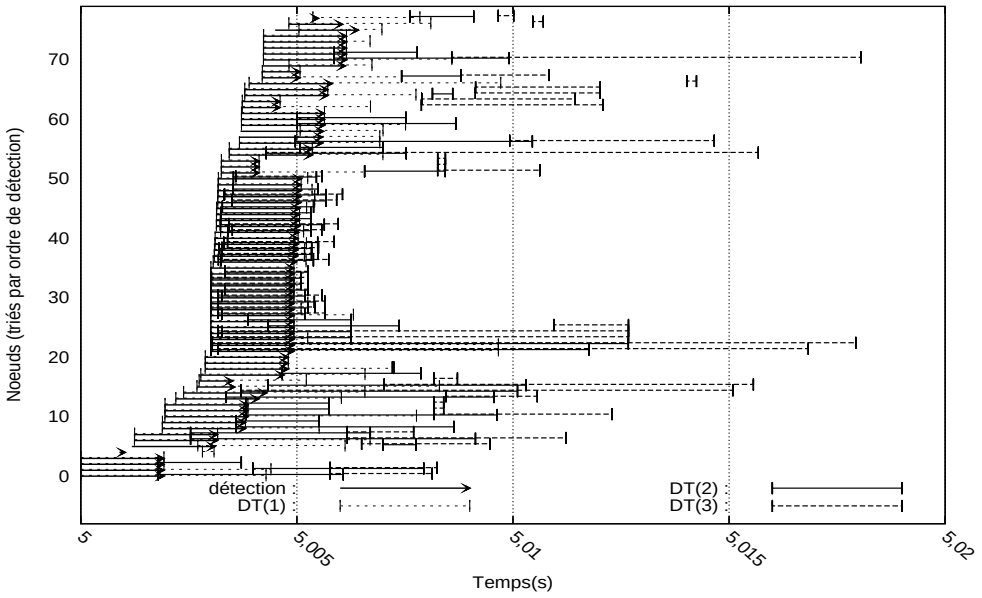


FIG. 4.14 – Temps de convergence - panne du lien entre BORDEAUX et TOULOUSE

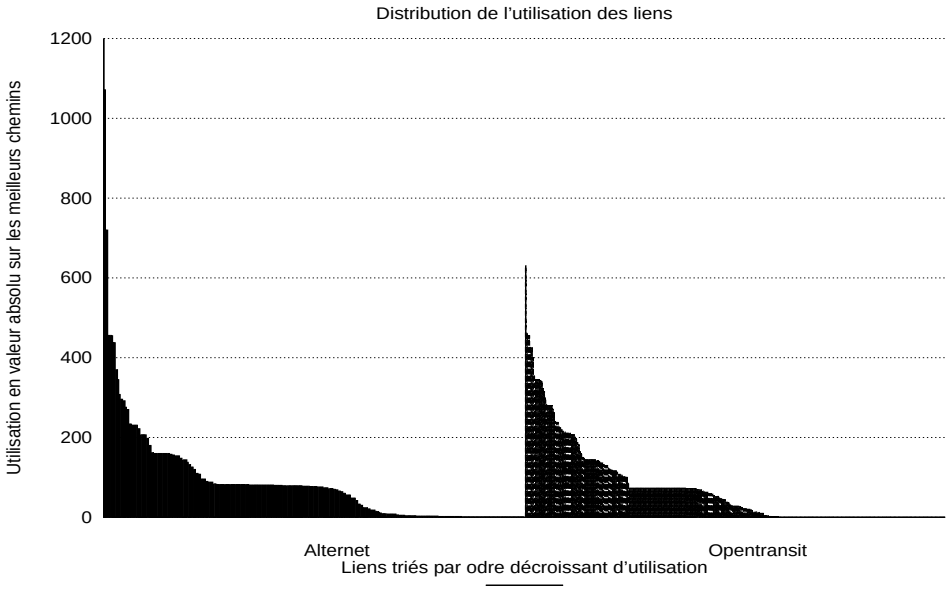


FIG. 4.15 – Distribution de l'utilisation des liens

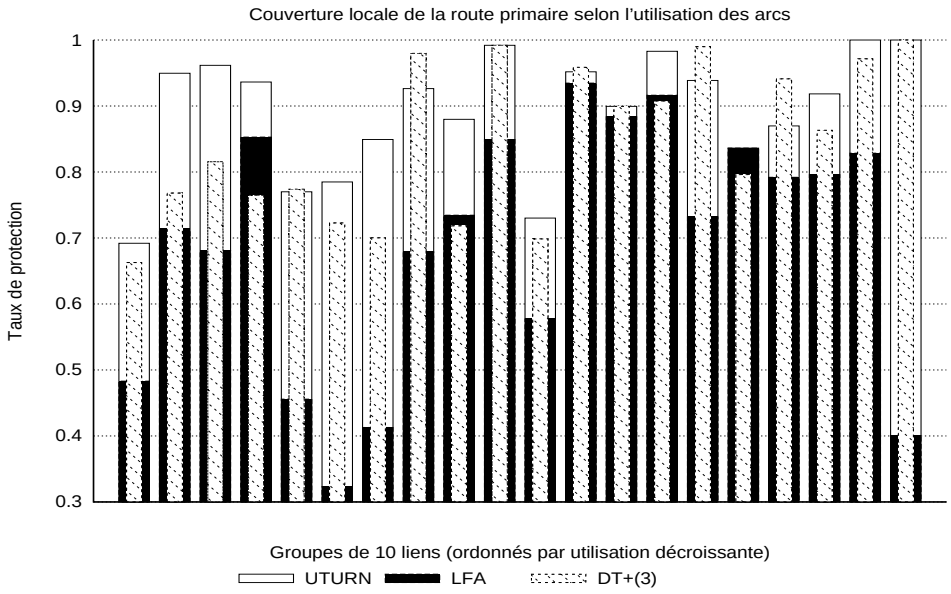


FIG. 4.16 – Protection en fonction de l'utilisation des liens

$y(l)$ et ils représentent des groupes de dix liens pour une meilleure lecture de la figure.
Cette figure met en avant l'absence de corrélation entre le niveau d'utilisation des liens et leur taux de protection. En outre, les résultats obtenus avec DT(3) ne sont pas nécessairement supérieurs à LFA et inférieurs à UTURN. Selon les liens et la configuration topologique locale, DT(3) est capable de protéger des arcs qui ne le seraient pas avec UTURN et inversement.
La figure 4.10 propose une distribution de la couverture selon la longueur du chemin optimal. Sur les topologies d'évaluation obtenues via mrinfo, certains liens ne sont pas protégeables dans la mesure où leur panne entraîne la partition du réseau en deux (liens isthmes). Nous avons retiré ces liens pour mesurer la couverture des 4 topologies concernées. Moins de 10% des liens sur Alternet et moins de 30% des liens sur OpenTransit sont des liens isthmes.

Dans l'ensemble de nos évaluations, pour $p > 1$, nous avons utilisé la modification proposée sur la sélection opérée par DT : DT+(p). Sur la figure 4.17, on observe que UTURN est légèrement plus efficace en terme de couverture moyenne que DT(3) quelle que soit la longueur des routes primaires. Néanmoins, DT(3) produit systématiquement une meilleure couverture moyenne que LFA.
Le tableau 4.2 synthétise l'ensemble des résultats obtenus sur les cinq topologies d'évaluation. Ces mesures ont été calculées en utilisant la formulation (3.1) du paragraphe 3.1.6 et en considérant seulement la protection du chemin primaire entre toute paire de nœud du réseau.
Les résultats regroupés dans ce tableau semblent confirmer la tendance observée sur la figure 4.17. DT(3) est capable de produire une couverture proche de UTURN alors que les routes sont aussi utilisables pour mettre en oeuvre un routage proportionnel. La condition IIC doit être vérifiée pour un

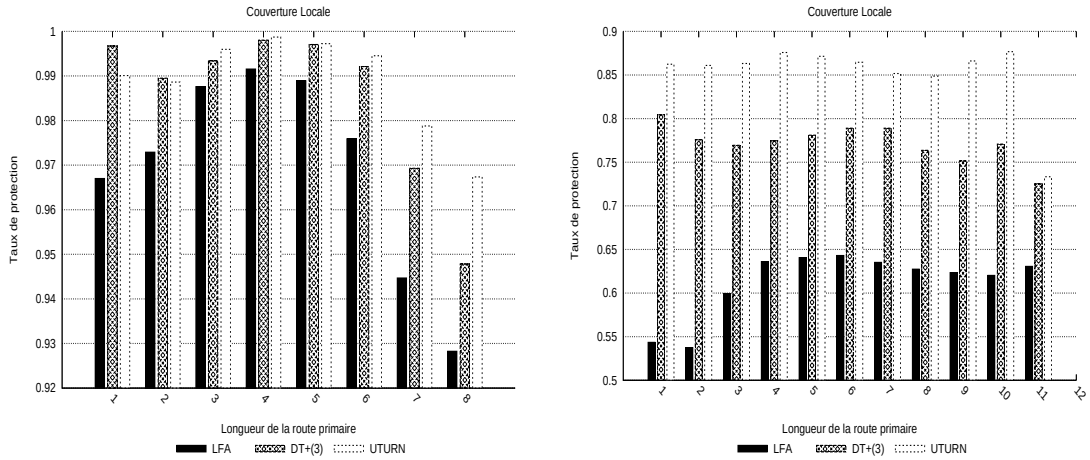


FIG. 4.17 – Couverture locale : distribution selon la longueur des chemins primaires

ensemble de composition de NHs aux coûts inégaux lorsque notre procédure de validation utilise une profondeur $p > 1$, alors que les protocoles de reroutage rapide tels que LFA ou UTURN ne prennent en compte que les meilleurs chemins pour satisfaire le critère d'absence de boucles. Malgré cette différence notable, DT(p) est capable de produire des résultats de couverture comparable aux techniques basiques de reroutage rapide. Les prochains sauts activés via DT(p) ont un autre intêret : ils peuvent être utilisés pour le partage de charge.
On peut observer que la valuation et la longueur des cycles du réseau GEANT ne favorise pas notre procédure de validation par rapport aux autres techniques. Sur cette topologie LFA produit de meilleurs résultats que DT+(3) pour la protection des routes primaires. Cet effet est en partie provoqué par la valuation hétérogène de GEANT, la condition LFA en bénéficie davantage que DT+(3).
Pour la redirection en cas de congestion, on observe que DT(p) produit une couverture supérieure à celle générée par LFI quel que soit $p \geq 1$. Néanmoins, pour $p = 1$, la couverture est seulement très légèrement supérieure à celle de LFI. Ces résultats sont à mettre en relation avec ceux présentés dans le paragraphe 4.3.3.

4.3.2.2 Global

Dans ce paragraphe, nous admettons que le réseau dispose d'un protocole de notification du même type que celui décrit en 3.1.7. Le tableau 4.3 regoupe l'ensemble des résultats (voir la formulation (3.2)). La figure 4.18 représente la distribution de la couverture en fonction de la longueur des routes primaires sur les topologies Alternet et OpenTransit.
Par nature, les mesures effectuées pour la protection globale sont meilleures que pour la couverture locale. Cependant, le temps de réaction est, dans ce cas, dépendant de la profondeur à laquelle l'annonce doit parvenir. La période nécessaire à la notification d'une panne est liée au délai séparant le routeur détecteur du routeur capable de dévier son trafic sur une alternative locale.

Réseau	LFI	LFA	DT(3)	DT+(3)	UTURN
Alternet	17.8	98	98.4	99.2	99.4
OpenTransit	15.9	62.1	66.8	77.8	86.6
Global Crossing	19.5	73.5	77.8	87.1	91.1
Renater	10	51.5	60.1	67.3	69.2
GEANT	64.5	87.2	81.2	86	90.9

TAB. 4.2 – Couverture locale - synthèse

Nous avons réalisé cette analyse uniquement pour DT(p) et LFI, car LFA et UTURN ne sont pas des règles de multiroutage. La composition de proche en proche des NHs de secours et des NHs primaires activés avec UTURN ou LFA provoquerait des boucles de routage.

Sur des topologies à valuation uniforme, LFI n'est pas capable de protéger les routes de longueur 1 : les liens. La figure 4.7 en est un exemple. DT(3) permet au réseau de disposer d'une couverture presque complète sur toutes les routes primaires. En revanche, la couverture assurée par LFI reste faible quelle soit la longueur de la route.
Sur cet exemple, nous pouvons également observer l'impact de la modification apportée à DT(p) sous le dénominatif DT+(p). Cette amélioration garantit, à complexité réduite, des résultats meilleurs que DT(p) (excepté pour les routes primaires de longueur 11 sur OpenTransit).

L'hétérogénité des résultats montre à quel point les caractéristiques topologiques influent sur la qualité du reroutage. La valuation des liens est par exemple un critère essentiel. Lorsque la valuation est très hétérogène, comme c'est le cas sur GEANT, l'optimisation décrite en 2.2.3 permet, grâce aux messages *Query'-P*, de satisfaire plus facilement les critères de validation.
Afin de produire une couverture complète, la véritable difficulté pour le processus de validation DT(p) est de garantir une diversité controlée : le nombre de NHs candidats calculés avec DT limite les chances de succès à profondeur élevée mais génère par définition plus de compositions possibles. La longueur des cycles du réseau et leur entrelacement est un facteur clé pour l'ajustement de la profondeur de

Réseau	LFI	DT(1)	DT(3)	DT+(3)
Alternet	34.2	98.5	99.6	99.8
OpenTransit	33.3	77	86.2	91.7
Global Crossing	43.7	80.7	91.3	96.1
Renater	21	65	73.5	85

TAB. 4.3 – Couverture globale - synthèse

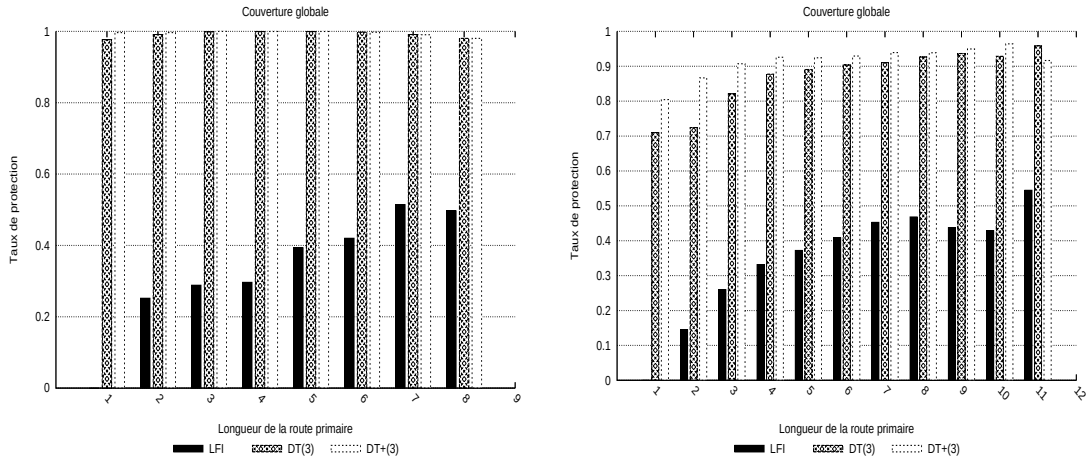


FIG. 4.18 – Couverture globale : distribution selon la longueur des chemins primaires

recherche. Par exemple, sur une topologie en anneau, la couverture locale assurée par *DT(3)* est, par construction, meilleure que celle générée avec *UTURN*. La pré-sélection des DT-voisins est également critique pour la validation des NHs.
Ces résultats ne concernent que la protection des meilleurs route alors que le multiroutage permet une commutation répartie sur des routes aux coûts hétérogènes. Néanmoins, ils reflètent une certaine forme de couverture globale pour l'ensemble des routes : s'il existe un NH alternatif alors celui-ci est protégé par le NH primaire et vice-versa. Nous n'avons pas considéré le cas de la panne de liens multiples qui est un phénomène relativement rare, sauf pour les SRLG.

4.3.3 Reroutage en cas de congestion

Ce paragraphe est consacré à l'étude des bénéfices générés par la diversité des routes pour l'équilibrage du trafic. L'objectif est de mesurer, pour une politique de répartition donnée, l'intérêt pour chaque routeur de disposer d'un large ensemble de routes afin de contourner les congestions. Nous avons analysé plusieurs indicateurs pour comparer l'efficacité de LFI, DT(1) et DT(3) associés au module de partage de charge décrit dans le paragraphe 3.2.4.2. Les deux indicateurs suivants ont été l'objet d'une attention particulière :

- Le nombre global de paquets perdus.
- La charge de chaque lien à l'échelle *t=1s*.

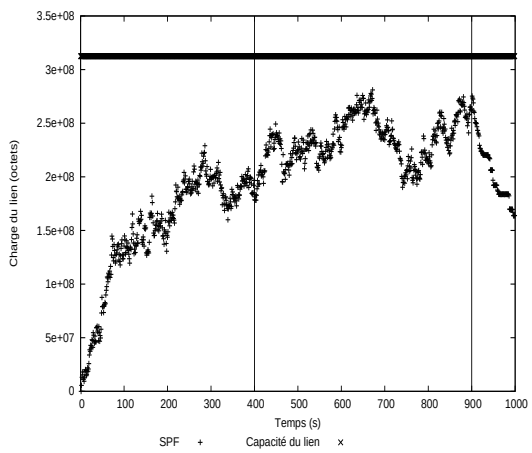
Le tableau 4.4 résume nos principaux résultats de simulation. La première ligne fournit le taux de réduction de perte de paquets par rapport aux pertes générées par le routage SPF. La seconde ligne décrit le taux de charge moyen sur le lien le plus chargé à l'échelle de temps t. Les moyennes de charge obtenues sur chaque simulation ont été analysées avec un niveau de confiance de 95% (avec l'outil de statistique *R-stats* (Dal02)). L'intervalle de confiance est inférieur à 0.1% de la capacité du lien pendant la période stable de chaque simulation.

Pour configurer le seuil signifiant la détection de la congestion, notre contrôleur de charge a été paramétré avec ces deux bornes : $\alpha = \frac{1}{2}$ et $\beta = \frac{\alpha}{2}$. Les moyennes représentent un échantillon de douze scénarios de simulation : 3 de n vers 1, 4 de 1 vers 1 et 5 de 1 vers n. Dans deux de ces scénarios, le contrôleur de charge déplace simplement la congestion vers un autre lien en aval. Dans ces cas, les améliorations ne sont pas visibles, il se peut même que, durant certains intervalles de temps, la charge déviée génère un trafic dont les rafales peuvent être nuisibles sur un lien en amont.
Dans l'ensemble, les résultats semblent confirmer que la diversité des prochains sauts générés avec DT(p) présente un intérêt significatif. Le réseau est plus fiable car mieux protégé face aux congestions. Globalement, le nombre de pertes et la charge moyenne du lien le plus chargé sont réduits de manière significative.
Afin de faciliter l'interprétation des résultats, il est intéressant d'observer les propriétés d'un exemple isolé. Il s'agit d'un scénario de congestion $1 \rightarrow n$ initié depuis le routeur si (voir la figure 4.3 pour la valuation et les identifiants des routeurs). Les figures 4.19 et 4.20 décrivent l'état du lien $\{si, at\}$ seconde par seconde selon la règle SPF, LFI ou DT(p) choisie. Ce lien est le plus chargé durant la période stable de la simulation. Les figures 4.19(a),(c) et 4.20(a),(c) indiquent la charge mesurée sur le lien le plus chargé via une méthode de routage donnée en fonction de la précédente. Les figures 4.19(b),(d) et 4.20(b),(d) repésentent le nombre de pertes de paquets mesurées par seconde.

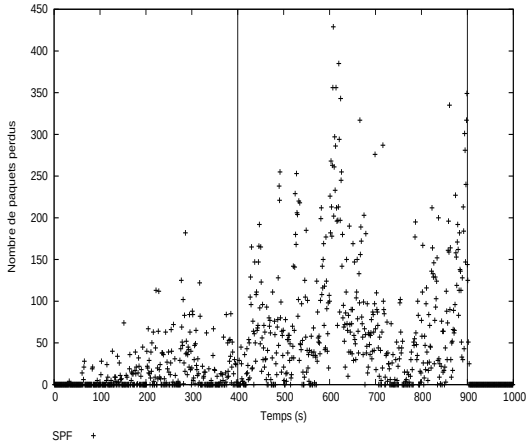
Cette série de figures met en évidence la pertinence du choix des NHs les plus appropriés. DT(3) facilite la validation d'alternatives plus efficaces pour le contournement de congestion. Les lignes de routage validés grâce à l'approfondissement de la procédure de recherche permettent d'activer des routes que les autres méthodes de validation ne sont pas capables d'utiliser.
Le routage proportionnel profite de cette diversité accrue pour détourner plus efficacement les paquets en cas de congestion isolée. On constate que seul DT(3) est capable de satisfaire les exigences du contrôleur de charge : le taux moyen de charge sur le lien congestionné est inférieur en moyenne à la moitié de sa capacité ($\alpha = \frac{1}{2}$). DT(p) permet au routage de s'adapter à plus de situations. Si les volumes de trafic ne sont uniformément répartis sur les intervalles correspondant aux proportions, il se peut qu'il n'existe peu ou aucune alternative pour les flux les plus volumineux. Dans ce cas, un routeur DT(p) a plus de chance d'être capable de dévier la charge induite par ces flux éléphants qu'un routeur vérifiant la régle LFI. De manière générale, la diversité des routes générées par DT(p) profite à un ensemble de

Indicateurs	LFI	DT(1)	DT(3)
taux moyen de réduction des pertes	3.8	4.2	6.5
charge moyenne sur le lien le plus chargé (SPF :76%)	61.4	61.4	51.8

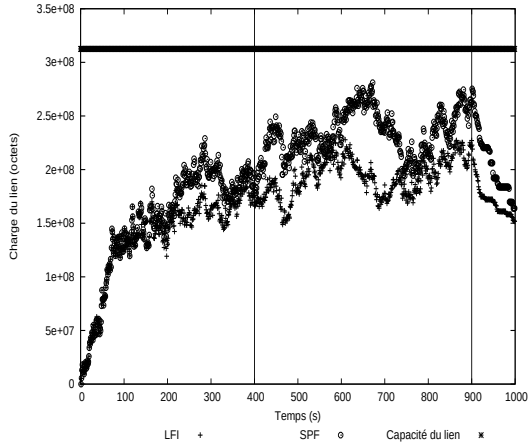
Tab. 4.4 – Améliorations apportées par DT(p)



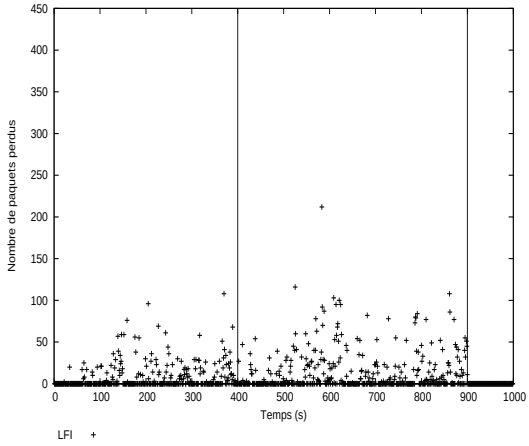
(a) Charge avec SPF



(b) Pertes avec SPF

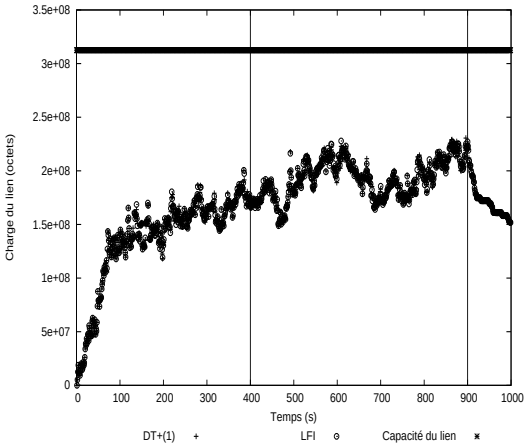


(c) Charge comparée entre SPF et LFI

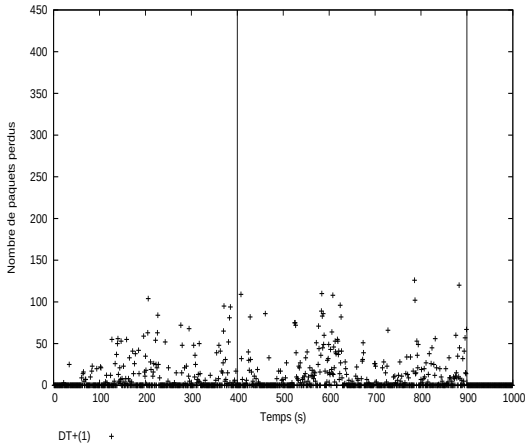


(d) Pertes avec LFI

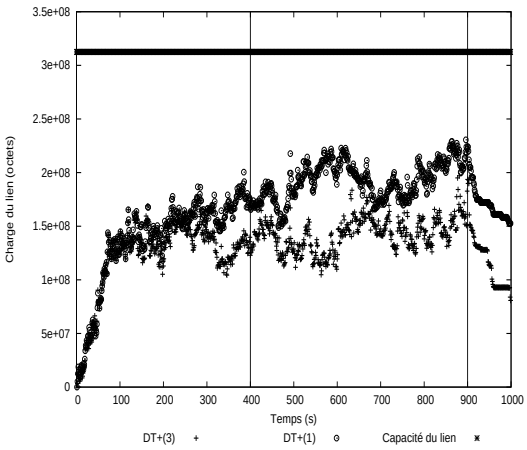
FIG. 4.19 – Etat du lien le plus chargé ($\{de1, de2\}$) avec SPF et LFI



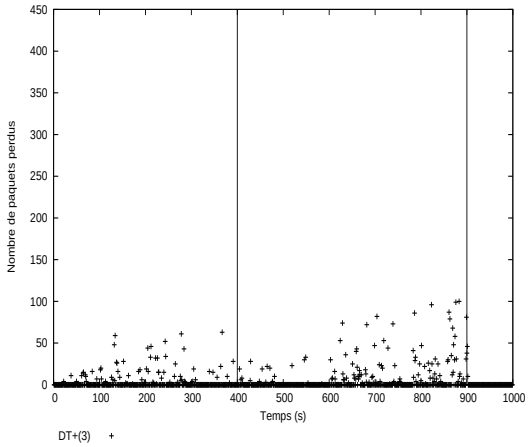
(a) Charge comparée entre LFI et DT(1)



(b) Pertes avec DT(1)



(c) Charge comparée entre DT(1) et DT(3)



(d) Pertes avec DT(3)

FIG. 4.20 – Etat du lien le plus chargé ($\{de1, de2\}$) avec DT(1) et DT(3)

flux plus important.
Nous avons également analysé les délais d'acheminement de chacun des flux TCP générés. La différence de durée de vie (du premier paquet au dernier acquittement) pour un flux donné, entre un routage SPF et le multiroutage n'est pas vraiment significative avec les conditions de simulation définies. Bien que légèrement à l'avantage du multiroutage en général, les chemins les plus longs provoquent l'allongement des délais. DT(1) génère les meilleurs résultats en termes de latence, mais la différence avec LFI est très faible. Avec DT(3), on constate une détérioration de l'acheminement des flux comparé à LFI ou DT(1) lorsque les fenêtres TCP sont limitées à 65 paquets.
Les routes alternatives présentent un RTT fréquemment plus élevé que les routes primaires. Cet accroîssement réduit les performances si le débit du flux est conditionné par la taille de la fenêtre choisie au niveau de la couche transport. Nous avons simulé le comportement de DT(p) ($p > 1$) dans le cas de fenêtre de taille supérieure. Cette contrainte levée, DT(p) permet de détourner les longs flux de leur route primaire sans que l'augmentation du RTT ne dégrade les délais d'acheminement. Les routes les plus coûteuses activées via DT(p) sont utiles pour les flux les plus longs et lorsque le problème est critique c'est-à-dire en cas de congestion persistante ou en cas de panne.
Les figures 4.21 et 4.22 illustrent la différence en termes de délais d'acheminement entre LFI et SPF puis entre DT(3) et LFI. Ces figures ont été obtenues sur le scénario précédemment utilisé : une congestion 1 vers n sur le lien $\{si, at\}$.
Les différences entre les délais sont triées selon la durée de vie des flux TCP mesurée avec SPF. Une valeur positive signifie un allongement du délai pour un flux donné et inversement. Nous avons observé que, pour une route donnée, LFI permet une réduction du délai d'acheminement proportionnelle à la durée de vie d'un flux (elle-même liée au volume de la demande). En revanche, DT(3) est moins performant que LFI car certaines routes sous-optimales présentent un RTT élevé.

La figure 4.23 présente une moyenne obtenue pour tous scénarios confondus : l'utilisation relative du lien le plus chargé selon sa capacité et la règle utilisée. Les données ont été recueillies sur le lien le plus chargé en fonction du scénario de congestion analysé.
L'abscisse donne le pourcentage de la charge du lien à l'échelle de la seconde en fonction de sa capacité. Cette charge est représentée de manière cumulative, il s'agit de la somme des demandes de trafic à commuter sur le lien. L'ordonnée correspond à l'utilisation cumulée du lien dans le temps. On peut interpréter un point x, y appartenant au graphe de la manière suivante (prenons l'exemple d'une charge inférieure de moitié à la capacité du lien) : la charge est inférieure à $x(= 50\%)$ pendant $y(= 40\%)$ du temps simulé avec SPF et durant $y(= 58\%)$ du temps avec DT(3).

DT(3) permet de réduire le temps d'utilisation critique des liens chargés de manière significative par rapport aux règles LFI ou SPF. Les résultats obtenus avec DT(1) sont similaires à ceux obtenus avec LFI.
La charge excédentaire est généralement déviée sur des liens peu utilisés. Ces liens ne sont pas comptabilisés dans le calcul car ils restent moins chargés que le lien initialement incriminé. DT(p) permet aux routeurs de mieux distribuer leur excédent de charge sur leurs liens locaux les moins chargés. L'équilibrage est local et peut s'apparenter à une forme d'ordonnancement de files d'attente multiples.

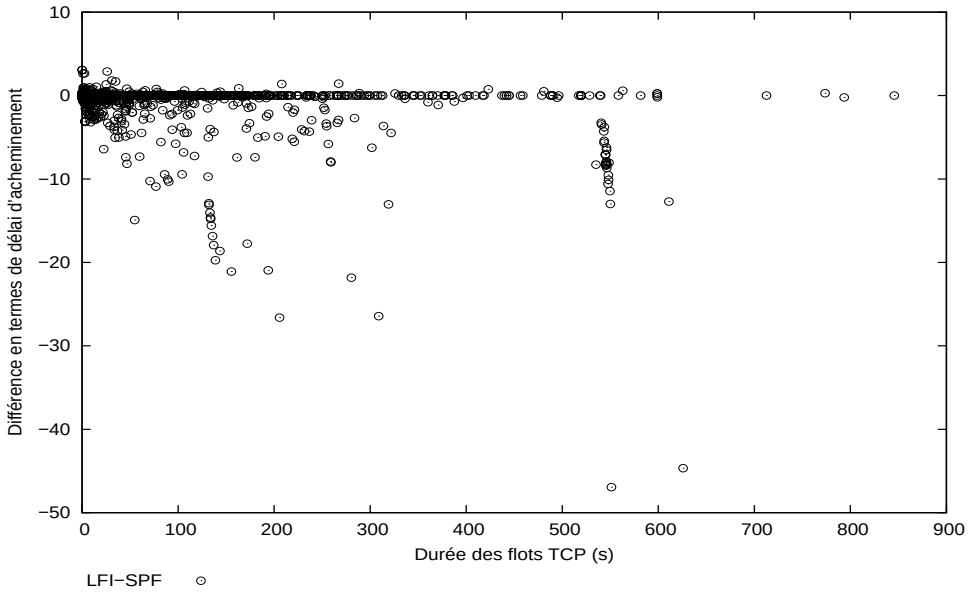


FIG. 4.21 – Comparaison sur les délais d'acheminement entre LFI et SPF

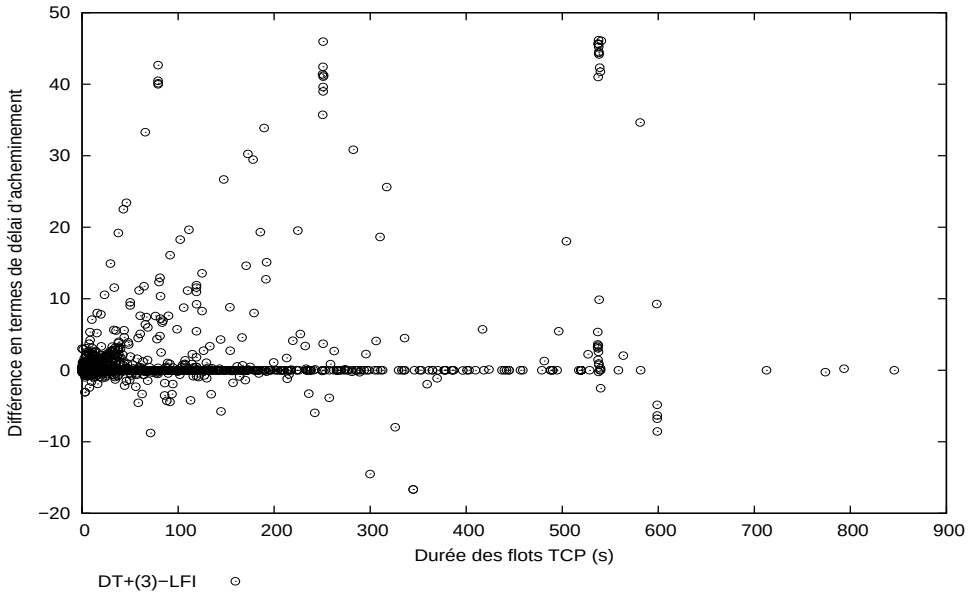


FIG. 4.22 – Comparaison sur les délais d'acheminement entre DT(3) et LFI

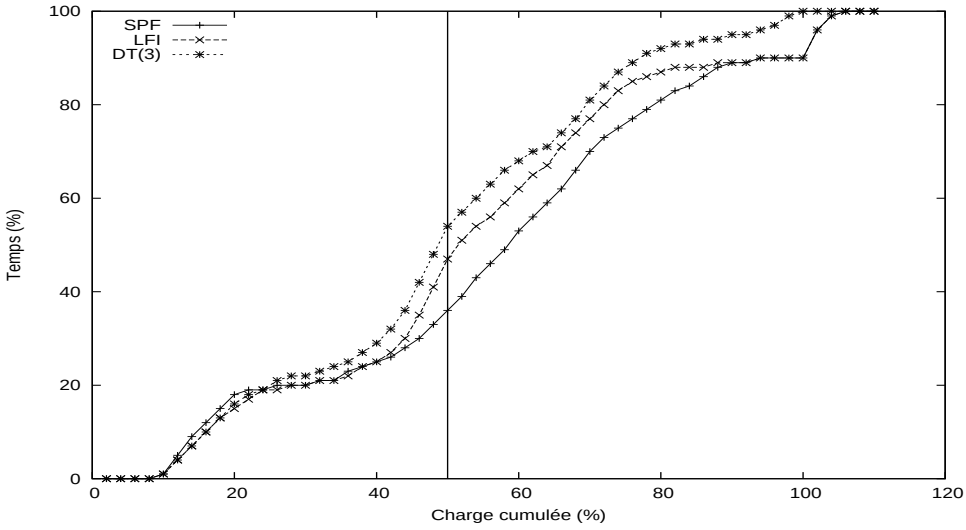


FIG. 4.23 – Répartition de la charge cumulée

La configuration du paramètre p est un enjeu clé pour minimiser la charge des liens les plus utilisés. Il faut essayer de maintenir un compromis entre la réduction de la charge maximal et l'utilisation des routes les plus longues.
Dans nos travaux futurs, il sera nécessaire de définir un mode coopératif entre les routeurs afin qu'ils puissent réguler conjointement la charge. Cette tâche est néanmoins plus complexe et présente des risques d'instabilité. Dans les cas où l'excès de charge induit par une congestion sature les liens du routeur en aval constituant une alternative locale, il est alors utile de mettre en œuvre une procédure de notification. Celle-ci permettrait d'alerter les routeurs en amont de l'incapacité ou de l'insuffisance locale de déviation. Une telle notification peut prendre forme dans des annonces relayées en amont du point de congestion. La transmission des annonces peut par exemple être sélective si celles-ci ne concernent que les prédecesseurs générant le plus de trafic. Pour une analyse appropriée, la granularité des mesures doit correspondre aux couples (interface d'entrée, destination). Par ailleurs, chaque annonce doit comporter une valeur désignant de manière relative ou absolue la charge à dévier. Cette valeur peut être déterminée au moyen des seuils utilisés pour détecter les congestions.

La diversité des routes permet de mettre en œuvre un routage fiable même lorsque le trafic est anormalement élevé. Les résultats obtenus indiquent que nos procédures forment un compromis intérêssant pour garantir un routage robuste. Comparé aux règles généralement utilisées pour garantir l'absence de boucles, nos méthodes permettent d'activer des routes supplémentaires dont l'intérêt est double. Celles-ci permettent de réguler le trafic en cas de pannes et/ou de congestions.
Par rapport aux conditions LFI ou ECMP, nos procédures sont supérieures en termes de diversité, autant pour le nombre moyen de prochains sauts que pour le nombre de routes générées. Nous avons mis en avant l'importance de la distribution des routes alternatives pour produire une couverture effi-

cace grâce à l'amélioration DT+(p). Nos simulations de trafic sur GEANT nous ont permis de mesurer l'intérêt de la diversité des routes et du taux de couverture pour déployer des mécanismes de redirection efficaces.

Conclusion

Le routage est l'une des clés de voûte pour assurer la fiabilité des réseaux IP. L'un des principaux objectifs est de garantir un service non dégradé en cas de problème. La faculté d'adaptation d'un réseau aux situations imprévisibles est capitale pour sa pérennité. Le réseau doit proposer un routage flexible réagissant rapidement aux brusques variations de charge ou aux pannes. En revanche, il faut que cette caractéristique dynamique soit maîtrisée pour éviter des oscillations trop fréquentes.
Dans ce travail de thèse, nous nous sommes focalisés sur les mécanismes de la couche réseau sans négliger les caractéristiques de la couche transport. Notre étude s'est portée sur le routage multichemins au saut par saut qui présente deux avantages importants : d'une part, le temps de reprise sur panne peut être réduit en utilisant des routes alternatives locales et pré-calculées, d'autre part le trafic peut être réparti simultanément sur plusieurs chemins afin d'équilibrer globalement la charge.
La mise en place d'une commutation offrant ces deux caractéristiques est plus difficile qu'il n'y paraît. En effet, nos contributions permettent d'acheminer les paquets sur des routes aux coûts inégaux, or dans ce cas le principe de sous-optimalité des meilleurs chemins n'est pas satisfait. L'utilisation simultanée de chemins aux coûts inégaux est soumise à un certains nombre de règles de validation pour la cohérence du routage. La commutation au saut par saut ne doit pas provoquer la formation de boucles de routage. Ces boucles sont d'autant plus difficiles à élaguer lorsque les routeurs proposent une diversité de commutation élevée.
Contrairement aux techniques de reroutage rapide, le multiroutage est capable d'acheminer simultanément les flux via des routes alternatives pour répartir la charge. Il s'agit d'un avantage notable pour prévenir la formation de congestions. Dans ce cas, le critère utilisé pour garantir l'absence de boucles de routage nécessite un processus de validation plus fin que ceux utilisés par les protocoles de reroutage rapide.
L'état de l'art réalisé dans le premier chapitre donne un aperçu des méthodes existantes pour permette l'utilisation de routes non optimales. Cependant, les règles utilisées sont généralement très restrictives pour générer une diversité de chemins suffisante. D'autres approches permettent d'ajouter un degré de flexibilité supplémentaire mais ne garantissent pas un routage sans boucle au niveau routeur.
Dans le second chapitre, nous apportons une réponse à ces problèmes en présentant une solution de multiroutage distribuée dont le déploiement est extensible. Notre contribution est double : notre première proposition est un algorithme de recherche opérationnelle pour le calcul de multichemins. Celui-ci calcule au moins une alternative de routage si elle existe pour une destination donnée. Dans le cadre de la protection, cette propriété permet la mise en œuvre d'une alternative de routage systématique quelle que soit la destination du trafic. Notre seconde proposition est une procédure de validation distribuée. Celle-ci peut s'appliquer au graphe de composition généré par la combinaison de proche en proche des prochains sauts retenus par notre algorithme de calcul. Ce graphe est élagué, en tenant compte de l'interface d'entrée, des compositions générant des boucles de routage pour constituer un ensemble de routes actives. Nos deux procédures peuvent être découplées : la phase de validation en profondeur est

indépendante du graphe de composition généré via notre algorithme de calcul des chemins et ce graphe peut être élagué avec une règle de validation différente de celles que nous avons définies.
La prise en compte de l'interface entrante étend le processus de commutation et permet de mettre en place une diversité de routes accrue pour un routage plus robuste. En effet, le graphe de composition est élagué de manière plus fine que lorsque le routage est indépendant de l'origine du trafic : pour une destination donnée, l'espace des prochains sauts actifs est différent selon la provenance des paquets. Notre solution est extensible et n'est pas liée au nombre de sources potentielles du domaine. La complexité de commutation dépend des paramètres locaux : le degré des nœuds de routage. Nous avons également proposé des mécanismes simples pour faciliter le passage à l'échelle en interdomaine. La diversité intra et interdomaine peut être combinée en distribuant les capacités de commutation selon le type de routeurs. Les routeurs de bordure profitent de la diversité interdomaine. Pour un préfixe donné, le domaine est alors partitionné en fonction du routeur de sortie choisi. Les routeurs de cœur bénéficient d'une diversité intradomaine vers le routeur de périphérie dans la limite de la partition à laquelle ils appartiennent. Néanmoins, plusieurs problématiques liées au passage à l'échelle restent en suspens. Quelle est la distribution de diversité la plus adéquate : faut t-il favoriser la diversité des chemins intra ou interdomaines ? Comment les routeurs de cœur doivent ils réagir en cas de panne sur le routeur de sortie choisi ?
Dans le troisième chapitre, nous avons mis en avant les objectifs d'une telle démarche : proposer un routage capable de générer une couverture performante en terme de protection tout en garantissant la possibilité de mettre en œuvre un routage proportionnel efficace. D'une part, nous avons introduit la notion de couverture en amont en admettant que les routeurs puissent coopérer dans un périmètre plus ou moins restreint afin d'annoncer les pannes non contournables localement. D'autre part, nous avons proposé un contrôleur de charge incrémental basé sur une analyse locale des demandes. La détection des congestions est réalisée via des seuils configurables et permet de dévier le trafic sur les prochains sauts les moins chargés. La simplicité de notre approche réactive et locale permet de minimiser le problème de la stabilité afin d'éviter les oscillations de charge pénalisantes. Nous avons aussi pris en compte la nature des flux TCP en utilisant des solutions simples pour éviter le déséquencement des paquets appartenant à un même flux. Ces propositions peuvent être étendues pour engendrer une différenciation de service à la source ou de manière distribuée.
Afin d'évaluer nos différentes propositions nous avons modélisé plusieurs topologies et scénarios. Leurs caractéristiques ont été recueillies sur la base de données réalistes et pertinentes. Ces outils nous ont été utiles pour simuler les propriétées de nos contributions par rapport aux techniques existantes.
Les résultats obtenus sont présentés dans le dernier chapitre. Ils soulignent l'efficacité de nos méthodes pour générer une diversité élevée en terme de flexibilité pour le routage. Nous avons analysé cette diversité selon trois perpectives : le nombre de prochains sauts moyen, le nombre de routes et la qualité de la couverture. Ces différentes angles de vue nous ont permis de mieux interpréter les caractéristiques de nos propositions.
La profondeur de recherche utilisée pour la validation est un critère déterminant dans la qualité des routes alternatives générées. Selon les caractéristiques topologiques locales et le degré des routeurs,

la diversité en nombre de routes est plus ou moins limitée par le nombre de liens sortants. Lorsque celle-ci est faible et que les alternatives potentielles forment de longs détours, alors la profondeur p de l'analyse est cruciale : la difficulté est de trouver un compromis entre complexité des procédures et nombre de routes activées. Ainsi, là où les capacités en termes de diversité sont faibles pour des raisons topologiques, il est possible de contourner les disparités du maillage pour valider des chemins plus longs. Cependant, dans la mesure où le processus de validation doit vérifier l'ensemble des compositions possibles, cette faculté est dépendante du nombre d'alternatives sélectionnées par l'algorithme de calcul de chemins sous-jacent. Le dernier chapitre met en avant l'importance de distribuer équitablement la diversité sur chaque routeur. En limitant le nombre de prochains sauts candidats, il est possible d'activer des routes alternatives de manière plus équilibrée pour l'ensemble du réseau. La diversité est alors limitée là où elle est par défaut suffisante et augmentée dans les zones moins maillées. Ainsi le réseau bénéficie d'un meilleur taux de protection en cas de pannes ou pour le contournement de congestions. Une autre approche possible consisterait à attribuer une profondeur d'analyse différente à chaque routeur selon les caractéristiques topologiques du voisinage.
Afin d'évaluer nos propositions dans un registre différent, nous avons modélisé différents scénarios de congestions. Les résultats obtenus sur un réseau hétérogènement chargé indiquent de manière significative l'importance de la diversité des routes engendrées. Lorsque les possibilités de redirection sont limitées à un sous-ensemble de destinations possibles, il se peut que l'excédent de trafic ne soit que partiellement redirigé. En effet, la charge générée par les multiples demandes n'est pas nécessairement uniformément répartie sur l'ensemble des destinations. De plus, dans le cadre de nos contributions, la commutation est aussi fonction de l'interface d'entrée. Il se peut qu'un couple (interface d'entrée,destination) spécifique soit, à lui seul, à l'origine de la congestion. Dans ce cas, la qualité de la déviation est entièrement conditionnée par l'existence d'une alternative pour ce couple. Nos procédures garantissent une couverture plus importante qu'avec les règles de multiroutage usuelles. Nous avons constaté qu'une profondeur de validation élevée favorise une meilleure répartition de la charge entre les liens surchargés et ceux peu sollicités. En revanche, l'utilisation de routes alternatives trop longues entraîne la dégradation des délais d'acheminement lorsque les débits sont limités par la fenêtre d'émission de TCP.

De nombreux aspects, et plus particulièrement pour le partage de charge, restent à approfondir. Nous avons par exemple évoqué les différences de réactions notables entre problèmes persistants et problèmes transitoires. Le choix de l'échelle de temps est critique selon la nature du problème à résoudre. Par exemple, les micro-congestions ne semblent pas évitables par le biais du routage. Comment définir un seuil efficace pour un compromis entre réactivité et stabilité ?
Par ailleurs, l'étiquetage des flux peut permettre de contribuer à la qualité de service. Dans nos simulations, l'étiquette est utilisée pour éviter le déséquencement des paquets appartenant à une même rafale TCP. Cependant, dans un contexte de routage à qualité de service celle-ci peut être mise à profit pour une commutation sensible au niveau de priorité des flux. Il nous reste à approfondir les aspects relatifs au choix de l'étiquette. Est-ce que la commutation doit tenir compte du coût statique des chemins ou

doit-elle se baser sur des informations de métrologie relatives aux routes ? Sur quels critères (volume, type d'application, etc) associer un flux à une étiquette dont la signification serait globale sur un domaine ?
Nous nous intéressons également aux modèles de distribution des annonces de notification. Quel que soit l'objectif, l'évitement de panne ou de congestion, la notification des ressources résiduelles amène diverses problématiques. La nature de la coopération, les délais de notifications engendrés et la stabilité d'un tel processus se révèlent un enjeu majeur pour la suite de nos travaux. Pour l'équilibrage de charge en particulier, ce problème reste ouvert dans un contexte distribué. Comment et qui prévenir en amont pour obtenir le routage le plus efficace en termes de délais ? Quels sont les indicateurs les plus pertinents ? Est-ce que ces solutions de notification distribuées peuvent s'appliquer sur des réseaux très chargés ?

Liste des publications

Communications à des manifestations internationales à comité de lecture :

[1] Pascal Mérindol, Jean-Jacques Pansiot, Stéphane Cateloin, *Path Computation for Incoming Interface Multipath Routing*, ECUMN07, 4th European Conference on Universal Multiservice Networks IEEE Computer Society Press pages 75–85 IEEE Toulouse, France - february 2007 (11 pages, taux de sélection $< 50\%$).

Articles à paraître (acceptés) :

[2] Pascal Mérindol, Jean-Jacques Pansiot, Stéphane Cateloin, *Providing Protection and Restoration with Distributed Multipath Routing*, SPECTS08, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Edinburgh, UK - June 2008 (8 pages, taux de sélection $< 50\%$).

[3] Pascal Mérindol, Jean-Jacques Pansiot, Stéphane Cateloin, *Improving Load Balancing with Multipath Routing*, ICCCN08, 17TH International Conference on Computer Communications and Networks, St. Thomas, Virgin Islands, U.S.A - august 2008 (8 pages, taux de sélection $< 25\%$).

Communications à des manifestations nationales à comité de lecture :

[4] Pascal Mérindol, Jean-Jacques Pansiot, Stéphane Cateloin, *Multiroutage par Interface d'entrée*, 8èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Algotel06 pages 69–72 Tregastel, France - mai 2006 (4 pages, taux de sélection $< 50\%$).

Bibliographie

[ABC04] D. Applegate, L. Breslau, and E. Cohen. Coping with network failures : routing strategies for optimal demand oblivious restoration. In *SIGMETRICS*, pages 270–281, New York, NY, USA, 2004. ACM.

[ABG+01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE : Extensions to RSVP for lsp tunnels. RFC 3209, IETF, 2001.

[ABKM01] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, pages 131–145, New York, NY, USA, 2001. ACM.

[AC03] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands : understanding fundamental tradeoffs. In *SIGCOMM*, pages 313–324, New York, NY, USA, 2003. ACM.

[ADF+01] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP Specification. RFC 3036, IETF, January 2001.

[AFT07] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *IMC '07*, pages 149–160, New York, NY, USA, 2007. ACM.

[AMA+99] D. O. Awduche, J. Malcom, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls. RFC 2702, IETF, 1999.

[AMS+03] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *SIGCOMM*, pages 353–364, New York, NY, USA, 2003. ACM.

[APM+04] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh. A comparison of overlay routing and multihoming route control. *SIGCOMM*, 34(4) :93–106, 2004.

[Atl06] A. Atlas. U-turn alternates for ip/ldp fast-reroute draft-atlas-ip-local-protect-uturn-03. Draft, IETF, February 2006.

[AZ07] A. Atlas and A. Zinin. Basic specification for ip fast-reroute : Loop-free alternates draft-ietf-rtgwg-ipfrr-spec-base-06. Draft, IETF, mar 2007.

[Bak95] F. Baker. Requirements for ip version 4 routers. RFC 1812, IETF, 1995.

[BBB01] H. Bettahar, A. Bouabdallah, and C. Beaujean. A parameterized distributed unicast routing algorithm with end-to-end delay constraint. In *ICT*, 2001.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, IETF, 1998.

[Bel58] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16 :87–90, 1958.

[Ber91] D. P. Bertsekas. *Linear network optimization : algorithms and codes.* MIT Press, Cambridge, MA, USA, 1991.

[BFF05] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50 milliseconds recovery upon bgp peering link failures. In *CoNEXT'05*, pages 31–42, New York, NY, USA, 2005. ACM Press.

[BFF07] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50 milliseconds recovery upon bgp peering link failures. *IEEE/ACM Transactions on Networking*, 15(5) :1123 – 1135, October 2007.

[BG87a] D. Bertsekas and R. Gallager. *Data networks.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[BG87b] D. Bertsekas and R. Gallager. *Data networks.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

[Bre95] L. Breslau. *Adaptive Source Routing of Real-Time Traffic in Integrated Services Networks.* PhD thesis, University of Southern California, December 1995.

[BS95] A. W. Brander and Mark C. Sinclair. A comparative study of k-shortest path algorithms. In *Proc. 11th UK Performance Engineering Worksh. for Computer and Telecommunications Systems*, September 1995.

[BSP06] S. Bryant, M. Shand, and S. Previdi. Ip fast reroute using not-via addresses draft-bryant-shand-ipfrr-notvia-addresses-03.txt. Draft, IETF, October 2006.

[BUQ04] O. Bonaventure, S. Uhlig, and B. Quoitin. The case for more versatile bgp route reflectors : draft-bonaventure-bgp-route-reflectors-00.txt. Draft, IETF, 2004.

[BZBH97] R. Braden, L. Zhang, S. Berson, and S. Herzog. Resource reservation protocol (RSVP). RFC 2205, IETF, September 1997.

[Cas02] R. Casellas. *Partage de Charge et Ingénierie de Trafic dans les Réseaux MPLS.* PhD thesis, Ecole Nationale Supérieur des télécommunications, ENST- INFRES Informatique et Réseaux, Paris, November 2002.

[CAT90] C. Cassandras, M. Abidi, and D. Towsley. Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communications*, pages 348–359, 1990.

[CDS99] Johnny Chen, Peter Druschel, and Devika Subramanian. A new approach to routing with dynamic metrics. In *INFOCOM*, pages 661–670, 1999.

[Che94] Y. L. Chen. Finding the k quickest simple paths in a network. *Inf. Process. Lett.*, 50(2) :89–92, 1994.

[CMPS06] M. Cha, S. Moon, C-D. Park, and A. Shaikh. Placing relay nodes for intra-domain path diversity. In *INFOCOM*, 2006.

[Col98] R. Coltun. The ospf opaque lsa option. RFC 2370, IETF, 1998.

[CSRL01] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[CWZ00] Z. Cao, Z. Wang, and E. W. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM*, pages 332–341, 2000.

[Dal02] P. Dalgaard. *Introductory Statistics with R*. Springer, 2002. ISBN 0-387-95475-9.

[Dij59] E.W. Dijkstra. A note on two problems in connection with graphs. *In Numerische Mathematik, vol. 1, pages :269-271*, 1959.

[EJLW01] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE : MPLS adaptive traffic engineering. In *INFOCOM*, pages 1300–1309, 2001.

[Epp94] D. Eppstein. Finding the k shortest paths. In *IEEE Symposium on Foundations of Computer Science*, pages 154–165, 1994.

[FB05] P. Francois and O. Bonaventure. An evaluation of ip-based fast reroute techniques. In *CoNEXT'05*, pages 244–245, New York, NY, USA, 2005. ACM Press.

[FB07] P. Francois and O. Bonaventure. Avoiding transient loops during the convergence of link-state routing protocols. *IEEE/ACM Transactions on Networking*, 15(6) :1280–1932, December 2007.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in Network*. Princeton University Press, Princeton, 1962.

[FFEB05] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving sub-second igp convergence in large ip networks. *SIGCOMM*, 35(3) :35–44, 2005.

[FHG04] S. Floyd, T. Henderson, and A. Gurtov. The newreno modification to tcp's fast recovery algorithm. RFC 3649, IETF, 2004.

[Flo03] S. Floyd. Highspeed tcp for large congestion windows. RFC 3649, IETF, 2003.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (cidr) : an address assignment and aggregation strategy. RFC 1519, IETF, 1993.

[FT00] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM*, pages 519–528, 2000.

[FT02] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. In *IEEE JSAC vol. 20*, pages 756–767, May 2002.

[FVA06] A. Farrel, J.-F. Vasseur, and A. Ayyangar. A framework for inter-domain mpls traffic engineering. RFC 4726, IETF, 2006.

[gea] The geant topology - geant network. `http://www.geant.net/server/show/nav.159`.

[GL93] F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

[GM01] L. Guo and I. Matta. The war between mice and elephants. Technical report, Boston University, Boston, MA, USA, 2001.

[GO02] Roch Guérin and Ariel Orda. Computing shortest paths for any number of hops. *IEEE/ACM Trans. Netw.*, 10(5) :613–620, 2002.

[GQX+04] D. K. Goldenberg, L. Qiuy, H. Xie, Y-R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. *SIGCOMM*, 34(4) :79–92, 2004.

[GS03] M. G. Gouda and M. Schneider. Maximizable routing metrics. *IEEE/ACM Trans. Netw.*, 11(4) :663–675, 2003.

[GZR03] I. Gojmerac, T. Ziegler, and P. Reichl. Adaptative multipath routing based on local distribution of link load information. In *Proc. 4th COST 263 International Workshop on Quality of Future Internet Services*, 2003.

[Hed88] C. Hedrick. Routing information protocol. RFC 1058, IETF, 1988.

[Hed91] C. Hedrick. An introduction to IGRP. *Ctr. Comput. Inform. Services, Lab. Comput. Sci. Res., Rutgers Univ., New Brunswick, NJ*, 1991.

[Hop00] C. Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992, IETF, November 2000.

[Hor80] G. J. Horne. Finding the K least cost paths in an acyclic activity network. *J. Operational Research Soc.*, 31 :443–448, 1980.

[ICBD04] G. Iannaccone, C-N. Chuah, S. Bhattacharrya, and Christophe Diot. Feasibility of ip restoration in a tier-1 backbone. *IEEE Networks Magazine*, March 2004.

[JAC+02] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, and N. Feldman. Constraint-based lsp setup using ldp. RFC 3213, IETF, January 2002.

[JV06] Z. Jia and Varaiya. On selection of candidate paths for proportional routing. *IEEE Transactions on Automatic Control*, 51 :707 – 712, 2006.

[KHR02] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. *SIGCOMM*, 2002.

[KKDC05] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope : Responsive yet stable traffic engineering. In *SIGCOMM*, pages 253–264, 2005.

[KKSB07] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic load balancing without packet reordering. In *SIGCOMM*, volume 37, pages 51–62, April 2007.

[KKW+03] H. Tahilramani Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi. Bananas : an evolutionary framework for explicit and multipath routing in the internet. *SIGCOMM*, 33(4) :277–288, 2003.

[KKY03] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630, IETF, 2003.

[KLS04] Murali S. Kodialam, T. V. Lakshman, and Sudipta Sengupta. A simple traffic independent scheme for enabling restoration oblivious routing of resilient connections. In *INFOCOM*, 2004.

[KMT98] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks : shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49, 1998.

[KZ89] A. Khanna and J. Zinky. The revised arpanet routing metric. In *SIGCOMM '89*, pages 45–56, New York, NY, USA, 1989. ACM.

[Law77] E. L. Lawler. Comment on a computing the k shortest paths in a graph. *Commun. ACM*, 20(8) :603–605, 1977.

[LFY07] A. Li, P. Francois, and X. Yang. On improving the efficiency and manageability of notvia. In *CoNEXT*, pages 1–12, New York, NY, USA, 2007. ACM.

[LH03] K. Lan and J. Heidemann. On the correlation of internet flow characteristics. Technical report, USC/ISI Technical Report ISI-TR-574, 2003.

[LM04] Z. Li and P. Mohapatra. Qron : Qos-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1) :29–40, 2004.

[LMJ98] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 6(5) :515–528, 1998.

[MFM+06] W. Muhlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an as-topology model that captures route diversity. In *SIGCOMM*, Pisa, Italy, September 2006.

[MH05] D. Magoni and M. Hoerdt. Internet core topology mapping and analysis. *Computer Communications*, 28(5) :494–506, march 2005.

[MIB+04] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an ip backbone. *INFOCOM*, 2004.

[Min74] E. Minieka. On computing sets of shortest paths in a graph. *Commun. ACM*, 17(6) :351–353, 1974.

[MKFV06] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. *SIGCOMM*, 36(4) :135–146, 2006.

[MLMB01] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite : Boston university representative internet topology generator (software). 2001.
`http://cs-www.bu.edu/brite`.

[Moy98] J. Moy. Ospf version 2. RFC 2178, IETF, April 1998.

[MP03] E. Martins and M. Pascoal. A new implementation of yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(2) :121-134*, 2003.

[mri] The mrinfo project.
`http://clarinet.u-strasbg.fr/~pansiot/mrinfo`.

[MRR78] J. McQuillan, I. Richer, and E. Rosen. ARPANET Routing Algorithm Improvements. BBN 3803, IETF, April 1978.

[MRR80] J. M. McQuillan, I. Richer, , and E. C. Rosen. The new routing algorithm for the arpanet. *IEEE Transactions on Communications*, pages 711–719, May 1980.

[MSM97] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[MSZ96] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. In *SIGCOMM*, pages 206–217, New York, NY, USA, 1996. ACM.

[MZK01] Z. Ma, P. Zhang, and R. Kantola. Influence of link state updating on the performance and cost of qos routing in an intranet. In *IEEE Workshop on High Performance Switching and Routing*, Dallas, Texas, USA, 2001.

[NBBB98] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474, IETF, 1998.

[NBTD07] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot. Igp link weight assignment for operational tier-1 backbones. *IEEE/ACM Trans. Netw.*, 15(4) :789–802, 2007.

[NLY+07] S. Nelakuditi, S. Lee, Y. Yu, Z-L. Zhang, and C-N. Chuah. Fast local rerouting for handling transient link failures. *IEEE/ACM Trans. Netw.*, 15(2) :359–372, 2007.

[NS99] P. Narvaez and K. Y. Siu. Efficient algorithms for multi-path link state routing. In *ISCOM'99*, Kaohsiung, Taiwan, 1999.

[ns2] The network simulator- ns2.
`http://www.isi.edu/nsnam/ns`.

[NST00] P. Narváez, K-Y. Siu, and H-Y. Tzeng. New dynamic algorithms for shortest path tree computation. *IEEE/ACM Trans. Netw.*, 8(6) :734–746, 2000.

[NZ01] S. Nelakuditi and Z.-L. Zhang. On selection of paths for multipath routing. In *Proceedings of IWQoS*, 2001.

[NZD04] S. Nelakuditi, Z-L. Zhang, and D. H. C. Du. On selection of candidate paths for proportional routing. *Comput. Netw.*, 44(1) :79–102, 2004.

[oim] Multicast maps.
`http://clarinet.u-strasbg.fr/~merindol/maps.tar.gz`.

[OR07] S. Oueslati and J. Roberts. Comparing flow-aware and flow-oblivious adaptive routing. In *CISS*, 2007.

[Ora01] D. Oran. Is-is intra-domain routing protocol. RFC 1142, IETF, February 2001.

[paj] Pajek - program for large network analysis.
`http://pajek.imfm.si/doku.php?id=pajek`.

[Pal01] H. S. Palakurthi. Study of multipath routing for qos provisioning. *manuscrit non publié*, October 2001.
`http://vega.icu.ac.kr/~gmlee/research/index.html`.

[Pan07] J-J. Pansiot. Local and dynamic analysis of internet multicast router topology. *Annals of telecommunications*, 62(3-4), 2007.

[pat] A linux tcp implementation for ns2.
`http://www.cs.caltech.edu/~weixl/technical/ns2linux-2.29-linux-2.6.16/`.

[Pax97] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5) :601–615, 1997.

[PB03] C. Pelsser and O. Bonaventure. RSVP-TE extensions for interdomain LSPs : draft-pelsser-rsvp-te-interdomain-lsp-00.txt. Draft, IETF, 2003.

[Pep99] I. Pepelnjak. *EIGRP Network Design Solutions*. Cisco Press, 1999.

[PG98] J.-J. Pansiot and D. Grad. On routes and multicast trees in the internet. *ACM Computer Communication Review*, 28(1) :41–50, january 1998.

[Pos81] J. Postel. Internet protocol. RFC 791, IETF, 1981.

[PTB+02] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. A pragmatic definition of elephants in internet backbone traffic. In *IMW '02*, pages 175–176, New York, NY, USA, 2002. ACM.

[QUP+03] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with bgp. *IEEE Communications Magazine Internet Technology Series*, 41(5) :122–128, May 2003.

[RB98] N. S. V. Rao and S. G. Batsell. Qos routing via multiple paths using bandwidth reservation. In *INFOCOM*, pages 11–18, 1998.

[RKSB06] H. Rahul, M. Kasbekar, R. Sitaraman, and A. Berger. Towards realizing the performance and availability benefits of a global overlay network. *In Proc. of Passive and Active Measurement Conference*, 2006.

[RL93] Y. Rekhter and T. Li. An architecture for ip address allocation with cidr. RFC 1518, IETF, 1993.

[RL95] Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). RFC 1771, IETF, 1995.

[RSB+03] S. Roy, D. Saha, S. Bandyopadhyay, T. Ueda, and S. Tanaka. A network-aware mac and routing protocol for effective load balancing in ad hoc wireless networks with directional antenna. In *MobiHoc*, pages 88–97, New York, NY, USA, 2003. ACM.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching (mpls). RFC 3031, IETF, January 2001.

[RX05] I. Rhee and L. Xu. Cubic : a new tcp-friendly high-speed TCP variant. *Proceedings of 3rd Workshop on Protocols for Fast Long Distance Networks*, 2005.

[SAA+99] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour : Informed internet routing and transport. *IEEE Micro*, 19(1) :50–59, 1999.

[SB08] M. Shand and S. Bryant. Ip fast reroute framework draft-ietf-rtgwg-ipfrr-framework-08.txt. Draft, IETF, February 2008.

[SG01] A. Shaikh and A. Greenberg. Experience in black-box ospf measurement. In *IMW '01*, pages 113–125, New York, NY, USA, 2001. ACM Press.

[SGD05] A. Sridharan, R. Guérin, and C. Diot. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. *IEEE/ACM Trans. Netw.*, 13(2) :234–247, 2005.

[SH03] V. Sharma and F. Hellstrand. Framework for multi-protocol label switching (mpls)-based recovery. RFC 3469, IETF, 2003.

[Shi76] Douglas R. Shier. Algorithms for finding the k shortest paths in a network. In *ORSA/TIMS Joint National Mtg.*, page 115, 1976.

[ski] Skitter project (caida). `http://www.caida.org/tools/measurement/skitter/`.

[SKS01] A. Schmid, Olaf Kandel, and Christoph Steigner. Avoiding counting to infinity in distance vector routing. In *ICN '01*, pages 657–672, London, UK, 2001. Springer-Verlag.

[SMW02] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *SIGCOMM*, 2002.

[SRS99] A. Shaikh, J. Rexford, and K. G. Shin. Load-sensitive routing of long-lived ip flows. *SIGCOMM Computer Communication Review*, 29(4) :215–226, 1999.

[Suu74] J. W. Suurballe. Disjoint paths in a network. *Networks, pp. 125–45*, 1974.

[TGRR05] R. Teixeira, T. G. Griffin, M. G. C. Resende, and J. Rexford. Tie breaking : tunable interdomain egress selection. In *CoNEXT*, pages 93–104, New York, NY, USA, 2005. ACM.

[TMSV03] R. Teixera, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in isp network. In *ACM IMC '03*, October 2003.

[tot] The totem project - toolbox for traffic engineering methods. `http://totem.info.ucl.ac.be/`.

[UQBL06] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *SIGCOMM*, 36(1), January 2006.

[VGLA99] S. Vutukury and J-J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *SIGCOMM*, pages 227–238, New York, NY, USA, 1999. ACM.

[VGLA01] S. Vutukury and J-J. Garcia-Luna-Aceves. Mdva : A distance-vector multipath routing protocol. In *INFOCOM*, pages 557–564, 2001.

[Vil99a] C. Villamizar. Mpls optimized multipath (mpls-omp) : draft-villamizar-mpls-omp-01.txt. Draft, IETF, February 1999.

[Vil99b] C. Villamizar. Ospf optimized multipath (ospf-omp) : draft-ietf-ospf-omp-02.txt. Draft, IETF, February 1999.

[Vut01] S. Vutukury. *Multipath Routing Mechanisms for Traffic Engineering and Quality of Service in the Internet*. PhD thesis, University of California, Santa Cruz, 2001.

[WC90] Z. Wang and J. Crowcroft. Shortest path first with emergency exits. *SIGCOMM*, 20(4) :166–176, 1990.

[WC92] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *SIGCOMM*, 22 :63–71, 1992.

[WC96] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal of Selected Areas in Communications*, 14(7) :1228–1234, 1996.

[WWZ01] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *INFOCOM*, 2001.

[WXQ+06] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. Cope : traffic engineering in dynamic networks. In *SIGCOMM*, pages 99–110, New York, NY, USA, 2006. ACM.

[WYL+07] H. Wang, Y. Richard Yang, P. H. Liu, J. Wang, A. Gerber, and A. Greenberg. Reliability as an interdomain service. In *SIGCOMM*, pages 229–240, New York, NY, USA, 2007. ACM.

[Yen71] J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science, 17(11) :712–716*, 1971.

[YW06] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *SIGCOMM*, volume 36, pages 159–170, october 2006.

[ZDA07] Y. Zhu, C. Dovrolis, and M. H. Ammar. Combining multihoming with overlay routing (or, how to be a better isp without owning a network). In *INFOCOM*, pages 839–847, 2007.

[Zeg96] E. W. Zegura. Gt-itm : Georgia tech internetwork topology models (software). 1996. `http://www.cc.gatech.edu/fac/ellen.zegura/gt-itm/gt-itm/tar.gz`.

[ZKT+05] C. Zhanga, J. Kurosea, D. Towsley, Z. Ge, and Y. Liu. Optimal routing with multiple traffic matrices tradeoff between average and worst case performance. In *ICNP*, pages 215–224, Washington, DC, USA, 2005. IEEE Computer Society.

[ZLG+05] C. Zhang, Y. Liu, W. Gong, J. F. Kurose, R. Moll, and D. F. Towsley. On optimal routing with multiple traffic matrices. In *INFOCOM*, pages 607–618, 2005.

Notations et glossaire

Notations

Notations	Définitions
$e = e.x, e.y$ $e^{-1} = e.y, e.x$	Arc $e \in E$ reliant le nœud x au nœud y, e^{-1} désigne l'arc d'orientation opposée.
$k^-(x)$, $k^+(x)$	Degrés entrant et sortant du nœud x.
$succ(x)$	Ensemble des voisins sortants de x $y \in succ(x)$ s'il existe un arc $e = e.x, e.y \in E$.
$pred(x)$	Ensemble des voisins entrants de x $y \in succ(x)$ s'il existe un arc $e = e.y, e.x \in E$.
$c(e)$	Capacité du lien correspondant à l'arc e.
$d(e)$	Délai de propagation du lien correspondant à l'arc e.
$\{C(s,d)\}$ $\{P(s,d)\}$	$\{C(s,d)\}$ correspond à l'ensemble des coûts calculés selon la valuation des arcs entre une racine s et une destination d. $\{P(s,d)\}$ désigne l'ensemble des chemins associés à ces coûts.
$NH_1(s,d)$	Meilleur prochain saut (*next hop*). $NH_1(s,d)$ est l'extrémité sortante du premier arc $\{s, e_1.y\}$ du meilleur chemin $P_1(s,d)$.

TAB. 4.5 – Notations génériques

Notations	Définitions
$P_j(s,d) =$ $(e_1,...,e_m)$	j^{eme} meilleur chemins reliant s à d. Récursivement, il s'agit du meilleur chemin dont le premier arc est différent du premier arc de $P_l(s,d)$ pour tout l tel que $1 \leq l < j$.
$C_j(s,d) =$ $\sum_{i=1}^{m} w(e_i)$	j^{eme} meilleur coût calculé par s vers d, c'est le coût du chemin $P_j(s,d)$ avec $(1 \leq j \leq k^+(s)), (0 < m < \|N\|)$.
$NH_j(s,d)$	j^{eme} meilleur prochain saut calculé par s vers d. Il s'agit du premier saut du chemin $P_j(s,d)$.
$NH(p,s,d)$	Ensemble des prochains sauts activés par le routeur s pour les paquets provenant du routeur p en entrée vers la destination d.
$NHE(s,r,d)$	Cet ensemble désigne les prochains sauts activés sur le routeur r pour le couple *Ingress/Egress* (s,d).
$NHE_i(s,r,d)$	Elément de l'ensemble $NHE(s,r,d)$ d'indice i, il s'agit d'un prochain saut sur r utilisé par le couple (s,d).
$(p,n,d)_s$ $n \in NH(p,s,d)$	Ligne de routage activé par le routeur s pour l'interface entrante p vers la destination d avec pour prochain saut le routeur n.
$F_s(y)$	Fonction *père*, $F_s(y) = x$ si $\{x,y\} \in P_1(s,y)$.
$f(x,d)$	Fonction *fils*, il s'agit du *fils* de x sur le chemin $P_1(x,d)$. $y = f_s(x,d)$ si $\{x,y\} \in P_1(x,d)$.
$F^i,\ 0 \leq i < \|N\|$	Composition i sauts en arrière de la fonction père. Un nœud $n = F_s^i(a) = F_s^j(b),\ a,b \in N$, tel que $\sum i+j$ soit minimal désigne l'ancêtre commun le moins éloigné des nœuds a et b.
$primaire$	Un chemin $P_1(s,d)$ ou un prochain saut $NH_1(s,d)$.
$alternatif$	Un chemin $P_j(s,d)$ ou un prochain saut $NH_j(s,d)$ tel que $j > 1$.

TAB. 4.6 – Notations spécifiques

Termes	Définitions
branche $branch_h(s), h \in succ(s)$	Sous-arbre enraciné en h regroupant l'ensemble des chemins $P_1(s,d),\ \forall d \in N$ ayant un premier arc commun $e_1 = \{s,h\}$.
arc transverse	Un arc e est transverse si $\exists q \neq h \in N$ tel que $e.x \in branch_h(s) \wedge e.y \in branch_q(s)$ ou si $e.x = s$ et $e \notin P_1(s,d),\ \forall d \in N$
arc interne	Un arc e est interne s'il connecte deux nœuds $e.x$ et $e.y$ d'une même branche $branch_h(s)(e.x, e.y \in branch_h(s))$ et que $e \notin branch_h(s)$
chemin k-transverse	Un chemin est k-transverse si il contient exactement k arcs transverses et pas d'arc interne.
chemin *transverse simple* $\mathfrak{P} \in Pt(s,d)$	Un chemin 1-transverse de m sauts $(e_1, e_2, ..., e_m)$ tel que $(e_1, e_2, ..., e_{m-1}) = P_1(s, e_{m-1}.y)$ et tel que e_m soit un arc transverse $(e_m.y = d)$.
chemin *transverse retour* $\mathfrak{P} \in Pbt(s,d)$	Un chemin de m sauts $(e_1, e_2, ..., e_m)$ tel qu'il existe z $(1 < z < m)$ avec $(e_1, ..., e_z) \in Pt(s, e_z.y)$ et tel que $(e_m^{-1}, e_{m-1}^{-1}, ..., e_{z+1}^{-1}) = P_1(d, e_{z+1}.y)$.
chemin *transverse avant* $\mathfrak{P} \in Pft(s,d)$	Un chemin de m sauts $(e_1, e_2, ..., e_m)$ tel qu'il existe z $(1 < z < m)$ avec $(e_1, ..., e_z) \in Pt(s, e_z.y) \vee Pbt(s, e_z.y)$ et tel que $(e_{z+1}, e_{z+2}, ..., e_m) = P_1(e_{z+1}.x, d)$.
NH-candidat	Prochain saut obtenu via un algorithme de calcul de multichemins.
DT-voisin	NH-candidat calculé avec DT et enregistré dans la matrice Mc sous la forme d'un coût non infini.

TAB. 4.7 – Terminologie relative à DT/DT(p)

Glossaire

ie Interface d'entrée, champ utilisé pour la commutation des paquets (2.2)

AHOP All Hops Optimal Paths, problème du chemin optimal en fonction du nombre de saut (1.3.3.1)

AS Autonomous System, domaine de routage autonome (1.4)

ASBR Autonomous System Border Router, routeur de bordure de domaine (1.4.2.2)

CR-LDP Constraint Based Routing-Label Distribution Protocol, mécanisme de signalisation hard-state (1.3.2)

DAG Directed Acyclic Graph, graphe acyclique orienté (1.3.3.1)

DT Dijkstra Transverse, algorithme de calcul de chemins multiples (2.1)

DT(p) Processus de validation appliqué à la composition des prochains sauts obtenus via DT (2.2.2)

ECMP Equal Cost Multipath, protocole de routage utilisant les chemins de meilleurs coût égaux (1.2.1.2)

ER-LSP Explicitly Routed Path, mécanisme de signalisation explicite pour MPLS (1.3.2)

ERO Explicit Route Object, champ contenant les nœuds abstraits pour l'établissement de LSP (1.3.2)

FAI Fournisseur d'Accès Internet (1.4)

FEC Forwarding Equivalent Class, classe d'équivalence de routage (1.3.1)

FIB Forwading Information Base, base de donnée utilisée pour la commutation (1.2.1.2)

GWR Gateway Router, routeur de bordure de domaine, noté aussi ABR (1.4)

i-SPF incremental Shortest Path First, algorithme incrémental de calcul des meilleurs chemins (3.1)

IGP Interior Gateway Protocol, protocole de routage intradomaine (1.1.1)

IIC Incoming Interface Condition, condition pour un routage spécifique à l'interface d'entrée et sans boucle au niveau routeur (2.2.1.2)

IPFRR Internet Protocol Fast Re-routing, groupe de travail de l'IETF pour le reroutage rapide sur IP (3.1)

ISP Internet Service Provider, fournisseur d'accès Internet (1.4)

LFA Loop Free Alternate, alternative de routage uni-chemin sans boucle (1.2.4.2)

LFI Loop Free Invariant, règle de routage sans boucle (1.2.3.2)

LSA Link State Advertisement, annonce de l'état d'un lien (1.2.1.2)

LSP Labels Switch Path, chemin dont la commutation est assuré via MPLS (1.3.1)

LSR Label Switch Router, routeur avec commutation MPLS (1.3.1)

LSRR Loose Source and Record Route, routage par la source relâché (1.3.1)

LSU Link State Update, vecteur d'état des liens (1.2.3.2)

MAJ Relation de majoration basée sur les DT-voisins (2.3.1)

MPLS Multi Protocol Label Switching, commutation par étiquette (1.3.1)

NH Next Hop, prochain saut (1.2.3)

OLSA Opaque LSA, LSA contenant des informations supplémentaire de QoS (1.3.1)

QoS Quality of Service, qualité de service (1.2.2.1)

RIB Routing Information Base, base de données topologiques utilisée pour le routage (1.2.1.2)

RRO Record Route Object, champ contenant le chemin enregistré (1.3.2)

RSVP-TE Resources Reservation Protocol -Traffic Engineering, signalisation explicite en mode soft-state (1.3.2)

SDH Synchronous Digital Hierarchy, hiérarchie numérique synchrone (3.1)

SLA Service Level Agreement, niveau de service requis (3.1.5.2)

SONET Synchronous Optical Networking, standard pour les réseaux optiques synchrones (3.1.2)

SPD Source Path Deflection, conditions pour un routage sans boucle au niveau lien (1.2.3.4)

SPF Shortest Path First, algorithme de calcul des plus courts chemins (2.3.1)

SPT Shortest Path Tree, arbre des plus courts chemins (1.2.1.2)

SRLG Shared Risk Link Group, groupe de liens partageant une infrastructure physique commune (1.4.2.2)

SWP Shortest Widest Path (1.2.2.2)

TMFIB Temps maximal de mise à jour de la FIB (3.1.4)

UTURN U-Turn alternate, alternative de routage monochemin sans boucle (1.2.4.3)

WSP Widest Shortest Path (1.2.2.2)

Annexes

Modifications dans ns2

La série de fichier donnée dans le tableau ci dessous (à partir du dossier racine de ns2) synthétise les principales modifications que nous avons apporté aux modules implémentés dans ns2.

Nom du module	Description des modifications
"linkstate/ls.cc"	Calcul des multichemins et protocole de validation.
"tcl/rtglib/route-proto.tcl"	Ajout/retrait des lignes de routage par interface d'entrée.
"tcl/lib/ns-node.tcl"	Association du couple (prochain saut, destination) à une interface d'entrée.
"classifier/classifier.cc"	Commutation par interface entrante et enregistrement/tri des prochains sauts.
"common/agent.cc"	Association flux et étiquette.
"common/packet.cc"	Ajout d'un champ correspondant à l'étiquette.
"tcl/lib/ns-link.tcl"	Monitoring des liens : bande passante résiduelle.

TAB. 4.8 – Modules modifiées dans ns2

Les fichiers "tcl/rtglib/ns-rtProtoLS.tcl", "tcl/lib/ns-lib.tcl" et "linkstate/rtProtoLS.cc" (et d'autres pour des modifications mineures) ont été modifiés pour gérer l'agencement entre les différentes modifications citées dans le tableau.

Figures en couleurs

Le lecteur est invité à consulter la version électronique du document afin de pouvoir observer les deux figures en couleur relatives au temps de convergence sur Renater.

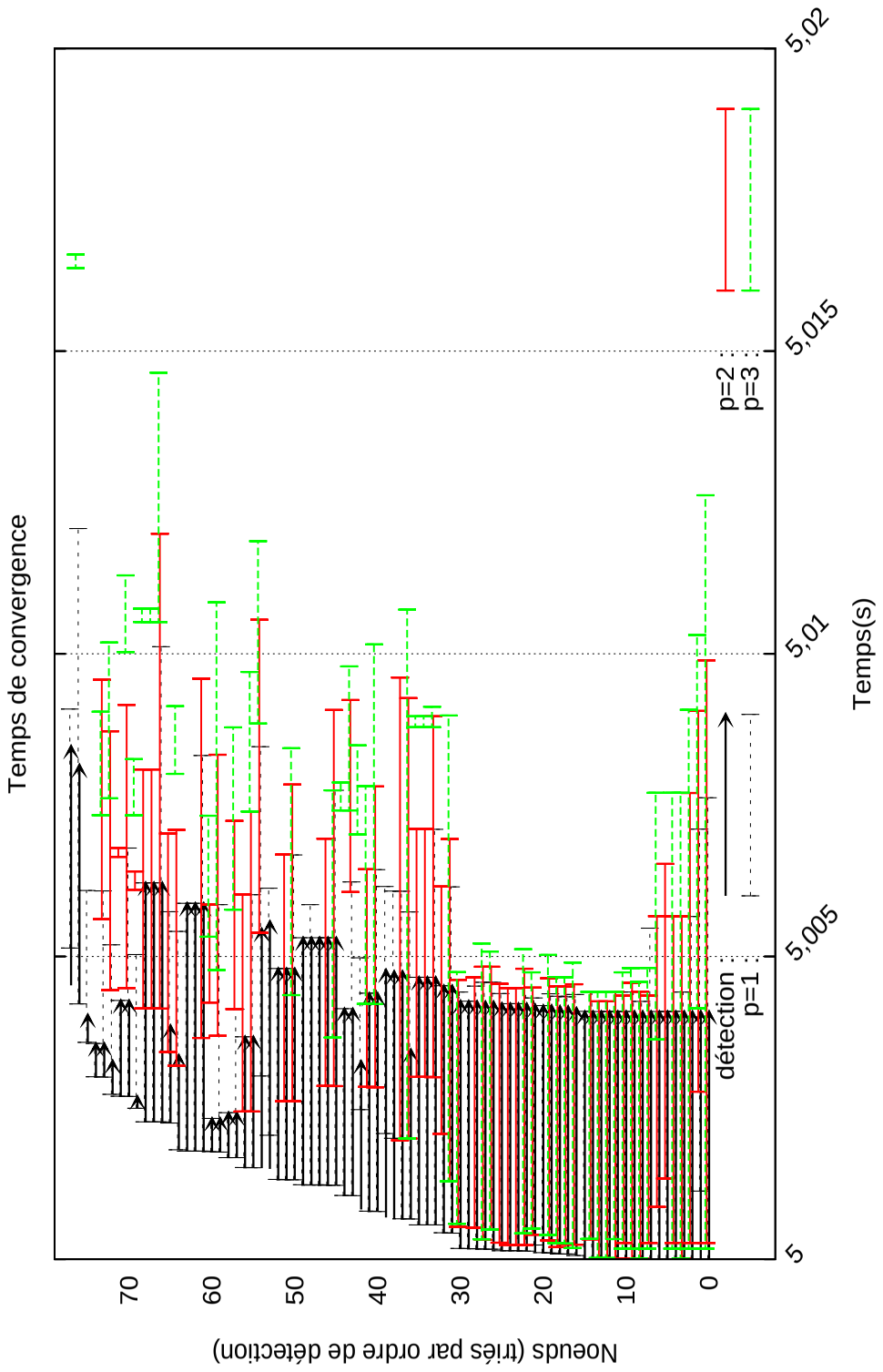


FIG. 4.24 – Temps de convergence - panne du lien entre PARIS et LYON

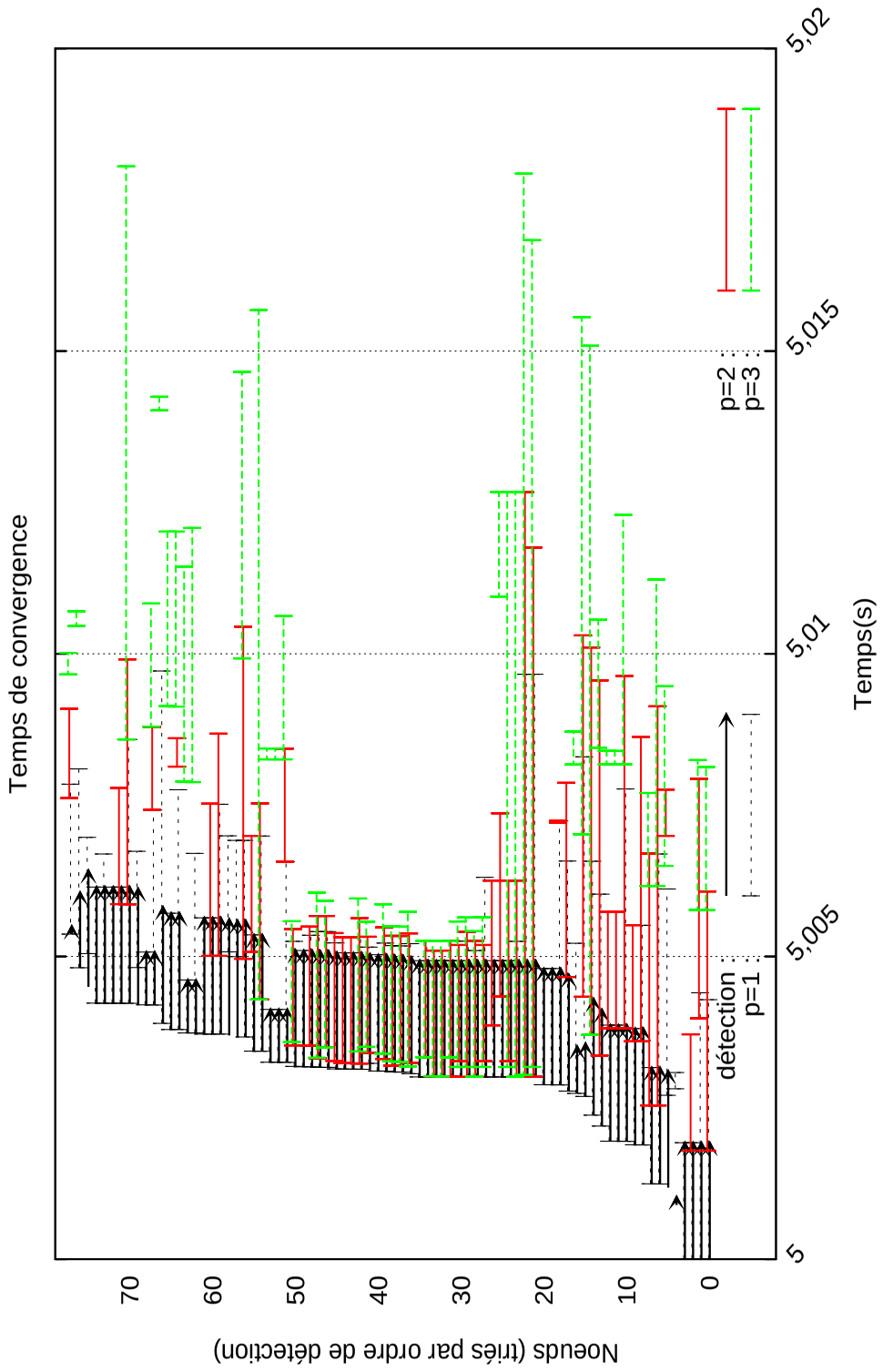


FIG. 4.25 – Temps de convergence - panne du lien entre BORDEAUX et TOULOUSE