

REPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE

UNIVERSITE DE DSCHANG

ECOLE DOCTORALE



REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND

UNIVERSITY OF DSCHANG

POST GRADUATE SCHOOL

DSCHANG SCHOOL OF SCIENCE AND TECHNOLOGY

UNITE DE RECHERCHE EN INFORMATIQUE FONDAMENTALE, INGENIERIE ET
APPLICATIONS (URIFIA)

SUJET :

GESTION AUTONOME DE LA QUALITE DE SERVICE DANS LES RESEAUX VIRTUELS

Thèse rédigée en vue de l'obtention du Diplôme de Doctorat/PhD en
Informatique

Option : Réseaux et services distribués

Spécialité : Réseaux virtuels / ingénierie de trafic

Par

YANKAM Yannick Florian

Matricule : CM04-10SCI1050
Master of Sciences en Informatique

Sous la direction de :

MYOUPO Jean Frédéric

Professeur,

Université de Picardie Jules Verne, Amiens, France

Et la co-direction de :

KENGNE TCHENDJI Vianney

*Chargé de Cours,
Université de Dschang, Cameroun*

2020

CERTIFICATION

Je soussigné, **YANKAM Yannick Florian**, matricule **CM04-10SCI1050**, certifie que, sauf remerciements raisonnables donnés, le travail dans cette thèse est le résultat de mes recherches personnelles menées au sein de l'Unité de Recherche en Informatique Fondamentale, Ingénierie et Applications (URIFIA) du Département de Mathématiques et Informatique de la Faculté des Sciences de l'Université de Dschang, sous la direction de **Pr MYOUPO Jean Frédéric**, Université de Picardie Jules Verne, Amiens, France et la codirection de **Dr KENGNE TCHENDJI Vianney**, Chargé de Cours, Université de Dschang (Faculté des Sciences). Ce travail n'a pas été soumis précédemment, en tout ou partie, pour être qualifié pour une autre diplomation universitaire.

Ce travail n'a pas été soumis précédemment, en tout ou partie, pour être qualifié pour une autre diplomation universitaire.

YANKAM Yannick Florian



Dr KENGNE TCHENDJI Vianney

Pr MYOUPO Jean Frédéric

DÉDICACES

*Au Dieu tout Puissant pour la santé et l'inspiration qu'il m'a donné pendant cette
thèse*

*À mes parents papa MBATCHOU Robert, maman TCHOUBWENG Lydienne et
maman NJAMI Elisabeth.*

REMERCIEMENTS

Cette thèse est le résultat de trois ans et demi de travail au cours desquels j'ai été accompagné et soutenu par de nombreuses personnes. C'est un plaisir d'avoir maintenant l'occasion d'exprimer ma gratitude à tous.

Mes sincères remerciements et ma reconnaissance à mes remarquables directeurs de thèse, Monsieur le Professeur MYOUPO Jean Frédéric et Monsieur le Docteur KENGNE TCHENDJI Vianney, qui m'ont non seulement guidé et encouragé tout au long de ma thèse, mais m'ont aussi beaucoup influencé dans l'attitude envers ma carrière et ma vie. Ils ont tout ce que l'on peut attendre de directeurs de thèse. Je me sens privilégié d'avoir pu bénéficier d'une telle expérience.

Je remercie vivement les membres du jury qui ont accepté de juger ce travail, à qui je présente mes sentiments de haute considération auxquels je joins mon plus profond respect.

Je remercie énormément les enseignants du département de Mathématiques-Informatique de l'Université de Dschang qui ont énormément contribué à ma formation. Vos diverses expériences et connaissances ont fortement contribué à bâtir l'être scientifique que je suis aujourd'hui. Recevez l'expression de ma profonde gratitude.

Je suis profondément et éternellement redevable à mes parents papa MBAT-CHOU Robert, maman TCHOUBWENG Lydienne et maman NJAMI Elisabeth, pour leur amour sans fin, leur soutien constant et leurs encouragements incommensurables tout au long de ma vie. Je suis fier de reconnaître ici le fruit des prières généreuses et durables que vous avez adressées en ma faveur tout au long des années d'efforts vers cette thèse.

Je remercie tout particulièrement mes frères et sœurs (Salomé, Fabrice, Nadine, Raoul) pour leurs encouragements et leur soutien inconditionnel pendant ces longues années universitaires. Merci d'être toujours à mes côtés.

Merci à Monsieur NJEUBONG Jonas et son épouse NJEUBONG Yvette pour leur soutien moral. Vos multiples conseils et encouragements ont su me guider tout au long de mon parcours universitaire.

A vous tous qui de prêt ou de loin, d'une manière ou d'une autre, avez contribué à l'édification de l'être que je suis aujourd'hui. Soyez rassurés de ma reconnaissance. Notre Père qui est aux cieux saura seul vous combler bien au-delà de vos mérites et désirs.

TABLE DES MATIÈRES

Dédicaces	iii
Remerciements	iv
Table des matières	vi
Liste des acronymes	xi
Liste des tableaux	xii
Table des figures	xiii
Résumé	xvii
Abstract	xix
 Introduction générale	 1
Contexte d'étude	1
Problématique	3
Objectif	4
Contributions	4
Structure de la thèse	5
 <i>Chapitre I ► Notions préliminaires et état de l'art</i>	 7
I.1 - Introduction	7
I.2 - Notion de virtualisation des réseaux	8
I.2.1 - Concept de virtualisation	8
I.2.2 - Intérêts de la virtualisation	9
I.2.3 - Propriétés de la virtualisation réseau	11
I.3 - Réseaux définis par logiciels (SDN)	14
I.3.1 - Définition	15
I.3.2 - Avantages du SDN	16
I.4 - Notion de qualité de service dans les réseaux	18
I.4.1 - Paramètres de QoS	18
I.4.2 - Modèles de QoS	19
I.5 - Pannes de nœuds et de liaisons	20
I.5.1 - Notion de panne	21
I.5.2 - Notions de capacité résiduelle et additionnelle	22

I.5.3 - Mécanismes de routage pour la tolérance aux pannes de noeuds et de liaisons	23
I.5.3.1 - Approches de rerouting proactives	24
I.5.3.2 - Approches de rerouting réactives	25
I.5.3.3 - Approches à chemins précalculés hybrides	26
I.6 - Mise à jour des tables de routage pour la tolérance aux pannes persistantes	28
I.6.1 - Formulation du problème de mise à jour	28
I.6.2 - Motivations de la mise à jour des tables de routage	31
I.6.3 - Routage en environnement distribué	32
I.6.3.1 - Protocoles de routage interne	33
I.6.3.2 - Protocoles de routage externe	38
I.6.4 - Routage en environnement centralisé	40
I.7 - Congestion du trafic	42
I.7.1 - Techniques d'approvisionnement en ressources	43
I.7.2 - Repositionnement des serveurs de services virtuels	44
I.8 - Conclusion	48
Chapitre II ► Notre méthode de rerouting par séparation de trafic	49
II.1 - Introduction	49
II.2 - Rerouting multichemins sans conflit dans les réseaux à base de commutateurs	50
II.2.1 - Illustration de notre solution utilisant la séparation de flux	50
II.2.1.1 - Exemple illustratif de la séparation de flux sans utilisation de capacité additionnelle	51
II.2.1.2 - Exemple illustratif de notre méthode de séparation de flux avec utilisation conjointe de capacité additionnelle et capacité résiduelle	52
II.2.2 - Existence des chemins de rerouting	54
II.3 - Modèle mathématique	57
II.3.1 - Description du modèle mathématique	57
II.3.2 - Convexité du modèle	61
II.3.3 - Heuristiques	62
II.3.3.1 - Heuristique 1 : sans gestion des conflits	62
II.3.3.2 - Heuristique 2 : avec gestion de conflit	65
II.4 - Analyse et interprétation des résultats de simulations	66
II.4.1 - Taux de restauration des pannes	66
II.4.2 - Coûts de restauration des pannes	70

II.5 - Conclusion	72
<i>Chapitre III ► Gestion de la QoS par mise à jour des tables de routage dans le cas des pannes persistantes</i>	74
III.1 - Introduction	74
III.2 - Réflexions sur quelques mécanismes de détection des nœuds à mettre à jour	75
III.2.1 - Mécanisme N°1 : Redirection de la totalité du flux sur le pont	76
III.2.2 - Mécanisme N°2 : Recherche de ponts secondaires à partir des fils directs de l'extrémité du pont dans la partie rouge G_r	77
III.2.3 - Mécanisme N°3 : Recherche de ponts secondaires à partir des fils de l'extrémité du pont dans la partie rouge G_r	78
III.2.4 - Mécanisme N°4 : Exploration à partir du nœud frontière N_f	80
III.2.5 - Mécanisme N°5 : Elimination du critère d'arrêt régissant la recherche des nœuds frontières dans le mécanisme N°4	81
III.3 - Notre stratégie de mise à jour de tables de routage en environnement centralisé	83
III.3.1 - Généralisation de notre approche de détection des nœuds critiques	84
III.3.2 - Principe général de mise à jour	85
III.3.3 - Mise à jour interne de proche en proche des nœuds du chemin de reroutage	86
III.3.4 - Reconstruction des tables de reroutage en fonction des nouvelles configurations de routage	89
III.3.5 - Intérêts de notre méthode	90
III.4 - Extension à la gestion des pannes multiples persistantes	91
III.4.1 - Cas de pannes de liaisons multiples non simultanées et persistantes	91
III.4.1.1 - Première solution : mise à jour consécutive des nœuds au fur et à mesure de la détection de la persistance des pannes	92
III.4.1.2 - Deuxième solution : mise à jour synchronisée des tables de routage des nœuds au fur et à mesure de la détection des pannes de liaisons persistantes	93
III.4.2 - Cas de la panne persistante d'un nœud	96
III.4.2.1 - Panne persistante d'un nœud virtuel	96
III.4.2.2 - Panne persistante d'un nœud physique	97

III.5 - Résultats de simulations	99
III.5.1 - Cas de panne d'une seule liaison	100
III.5.1.1 - Délai de transit des paquets	100
III.5.1.2 - Analyse du taux de perte de paquets	104
III.5.2 - Cas de panne persistante de plusieurs liaisons	106
III.5.2.1 - Délai de transit des paquets	106
III.5.2.2 - Taux de perte de paquets dans un réseau de grande taille	108
III.6 - Conclusion	109
<i>Chapitre IV ► Gestion de congestions par repositionnement dynamique des serveurs de services virtuels</i>	111
IV.1 - Introduction	111
IV.2 - Méthode de Horiuchi et Tachibana complétée	112
IV.2.1 - Traitement du cas de l'égalité des poids de flux	112
IV.2.2 - Réduction des repositionnements	114
IV.2.3 - Mécanisme de migration des données	115
IV.3 - Technique de séparation du trafic de migration	117
IV.3.1 - Repositionnement de serveurs par séparation de flux centrée sur la bande passante : FSB-DReViSeR-bandwidth	119
IV.3.1.1 - Hypothèses et notations	119
IV.3.1.2 - Description de la méthode	119
IV.3.2 - Repositionnement de serveurs par séparation de flux centré sur le débit des liaisons : FSB-DReViSeR-throughput	124
IV.3.2.1 - Difficulté du choix de chemins de migration sur la base du débit	125
IV.3.2.2 - Description de la méthode d'exploitation du débit	125
IV.4 - Simulations et discussion	127
IV.4.1 - Performances de la méthode de Horiuchi et Tachibana améliorée	128
IV.4.1.1 - Impact du nombre de repositionnements sur la QoS	128
IV.4.1.2 - Impact du temps de migration sur la QoS	129
IV.4.2 - Performances du repositionnement par séparation de flux	132
IV.4.2.1 - Impact de la séparation de flux avec usage de Bande passante comme critère de choix de chemins	132
IV.4.2.2 - Impact de la séparation de flux avec usage du débit comme critère de choix de chemins	133

IV.4.2.3 - Impact de la séparation de flux sur la cohérence du service après repositionnement du serveur	134
IV.5 - Conclusion	135
Conclusion générale	138
Rappel de la problématique étudiée	138
Analyse critique des résultats obtenus	138
Quelques perspectives	140
Bibliographie	143
<i>Annexe A ▶ Publications issues de cette thèse</i>	151

LISTE DES ACRONYMES

API	Application Programming Interface ;
ARPANet	Advanced Research Projects Agency Network ;
AS	Autonomous System ;
BGP	Border Gateway Protocol ;
BPMN	Business Process Model and Notation ;
CPU	Central Processing Unit ;
DARPA	Defense Advanced Research Projects Agency ;
EIGRP	Enhanced Interior Gateway Routing Protocol ;
FIP	Fournisseur d'Infrastructure Physique ;
FS	Fournisseur de Service ;
GNU	General Public License ;
ILP	Integer Linear Program ;
IP	Internet Protocol ;
IPFRR	Internet Protocol Fast ReRoute ;
LAN	Local Area Network ;
MAC	Media Access Control ;
MPLS	Multi-Protocol Label Switching ;
MTU	Maximum Transmission Unit ;
MV	Machine Virtuelle ;
NED	Network Descriptor ;
OSPF	Open Shortest Path First ;
QoS	Quality of Service ;
SDN	Software Defined Networking ;
TCP	Transmission Control Protocol .

LISTE DES TABLEAUX

I - Différentes notations utilisées dans notre modélisation.	58
II - Différentes notations utilisées dans les heuristiques.	63
III - Taux de restauration des pannes avec gestion de conflits.	69
IV - Taux de restauration des pannes sans gestion de conflits.	69
V - Coût de restauration pour pannes simples.	70
VI - Comparaison de capacité additionnelle utilisée pour différentes techniques.	71
VII - Différentes notations utilisées dans FSB-DReViSeR.	119

TABLE DES FIGURES

1 - Trois machines virtuelles dans une machine physique.	8
2 - Environnement de virtualisation réseau (Chowdhury & Boutaba, 2009).	12
3 - Architecture réseaux traditionnels vs réseaux SDN (Nunes, Mendonca, Nguyen, Obraczka, & Turletti, 2014).	16
4 - Les différents arcs d'une liaison (A-B).	21
5 - Différents types de capacité.	23
6 - Illustration du conflit de routage.	25
7 - Méthode de reroutage de Son (2014) : Cas de pannes simples (Son, 2014).	28
8 - Le problème de mise à jour des tables de routage.	29
9 - Génération de cycle dans la mise à jour des tables de routage.	30
10 - Structure d'un message RIP.	34
11 - Exemple de message de routage RIP pour plusieurs destinations (G. Malkin, 1998).	34
12 - Mauvaise politique de choix de chemin par RIP.	35
13 - Structure d'une entrée de table de routage OSPF.	36
14 - Structure d'un LSU (J. Moy, 1997).	37
15 - Structure d'un UPDATE message du protocole BGP (Rekhter, Li, & Hares, 2006a).	39
16 - Structure d'un message RANN (Lim, Wang, Kado, & Zhang, 2008).	40
17 - Structure d'un message RSET (Lim et al., 2008).	41
18 - Technique de mise à jour proposée par Son (2014) : Cas de pannes simples (Son, 2014).	43
19 - Repositionnement de serveur virtuel selon Koyanagi et Tachibana (2014).	45
20 - Repositionnement de serveur virtuel selon Horiuchi et Tachibana (2018).	46
21 - Problème d'égalité de poids de trafic dans la méthode de Horiuchi et Tachibana (2018).	46
22 - Reroutage sans capacité additionnelle.	51
23 - Réseau avec capacité résiduelle et additionnelle.	53
24 - Reroutage de Son (2014) vs notre méthode.	54

25 - Preuve d'existence de plusieurs chemins de reroutage.	55
26 - Preuve par récurrence	56
27 - Résolution de conflit.	65
28 - Quelques réseaux de tests.	68
29 - Courbe comparative des restaurations avec conflits et sans conflits. . .	69
30 - Coûts de restauration de différentes méthodes.	71
31 - Capacité additionnelle utilisée par les différentes méthodes.	73
32 - Mécanisme N°1.	77
33 - Mécanisme de recherche N°2.	79
34 - Mécanisme N°3.	80
35 - Mécanisme de recherche N°4.	82
36 - Mécanisme N°5.	83
37 - Notre stratégie généralisée de détection des nœuds à mettre à jour. . .	84
38 - Structure du réseau après mise à jour.	86
39 - Vecteur de liste de triplets de mise à jour.	87
40 - Mise à jour d'un nœud à un autre avec le vecteur de triplet.	88
41 - Fragmentation du message de mise à jour.	89
42 - Première solution : pannes de liaison multiple non simultanées persistantes.	92
43 - Panne du nœud 14.	96
44 - Panne d'un nœud physique.	98
45 - Structure du réseau 1.	100
46 - Arbre nominal avec pour destination le nœud[3] dans le reseau1. . .	101
47 - Variations des délais de transit des paquets du nœud[3] dans le reseau1. .	101
48 - Variations des délais d'acheminement des paquets du nœud[4] dans le reseau1.	102
49 - Variations des délais d'acheminement des paquets du nœud[1] dans le reseau1.	103
50 - Comportement du reseau2 en présence de panne persistante de liaison. .	103
51 - Variations des délais de transit des paquets pour le nœud[4] du reseau2. .	104
52 - Variations des délais de transit des paquets pour le nœud[4] du reseau4. .	105
53 - Comparaison du taux de perte de paquets de notre stratégie de mise à jour des tables de routage avec celle de Son (2014) dans le reseau1. . .	105
54 - Aspect du reseau2 en présence de la panne du nœud[3].	106
55 - Variations des délais d'acheminement des paquets du nœud[4] du reseau2 dans le cas de panne du nœud[3].	107

56 - Variations des délais d'acheminement des paquets du nœud[4] du réseau4 dans le cas de panne du nœud[3].	108
57 - Taux de perte de paquets de notre stratégie de mise à jour des tables de routage vs la méthode de Son (2014) vs le protocole TBR pour le cas de panne de nœud dans le reseau4.	109
58 - Solution pour le problème de poids de trafic égaux dans les nœuds terminaux avec des parents de poids de trafic différents.	113
59 - Solution pour le problème du poids de trafic égaux dans les nœuds feuilles avec une hauteur d'arbre $h = 1$	114
60 - Algorithme de pré-copie (Kherbache, 2016).	116
61 - Algorithme de post-copie (Kherbache, 2016).	116
62 - Algorithme de post-copie hybride (Kherbache, 2016).	117
63 - Mauvais choix de chemin avec (Liu, Li, & Jin, 2014) dans le cas de liaisons avec bandes passantes égales.	120
64 - Mauvais choix de chemin avec (Liu et al., 2014) dans le cas de liaisons avec bandes passantes différentes.	121
65 - Sélection des chemins de migration par séparation de flux dans le cas de chemins avec bandes passantes égales.	122
66 - Sélection des chemins de migration par séparation de flux dans le cas de chemins avec bandes passantes différentes.	123
67 - Sélection des chemins de migration par séparation de flux sur la base du débit de données des liaisons	127
68 - Taux de repositionnement pour un réseau de 20 nœuds.	129
69 - Taux de repositionnement pour un réseau de 60 nœuds.	129
70 - Variation du poids de trafic pour le réseau de 20 nœuds.	130
71 - Variation du poids de trafic pour le réseau de 60 nœuds.	130
72 - Variation du temps de migration pour le réseau de 20 nœuds.	131
73 - Variation du temps de migration pour le réseau de 60 nœuds.	131
74 - Temps de repositionnement pour le réseau de 20 nœuds.	132
75 - Temps de repositionnement pour le réseau de 60 nœuds.	133
76 - Temps moyen de migration vs taux de migration pour le réseau de 20 nœuds.	134
77 - Temps moyen de migration vs taux de migration pour le réseau de 60 nœuds.	135
78 - Temps moyen de migration vs taux de migration pour le réseau de 20 nœuds.	136

79 - Temps moyen de migration vs taux de migration pour le réseau de 60 nœuds.	136
--	-----

RÉSUMÉ

Pendant plusieurs années, internet a offert aux utilisateurs de nombreux services de plus en plus utiles et à forte valeur ajoutée pour les entreprises. Mais, avec l'évolution sans cesse variée des besoins des utilisateurs, l'infrastructure de base d'internet ne permettait plus de satisfaire convenablement ces derniers. La virtualisation réseau a été proposée et adoptée comme solution très prometteuse pour répondre à ces besoins.

Face aux difficultés de gestion de trafic dans les infrastructures de réseaux virtuels en adéquation avec les ressources mises à leur disposition, nous proposons dans le cadre de cette thèse, la séparation de flux dont l'idée principale est de subdiviser un flux original en plusieurs sous flux afin de faciliter son acheminement. Dans la première contribution de cette thèse, nous l'appliquons à la prise en charge de la panne temporaire d'une liaison virtuelle, en modélisant une stratégie de reroutage multichemins à économie de ressources basée sur un contrôleur SDN (Software-Defined Network).

La deuxième contribution résoud le problème de la persistance des pannes de liaisons et de nœuds en définissant des approches de mises à jour des règles de routage et reroutage dans les nœuds à partir d'un contrôleur.

L'activité des utilisateurs causant des variations de trafic conduisant souvent à des congestions au sein des serveurs de services virtuels, dans la troisième contribution, nous proposons deux stratégies de repositionnement de ces serveurs vers des nœuds offrant les ressources nécessaires pour garantir la QoS : "FSB-DReViSeR bandwidth" et "FSB-DReViSeR throughput".

Les simulations réalisées permettent de se rendre compte que les politiques de gestion de trafic par séparation de flux proposées, réduisent les taux de perte de paquets, les délais de transit des paquets et la gigue qui sont des éléments très importants d'appréciation de la QoS dans les réseaux.

Mots clés : Virtualisation réseau, Qualité de service, Réseaux définis par logiciels, Séparation de flux, Panne persistante, Repositionnement de serveur.

Autonomous management of quality of service in virtual networks

ABSTRACT

For several years, the internet has offered many increasingly useful services to the users, with high added value for businesses. But, with the ever-changing needs of users, the basic internet infrastructure no longer satisfied them adequately. Network virtualization has been proposed and adopted as a very promising solution to meet these needs.

Dealing with the traffic management challenges in virtual network infrastructures (VNI) in line with the available resources, we propose in this thesis, the flow splitting approach whose main idea is to split an original flow into several flowlets (sub-streams) in order to facilitate routing. In the first contribution of this thesis, we apply flow splitting to a virtual link failure recovery scenario, by modeling a resource-saving multi-path rerouting scheme based on a SDN (Software Defined Networking) controller.

The second contribution solves the persistent links and nodes failures challenge, by defining routing tables' updates methods for network nodes from a controller. While the user activity causes some traffic variations in the network and often leads to service congestion within the servers, we propose two main server replacement strategies towards the nodes offering enough resources to satisfy QoS : "FSB-DReViSeR bandwidth" and "FSB-DReViSeR throughput".

The simulations carried out reveal that the proposed flow splitting-based traffic management policies, help to reduce the data loss rate, the packet transit delays and the jitter which are key QoS appreciation metrics in computers networks.

Keywords : Network virtualization, Quality of service, Software defined networks, Flow splitting, Persistent failure, Server replacement.

INTRODUCTION GÉNÉRALE

SOMMAIRE

Contexte d'étude	1
Problématique	3
Objectif	4
Contributions	4
Structure de la thèse	5

Contexte d'étude

Mise en place par l'armée américaine (sous couvert de la DARPA (Defense Advanced Research Projects Agency)) dans les années 1967 sous l'appellation originale d'ARPANet (Advanced Research Projects Agency Network), internet fut conçu pour offrir à l'armée et aux chercheurs une plateforme de télécommunications plus accessible et ouverte aux échanges. Avec son ouverture à une plus grande échelle au début des années 1980 et le franc succès réalisé notamment grâce à l'avènement des équipements de routage de paquets, le protocole TCP/IP (Transmission Control Protocol / Internet Protocol) et surtout le *web*, internet a rapidement vu l'adhésion d'un grand nombre d'utilisateurs à travers le monde. Cette adhésion s'est suivie au fil des années, du développement et de l'intégration de services de plus en plus complexes grâce au protocole TCP/IP, afin de satisfaire les utilisateurs finaux. C'est ainsi qu'il est devenu de plus en plus difficile d'apporter des changements notoires dans l'architecture profonde d'internet pour proposer de nouveaux paradigmes réseaux devant répondre au mieux aux nouveaux besoins des acteurs d'internet ; les seules modifications possibles ne se limitaient alors qu'à des mises à jours de protocoles existants : il s'agit du phénomène d'ossification ([Chowdhury & Boutaba, 2009](#) ; [Niebert et al., 2008](#)).

Plusieurs entreprises et organisations ont adressé le problème d'ossification de l'internet, et la *virtualisation* a été présentée comme la solution la plus prometteuse (Fernandes et al., 2011). En effet, la virtualisation est une technique permettant une meilleure utilisation des ressources grâce à leur abstraction, ce qui permet de s'affranchir des particularités de chaque équipement. L'adoption de cette technique dans les réseaux informatiques, a conduit à la virtualisation des réseaux (Chowdhury & Boutaba, 2009 ; Fernandes et al., 2011 ; Jain & Paul, 2013). La virtualisation des réseaux est un paradigme réseau dans lequel de nouvelles entités réseaux logiques appelées réseaux virtuels sont créés grâce à la mise en place et l'interconnexion de liaisons et nœuds logiques à travers une infrastructure physique (Carapinha & Jiménez, 2009 ; Chowdhury & Boutaba, 2009). Grâce à la flexibilité architecturale et le mécanisme d'isolation¹ qui confère aux réseaux virtuels une indépendance fonctionnelle vis à vis du réseau physique, la création des services adaptés et leur approvisionnement en ressources est grandement facilitée. De plus, chaque réseau virtuel peut implémenter des politiques de gestion de QoS (Quality of Service) propres en fonction des ressources disponibles. D'ailleurs, le Cloud Computing² qui est l'une des dernières grandes évolutions dans le domaine des réseaux informatiques aujourd'hui, est l'une des applications les plus visibles de la virtualisation. Afin de mieux contrôler l'ensemble des entités hétérogènes impliquées dans la virtualisation des réseaux, les fournisseurs de services (FSs) ont adopté le SDN (Software Defined Networking).

En effet, le SDN est une technologie dans laquelle le plan de contrôle³ de chaque équipement est découplé de son plan de données⁴ (Farhady et al., 2015 ; Guirlanger, 2015 ; Jain & Paul, 2013). Le SDN offre ainsi la possibilité de programmer dynamiquement le comportement des composants réseaux. A partir d'une plateforme centralisée, les politiques de gestion du réseau (QoS, adressage, routage, tolérance aux pannes, congestions, ...) peuvent être programmées et automatisées pour assurer une réaction rapide et efficace du réseau face aux mutations de trafic. Cette façon de faire permet aux FSs de mieux superviser leur infrastructure, indépendamment du nombre d'équipements en place. Les mutations de trafic peuvent être dues aux pannes de nœuds ou de liaisons (Pham et al., 2012 ; Shahriar et al.,

1. L'**isolation** est une propriété garantissant que tout problème de configuration dans un réseau virtuel est circonscrit à ce seul réseau et n'affecte pas les autres réseaux virtuels avec lesquels il coexiste (Chowdhury & Boutaba, 2009).

2. **Cloud Computing** : paradigme réseau permettant l'accès à distance à des ressources réseau importantes (stockage, CPU, bande passante) afin d'en augmenter la rentabilité (Jain & Paul, 2013).

3. Le **plan de contrôle** est la partie d'un commutateur réseau définissant les politiques de gestion des flux (Farhady, Lee, & Nakao, 2015).

4. Le **plan de données** est la partie d'un commutateur réseau responsable de l'acheminement des flux suivant les règles définies par le plan de contrôle (Farhady et al., 2015).

2016 ; Veerasamy, Venkatesan, & Shah, 1994), l'ajout ou le retrait de composants du réseau (liens, nœuds) (Horiuchi & Tachibana, 2018 ; Shahriar et al., 2016) ou alors à la mobilité et la quantité des utilisateurs par rapport à un service (Horiuchi & Tachibana, 2018). Afin d'assurer sa continuité, le trafic est souvent redirigé vers un ou plusieurs chemins de repli. Dans le cas des réseaux virtuels, ces mutations influencent les ressources fournies par le réseau physique sur lequel repose la stabilité de toute l'infrastructure virtuelle.

Le but de notre étude est d'assurer une bonne qualité de service dans les réseaux virtuels sujets aux pannes de nœuds, de liaisons et aux variations de trafic menant aux congestions. Notre thèse est que la séparation de flux constitue une bonne technique de gestion du trafic pour garantir une QoS satisfaisante dans les réseaux virtuels en cas de pannes de liaisons ou de noeuds, ou de mobilité des utilisateurs.

Problématique

Dans une infrastructure SDN, le plan de contrôle de chaque équipement est dé-localisé au sein d'un contrôleur. Ce contrôleur propose des APIs (Application Programming Interfaces) permettant aux utilisateurs de définir des règles de gestion du réseau. Ces règles de gestion sont exécutées par les nœuds du réseau, qui ne détiennent plus que le plan de données responsable de l'acheminement des données. Ainsi, lorsque des modifications doivent être effectuées dans le réseau, par exemple pour fixer les chemins de rerouting du trafic à la suite d'une panne de liaison, ces dernières sont effectuées dans le contrôleur qui peut aisément les transmettre aux nœuds concernés par ce rerouting. Cette approche implique qu'à chaque panne, une demande est adressée au contrôleur pour qu'il calcule les règles de rerouting du flux issu de la panne. Ces règles devront par la suite être acheminé jusqu'aux nœuds concernés à travers des chemins de rerouting appropriés. Mais, le temps de latence dans la prise en main de pannes est assez important, surtout lorsque la taille de l'infrastructure réseau est assez conséquente.

Pour résoudre ce problème de lenteur dans la prise en main du rerouting, l'approche par précalcul des chemins de rerouting a été proposée (Kamamura, Shima-zaki, Hiramatsu, & Nakazato, 2013 ; Son, 2014 ; Xi & Chao, 2007). Elle consiste à précalculer par le contrôleur, avant la mise en marche du réseau, l'ensemble des chemins de rerouting possibles pour tous les cas potentiels de pannes de liaisons et intégrer ces chemins au sein des nœuds. De ce fait, lorsqu'une panne de liaison se produit, l'un des nœuds composant la liaison réagit directement à la panne en appliquant les règles de rerouting précalculées pour cette panne spécifique : cette

approche est connue sous l'appellation d'IPFRR (Internet Protocol Fast ReRoute). Elle a l'avantage de réduire le temps de latence dans la prise en main du rerouting. Toutefois, lorsque la panne devient persistante, les règles de rerouting précédemment calculées deviennent obsolètes car ne proposant plus désormais des chemins de rerouting optimaux adaptés à la situation de panne. Dès lors, deux grands problèmes sont à relever : le premier problème est celui du recalcul des chemins de rerouting et la structure du message supposé porter les informations de mise à jour aux nœuds. Le second problème est celui de la politique de mise à jour des tables de routage des nœuds sans nuire au trafic courant.

Cependant, que ce soit en présence ou en absence de persistance de panne, la question d'utilisation efficiente des ressources dans le rerouting se pose. Des méthodes de rerouting à économie de ressource doivent donc être mises en avant pour éviter une mauvaise QoS due à la sous-exploitation ou la surexploitation des chemins de reroutages. En cas d'indisponibilité de ressources, une approche d'approvisionnement peut être aménagée.

Objectif

Ce travail a pour objectif principal de proposer des protocoles visant à maintenir une bonne qualité de service dans les réseaux virtuels face aux pannes. Spécifiquement, il s'agit de proposer des protocoles pour :

- rerouter judicieusement le trafic perturbé par une panne temporaire de liaison ou de nœud ;
- rendre le réseau apte à conserver une qualité de service satisfaisante face aux cas de pannes persistantes ;
- exploiter au maximum les ressources du réseau virtuel avant d'envisager un approvisionnement en ressources.

Contributions

Dans ce manuscrit, nous proposons des solutions au problème de gestion de trafic dans les réseaux virtuels en tenant compte du critère de qualité de service. Ainsi, nos contributions se déclinent en trois points :

- une stratégie implémentée démontrant que le rerouting d'un flux à travers plusieurs chemins est préférable à une solution unicemin. L'idée est de séparer le trafic en amont d'une panne de liaison ou de nœud, à travers plu-

- sieurs chemins qui consomment le minimum de bande passante possible. Un modèle mathématique mis en place, démontre la faisabilité de cette solution (Tchendji, Yankam, & Myoupo, 2018);
- une politique de mise à jour des tables de routage et de reroutage des nœuds lorsque la panne devient persistante. Dans un contexte où les chemins sont précalculés par un contrôleur avant la mise en place du réseau, la persistance d'une panne de liaison ou de nœud rend ces chemins obsolètes. L'approche consiste à recalculer les chemins de reroutage spécifiquement à la panne et à transmettre les informations de mise à jour aux nœuds sans nuire au trafic courant (Myoupo, Yankam, & Tchendji, 2018 ; Yankam, Myoupo, & Tchendji, 2019);
 - une solution de repositionnement dynamique de service virtuel permettant de pallier à la congestion d'un serveur de service virtuel. L'idée consiste à faire migrer un service virtuel d'un nœud virtuel vers un autre offrant suffisamment de ressources pour gérer la mutation de trafic (Myoupo, Yankam, & Tchendji, 2020).

Structure de la thèse

Nos travaux dans le cadre de cette thèse sont structurés en quatre chapitres.

Le chapitre 1 présente notre environnement de travail et quelques travaux liés à la tolérance aux pannes dans cet environnement. Nous commençons d'abord par les notions de virtualisation, de SDN et de QoS dans les réseaux ; un exposé sur les intérêts de ces technologies est également présenté. Ensuite, nous décrivons quelques cas de pannes rencontrés dans les réseaux virtuels ; il s'agit notamment des pannes de liaison et de nœuds, puis des congestions de trafic. Une analyse des causes liées à ces pannes y est proposée ainsi qu'un état de l'art sur les travaux effectués dans la perspective de les résoudre.

Dans le chapitre 2, nous proposons une approche de reroutage par séparation de flux pour résoudre le problème lié à la panne ponctuelle d'une liaison. Cette approche maximise l'utilisation des ressources du réseau virtuel siège de la panne et minimise ainsi la quantité de ressources additionnelles sollicitées sur le réseau physique.

Le chapitre 3 aborde le problème de persistance des pannes au sein des réseaux virtuels en proposant une approche de mise à jour des tables de routage. Nous y montrons que lorsqu'une panne de liaison ou de nœud persiste dans un réseau

programmable, le contrôleur peut recalculer les chemins de reroutage et mettre à jour les tables de routage des nœuds sans nuire significativement au trafic courant.

La gestion des congestions de trafic fait l'objet du chapitre 4. Nous proposons une approche de repositionnement d'un serveur de service virtuel. Face au manque de ressources d'un serveur, nous faisons migrer le service hébergé vers un nœud virtuel offrant davantage de ressources pour garantir une QoS satisfaisante. Cette approche s'entoure des précautions de migration garantissant la cohérence du service restauré dans un autre nœud virtuel.

Nous concluons cette thèse en dressant un bilan de nos contributions. Des perspectives liées à ce travail sont également suggérées.

I

CHAPITRE

NOTIONS PRÉLIMINAIRES ET ÉTAT DE L'ART

SOMMAIRE

I.1 - Introduction	7
I.2 - Notion de virtualisation des réseaux	8
I.3 - Réseaux définis par logiciels (SDN)	14
I.4 - Notion de qualité de service dans les réseaux	18
I.5 - Pannes de nœuds et de liaisons	20
I.6 - Mise à jour des tables de routage pour la tolérance aux pannes persistantes	28
I.7 - Congestion du trafic	42
I.8 - Conclusion	48

I.1. Introduction

Notre environnement de travail est celui des réseaux virtuels programmables, qui désignent des environnements réseaux dans lesquels les politiques de gestion des équipements et du trafic réseau sont entièrement automatisées. Dans ce chapitre, nous présentons dans un premier temps les notions et technologies rattachées à de tels environnements, ainsi que leurs caractéristiques et intérêts. Il s'agit entre autres de la virtualisation et du SDN (Software Defined Networking). Ensuite, dans un second temps, nous présentons en rapport avec cet environnement, les concepts de qualité de service, panne, capacité résiduelle et capacité additionnelle, dont la gestion fait l'objet de cette thèse. Enfin, nous passons en revue quelques travaux pertinents liés à la gestion des pannes dans des environnements réseaux similaires ou identiques au nôtre.

I.2. Notion de virtualisation des réseaux

I.2.1. Concept de virtualisation

La virtualisation est une technologie qui consiste à l'abstraction des ressources d'un équipement physique (ordinateur, routeur, switch, ...) (Carapinha & Jiménez, 2009). Autrement dit, à partir d'un équipement physique existant, l'on crée un ou plusieurs autres équipements logiques possédant les mêmes fonctions que l'original : ces équipements logiques sont appelés *machines virtuelles*. Une machine virtuelle est dotée d'un processeur, d'une mémoire principale ainsi qu'un espace de stockage au même titre que la machine physique à partir de laquelle elle a été créée. Il est alors possible d'installer également des systèmes d'exploitation dans ces machines virtuelles. La coexistence de ces systèmes d'exploitation dans l'hôte est assurée grâce à un logiciel appelé *hyperviseur*, à l'exemple de Xen, Oracle VM, VmWare workstation, virtualBox, Citrix, ... L'hyperviseur permet de créer une ou plusieurs machines virtuelles dans les limites de la quantité de ressources disponibles dans l'équipement physique. La figure 1 présente un exemple de machine physique au sein de laquelle trois machines virtuelles ont été créées.

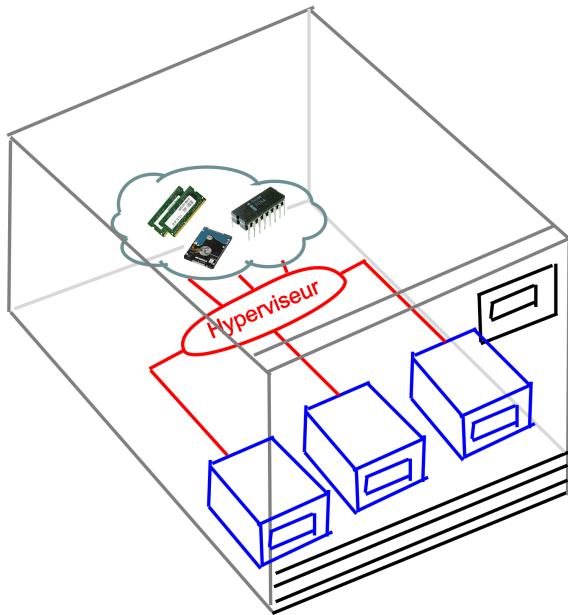


Figure 1 – Trois machines virtuelles dans une machine physique.

L'interconnexion des machines virtuelles entre elles constitue un réseau virtuel. Les réseaux virtuels peuvent couvrir une seule machine physique ou un ensemble de machines physiques qui doivent au préalable être interconnectées. La virtualisation du réseau est donc la reproduction logicielle d'un réseau physique. Les réseaux virtuels issus de cette reproduction, offrent les mêmes fonctions et garan-

ties que les réseaux physiques, les applications s'y exécutant pareillement qu'en environnement physique. Grâce à cette capacité de créer des machines virtuelles offerte par la virtualisation, les ressources d'un équipement et d'un réseau sont mieux exploitées. Mais la virtualisation permet de profiter de beaucoup d'autres avantages aussi bien sur le plan académique que professionnel.

I.2.2. Intérêts de la virtualisation

Sur le plan professionnel, la virtualisation des serveurs permet d'avoir moins de serveurs physiques, réduisant de ce fait les coûts d'investissements de l'entreprise à plusieurs niveaux (électricité, matériel, immobilier, maintenance). En effet, puisque plusieurs machines virtuelles peuvent fonctionner sur un unique serveur physique, le nombre de serveurs à acheter et à entretenir est réduit. Aujourd'hui, chaque application doit être installée sur un serveur dédié afin de gérer aisément les montées en charge ([VMware, 2016](#)). Or, comme les serveurs sont généralement trop puissants pour les besoins des applications, une grande partie des ressources de ces serveurs n'est pas utilisée. En virtualisant, plusieurs applications peuvent coexister au sein d'un même serveur physique, dont on optimise ainsi l'efficacité. La conséquence immédiate de cette réduction du nombre de serveurs physiques est la consommation réduite de l'énergie électrique ; ce qui permet à l'entreprise de réaliser de belles économies d'électricité, les dépenses dues à l'alimentation en énergie et au refroidissement des serveurs informatiques étant considérables ; d'autant plus qu'un serveur est déployé généralement pour fonctionner sans arrêt pendant une très longue période. L'évaluation des dépenses liées à l'alimentation en énergie électrique tout au long de la durée de vie d'un serveur représente une somme non négligeable. La virtualisation permet de réduire, voire de diviser les coûts en électricité.

La portabilité des machines virtuelles est un autre avantage important de la virtualisation. Un serveur virtuel n'est qu'une suite de fichiers. Il est alors possible de déplacer ces fichiers sur n'importe quel autre type de matériel sans prendre le risque d'observer des erreurs au démarrage. De plus, ces fichiers peuvent être copiés sur un espace de stockage mobile (disque dur par exemple) pour ensuite être réinstallés sur une autre machine physique. Cela permet en cas de sinistre par exemple, de relancer l'activité de l'entreprise très rapidement. Sans virtualisation, il faudrait passer par le long processus de commande du matériel, d'installation des logiciels et mises à jour nécessaires, qui peuvent prendre plusieurs semaines.

La virtualisation permet également aux entreprises de réduire l'impact environnemental de leur activité. Les machines physiques en fonctionnant, produisent du

carbone à travers la chaleur dissipée. Plus le nombre de machines physiques est important, plus les gaz sont rejetés dans l'environnement. Le remplacement des postes par des clients légers et la centralisation des serveurs contribuent ainsi à réduire le bilan carbone de l'entreprise, l'amenant à se diriger vers une politique de Green IT (informatique verte) ([Dasilva, Liu, Bessis, & Zhan, 2012](#) ; [Thomas, Costa, & Oliveira, 2016](#)). Les clients légers consomment 10 fois moins d'électricité qu'un poste standard. Ils dissipent donc moins de chaleur dans les bureaux. Dans le contexte actuel de lutte acharnée contre le réchauffement climatique, la virtualisation apparaît ainsi comme une solution salutaire dans la réduction des gaz à effet de serre dans l'atmosphère.

Grâce à la virtualisation, les services proposés par une entreprise profitent d'une meilleure sécurité. Habituellement dans une infrastructure informatique traditionnelle, les services de partage de fichiers, messagerie, serveur web, VOIP (VOice by Internet Protocol), tournent dans le même serveur physique. Mais, si la messagerie est atteinte par un malware, l'ensemble des applications hébergées dans cette machine est menacée. En cloisonnant les services dans des machines virtuelles différentes isolées les unes des autres, des solutions fiables au problème de sécurité peuvent être trouvées ([Chowdhury & Boutaba, 2009](#)).

De nouvelles technologies et services peuvent être testés aisément sans débourser des frais supplémentaires. La virtualisation permet de créer une machine virtuelle (MV) vierge en quelques minutes. Tant que le serveur physique possède assez de ressources, il est possible de lui ajouter de nouvelles machines virtuelles à gérer. Les développeurs ou administrateurs système peuvent ainsi exploiter cette particularité pour essayer de nouveaux services, sans dépenser le moindre centime. Là où un serveur de test était auparavant nécessaire, une simple machine virtuelle sur un serveur existant de l'entreprise suffit aujourd'hui.

La virtualisation offre aussi la mise sur pied de réseaux adaptés aux besoins des utilisateurs et des applications en tenant compte de la criticité et de l'usage attendus des machines virtuelles. La virtualisation donne ainsi la possibilité à l'entreprise de proposer un catalogue de services et de tailles de machines virtuelles à des tiers, pouvant aboutir à un Cloud privé. Ces réseaux peuvent aussi servir sur le plan académique et de la recherche, à expérimenter de nouveaux protocoles et architectures de réseaux (projet PlanetLab ([Consortium et al., 2005](#)), GENI ([Foundation, 2019](#)), ...).

I.2.3. Propriétés de la virtualisation réseau

Afin de profiter pleinement des avantages liés à la virtualisation, la virtualisation des réseaux s'entoure de certains principes architecturaux et de conception ([Chowdhury & Boutaba, 2009](#)). Ces principes permettent d'implémenter des paradigmes réseaux futuristes et plus évolués.

La coexistence

Avant l'avènement de la virtualisation, les acteurs qui proposaient les infrastructures physiques, proposaient également les services aux utilisateurs. Mais la virtualisation des réseaux a poussé à la division de ces acteurs en deux principaux : les fournisseurs d'infrastructures physiques (FIPs) qui proposent le matériel et le réseau physique aux clients et les fournisseurs de services (FSs) qui s'occupent de développer et déployer des services sur des infrastructures louées auprès d'un ou de plusieurs FIPs. La coexistence renvoie au fait qu'on puisse avoir au sein d'un environnement de réseau virtuel, des réseaux virtuels bâtis sur des ressources provenant de FIPs différents.

La récursion

Des réseaux virtuels d'un FS2 peuvent être construits à partir des ressources d'un FS1, créant ainsi une hiérarchie père-fils entre réseaux virtuels. Dans ce cas, le FS1 cède au FS2 certaines de ses ressources. Le FS1 est alors vu par le FS2 comme un FIP virtuel. La figure 2 illustre cette récursion entre deux fournisseurs de services FS1 et FS2. Dans cette figure, une partie du réseau du FS2 est construite à partir du FS1 et le reste vient du FIP1.

L'héritage

Les réseaux virtuels fils peuvent hériter des attributs architecturaux et des contraintes de leurs parents. Par exemple, les contraintes imposées au réseau père sont transmises aux fils. Le fournisseur de service père peut également y ajouter d'autres contraintes avant de proposer ces réseaux virtuels à d'autres FS.

La revisitaton

C'est la capacité qu'a un fournisseur de service à reconfigurer logiquement ses réseaux virtuels et simplifier leur gestion grâce à l'hébergement de plusieurs nœuds virtuels au sein d'un même nœud physique.

La flexibilité

Il s'agit du principe de conception des RVs qui donne la liberté à l'utilisateur de définir la topologie, les schémas de routage et de reroutage, ses protocoles

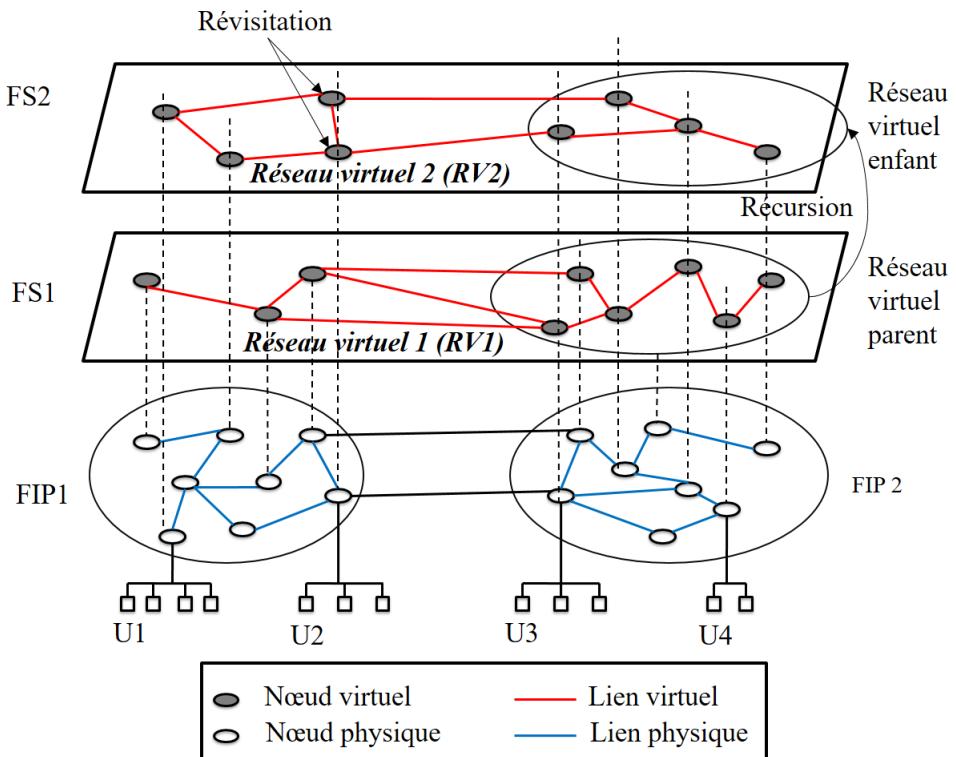


Figure 2 – Environnement de virtualisation réseau (Chowdhury & Boutaba, 2009).

de gestion de trafic indépendamment de ceux du réseau physique. De ce fait, le propriétaire d'un RV ne devrait pas avoir besoin de s'accorder obligatoirement avec un tiers pour définir ses politiques de gestion.

La scalabilité

La virtualisation réseau offre la possibilité d'implémenter plusieurs réseaux virtuels au-dessus de la même infrastructure physique. Elle doit donc intégrer l'évolution rapide du nombre de réseaux virtuels et ce, sans détériorer leurs performances. Tous les FIPs et FSs doivent chacun en ce qui leur concerne, mettre en place des stratégies de supervision de ces RVs. La figure 2 par exemple présente un réseau physique (celui formé par les FIP1 et FIP2) et deux réseaux virtuels (celui du FS1 et celui du FS2).

L'isolation

Les différents réseaux virtuels doivent être totalement indépendants les uns des autres en termes de stratégie de routage, protocoles de sécurité et de confidentialité ; cela permet en cas d'erreurs d'implémentation de protocole au sein d'un réseau virtuel, de ne pas répercuter l'erreur sur d'autres réseaux virtuels.

La stabilité et la convergence des réseaux virtuels

L'isolation permet certes de restreindre les erreurs de configuration au sein du RV source, mais le problème devient plus important lorsque ces erreurs ont lieu dans le réseau physique. En effet, un problème sur le plan physique, peut entraîner de graves perturbations au sein des réseaux virtuels. La virtualisation doit permettre de revenir facilement à l'état de stabilité de l'environnement virtuel.

La programmabilité

Les réseaux virtuels étant des réseaux logiques pouvant être générés par le biais de plateformes logicielles, on peut envisager la programmabilité de ces réseaux. Cela permettrait ainsi aux fournisseurs de services de développer et déployer rapidement leurs services.

La gestion de l'hétérogénéité des éléments en présence

L'hétérogénéité dans les réseaux virtuels renvoie à deux éléments principaux : les technologies utilisées dans les réseaux physiques (fibre optique, sans fil, capteurs, réseaux Ad hoc) et l'hétérogénéité des réseaux virtuels eux-mêmes. Les fournisseurs de services doivent avoir la liberté de mettre en place leurs réseaux virtuels sur plusieurs domaines d'administration sans être contraints à utiliser une solution imposée. Cela signifie que les FIP doivent proposer des infrastructures qui s'adaptent aux différents protocoles mis en place par leurs fournisseurs de services.

Ces propriétés de virtualisation réseau rendent ce concept difficile à implémenter et à intégrer dans les réseaux existants, raison pour laquelle plusieurs travaux ont essayé au fil des années de proposer des solutions. Ces projets ont abordé plusieurs aspects de la virtualisation à travers :

- La technologie d'interconnexion à utiliser (ATM, IP, ATM/IP, SONET) : le projet GENI (Global Environment for Network Innovations) ([Foundation, 2019](#)) est le plus récent sur ce point. Ce projet propose la création de réseaux virtuels personnalisés pour des expérimentations à grande échelle. La plate-forme d'expérimentation proposée sur internet dans le cadre de ce projet allie plusieurs technologies d'interconnexion hétérogènes (IP, SONET,...).
- La couche de virtualisation : s'inspirant du modèle en couche proposé par TCP/IP sur internet, beaucoup de projets ont essayé d'implémenter la virtualisation des réseaux à travers la virtualisation des couches IP. Ces projets vont ainsi de la virtualisation de la couche physique (projet UCLP ([Uclp, 2019](#))) à celle de la couche application (projet VIOLIN ([Ruth, Jiang, Xu, & Goasguen, 2005](#))) en passant par la couche liaison de données (projet VINI

(Bavier, Feamster, Huang, Peterson, & Rexford, 2006)) et la couche réseau (AGAVE (Boucadair et al., 2007), X-Bone (Touch, 2001)).

- Le domaine architectural : les projets allant dans ce sens avaient pour objectif de définir les choix d'implémentation de services devant fonctionner dans les environnements virtuels. Il s'agit par exemple de la gestion de ressources (projet Darwin (Chandra et al., 1998), SAIL (7Th Framework Program, 2010)) et de la supervision du réseau (projet VNRMS (Ng, Jun, Chow, Boutaba, & Leon-Garcia, 1999)).
- Le niveau de virtualisation : il s'agit du niveau d'indépendance de chaque réseau virtuel dans sa capacité à s'auto administrer. On a des RVs bâtis au sein d'un seul nœud, tout comme d'autres créés en combinant les nœuds virtuels issus de plusieurs nœuds physiques. Les projets PlanetLab (Consor-tium et al., 2005) et GENI (Foundation, 2019) sont les plus évolués dans ce sens. Ils ont fourni aux chercheurs une plateforme d'expérimentation à large échelle, permettant le déploiement et l'expérimentation de nouveaux services et protocoles.

Aujourd'hui, les projets de virtualisation tendent beaucoup plus à rendre les réseaux virtuels programmables en automatisant les tâches de gestion des entités en présence (routeurs, serveurs, ...). Dans ce sens, nous avons par exemple la solution Cisco OnePK (Cisco Open Network Environment Platform Kit) qui rend les équipements Cisco programmables, RouteFlow (Nascimento et al., 2011) et le projet SAIL (Scalable and Adaptive Internet soLutions) (7Th Framework Program, 2010) qui propose une architecture réseau capable de connecter rapidement et automatiquement les utilisateurs tout en optimisant l'utilisation des ressources. Mais, la solution SDN est l'une des solutions de plus en plus intégrée à ce jour dans les réseaux en général et les réseaux virtuels en particulier (Pujolle, 2015), grâce aux nombreux avantages que les entreprises en tirent.

I.3. Réseaux définis par logiciels (SDN)

Cette section présente la solution SDN (Software Defined Networking) qui est un nouveau pas vers les réseaux entièrement programmables. Nos contributions dans cette thèse sont bâties autour de cette solution qui est de plus en plus adoptée par les grands acteurs des réseaux (CISCO, JUNIPER, GOOGLE, ...).

I.3.1. Définition

Le SDN (Software Defined Networking) désigne littéralement la mise en réseau par logiciel. C'est un nouveau concept architectural des réseaux informatiques qui a vu le jour en 2008, du travail des équipes de recherche des universités de Berkeley et Stanford ([Guirlanger, 2015](#)). Il s'agit d'un paradigme réseau dans lequel le plan de contrôle des équipements réseau est découplé du plan de données de ces équipements. En effet, le plan de contrôle (*control plane* en anglais) désigne la partie d'un commutateur chargée de définir des stratégies de routage des paquets, tandis que le plan de données (*data plane* en anglais) est la partie qui gère le cœur de métier d'un commutateur ; son rôle est d'acheminer des paquets depuis un point A vers un point B. Autrement dit « switcher » et/ou « router » en se basant sur des informations contenues dans des tables. De ce fait, le plan de contrôle est situé dans un équipement appelé contrôleur SDN qui offre des APIs permettant de programmer le comportement du réseau, tandis que les plans de données restent hébergés dans les commutateurs.

Habituellement, dans un réseau classique, lorsqu'un paquet arrive sur un port d'un commutateur ou d'un routeur, celui-ci applique les règles de routage ou de commutation qui sont inscrites dans son système d'exploitation. Généralement, tous les paquets qui ont la même destination suivent le même chemin. Dans les modèles évolués, les équipements réseaux sont capables d'appliquer des règles spécifiques en fonction du type de trafic détecté. Mais cette programmation est rigide, car ne pouvant être modifiée manuellement que par l'administrateur ; ce qui prend évidemment du temps et ne se prête guère à des changements de contextes rapides ([Hu, Hao, & Bao, 2014](#)). En plus, lorsque le trafic réseau devient très dense ou le nombre de nœuds augmente, cette façon de procéder pose d'énormes problèmes de performance. Mais, grâce au principe du SDN, il est possible d'automatiser le fonctionnement de ces commutateurs en programmant leur plan de contrôle grâce aux APIs offerts par un contrôleur SDN. L'administrateur définit les règles dans le contrôleur, et celles-ci sont aussitôt transmises aux équipements réseaux. On peut par exemple automatiser le changement de bande passante dans un centre de données pour réaliser des sauvegardes à une certaine heure de la nuit et rétablir cette bande passante en journée, ou alors optimiser le coût des liaisons en fonction de l'heure ou du type de trafic. La figure 3 donne une illustration de cette approche par rapport aux réseaux conventionnels.

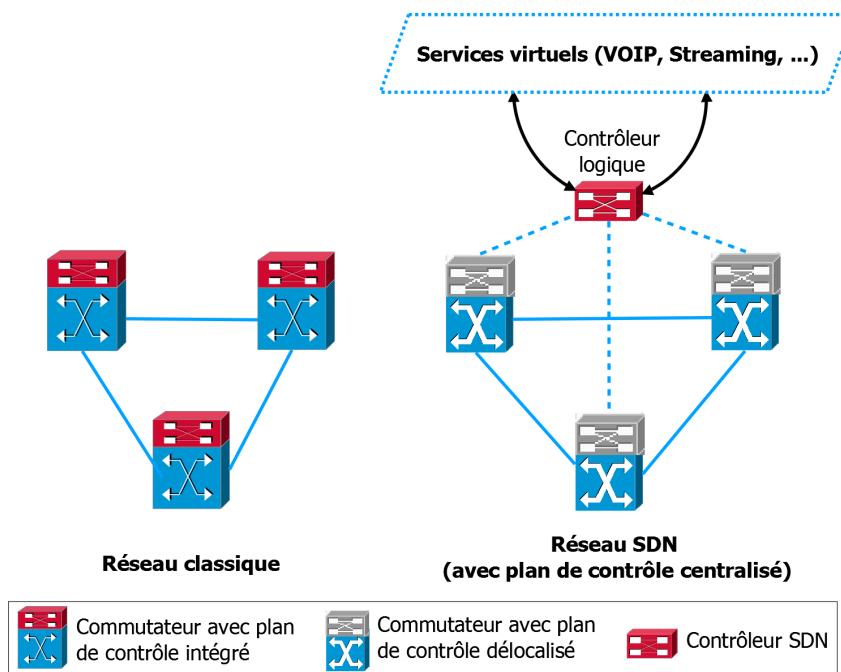


Figure 3 – Architecture réseaux traditionnels vs réseaux SDN (Nunes et al., 2014).

I.3.2. Avantages du SDN

Le SDN permet d'améliorer les performances du réseau en termes de supervision, contrôle et gestion des données.

En utilisant le SDN, les administrateurs ont la possibilité de contrôler le flux de données ainsi que de modifier les caractéristiques des dispositifs de commutation (ou de routage) du réseau à partir d'un emplacement central, avec l'application de contrôle implémentée en tant que module logiciel sans nécessité de traiter chaque appareil individuellement. Cela donne aux administrateurs réseaux la possibilité de modifier arbitrairement les tables de routage (chemins de routage) dans les périphériques de routage réseau. Cela permet également une couche supplémentaire de contrôle sur les données du réseau puisque l'administrateur peut attribuer des priorités hautes ou basses à certains paquets de données ou alors autoriser ou bloquer certains paquets circulant sur le réseau (Bakshi, 2013).

Du point de vue du Cloud computing, le SDN offre de grands avantages. Premièrement, cela facilite le déploiement des périphériques de différents fournisseurs par les fournisseurs de Cloud. Traditionnellement, les grands fournisseurs de services Cloud (tels que Google, Amazon, ...) doivent acheter les commutateurs/routeurs hautes performances auprès du même fournisseur afin de reconfigurer facilement les paramètres de routage (tels que la période de mise à jour de la table de routage). Les routeurs de différents fournisseurs ayant leurs propres avantages et inconvénients, personnaliser chaque routeur est un casse-tête car chaque

fournisseur peut avoir sa propre syntaxe de langage. L'interopérabilité n'est donc pas évidente à gérer. Grâce à l'intégration du protocole OpenFlow conçu pour gérer les interactions entre les équipements dépourvus de leur plan de contrôle, SDN permet désormais à un fournisseur de Cloud de redéfinir rapidement les règles de routage et d'allocation de ressources en cas de problèmes, à condition que les routeurs de chaque fournisseur se conforment à la norme SDN-OpenFlow.

Le SDN permet aussi à un utilisateur dans le Cloud, d'utiliser plus efficacement les ressources du Cloud ou de mener des expériences scientifiques en créant des tranches de flux virtuels. Le protocole OpenFlow est compatible avec le standard GENI, ce qui permet à un utilisateur de créer de manière arbitraire des couches virtuelles sans se soucier de l'infrastructure réseau physique. Quelle que soit l'infrastructure, qu'il s'agisse d'un réseau sans fil ou câblé, et quelle que soit la manière dont le fournisseur de services Cloud déploie différentes unités de stockage dans différents emplacements, le concept de flux virtuel dans un SDN permet au flux de données de traverser de manière transparente tous les périphériques du Cloud. Dans une telle architecture, le commutateur OpenFlow et le contrôleur communiquent via le protocole OpenFlow, qui définit des messages particuliers tels que le *packet-received*, *packet-in*, *send-packetout*, *modify-forwarding-table*, et *get-stats* (Son, 2014). Quand un commutateur OpenFlow reçoit un paquet qu'il n'avait jamais vu auparavant et pour lequel il n'a aucun flux d'entrée correspondant dans sa table de routage, il envoie ce paquet au contrôleur. Le contrôleur prend alors une décision sur la façon de gérer ce paquet. Il peut décider de supprimer le paquet, ou il peut ajouter une entrée dans la table de commutation du switch pour lui indiquer comment transférer des paquets similaires dans le futur. Cette grande flexibilité des équipements OpenFlow (Specification, 2009) laisse facilement cours à l'innovation de la part des opérateurs de réseaux, les chercheurs et les vendeurs d'équipements OpenFlow.

Par ailleurs, grâce à sa capacité à optimiser la distribution des flux de données à travers une interface de contrôle, le SDN intègre l'intelligence, la rapidité dans les transmissions ainsi qu'un usage efficient des ressources réseau. De plus, à travers le contrôle global qu'ils ont du réseau, les administrateurs peuvent modifier aisément les services et la connectivité entre équipements en fonction du trafic. Ceci se fait grâce à des protocoles développés et intégrés dans le contrôleur et qui réagissent automatiquement face aux variations du réseau. Ainsi, pour redéfinir par exemple les règles de routage, le contrôleur grâce à sa vue globale du réseau, calcule les règles de routage et les transmet aux nœuds du réseau plus rapidement (Hu et al., 2014 ; Son, 2014). La vue globale du réseau par le contrôleur permet ainsi

de remplacer les protocoles de routage distribués (OSPF, EIGRP, BGP,...) par des mécanismes plus simples (pas besoin de découvrir la topologie, pas de problème de convergence...). Tous les avantages ainsi présentés justifient notre intérêt d'intégration du SDN et des switch OpenFlow dans notre ambition d'amélioration de la qualité de service dans les réseaux virtuels.

I.4. Notion de qualité de service dans les réseaux

La notion de qualité de service ou QoS renvoie à la capacité d'un réseau à transporter de manière satisfaisante les flux de données en tenant compte des besoins des utilisateurs ([Pujolle, 2014](#)). Face aux exigences sans cesse croissantes des applications de plus en plus diverses, les opérateurs de réseaux ont toujours eu à faire face à ce grand défi qu'est la gestion de la QoS. Plusieurs travaux jusqu'à présent ont été menés dans la perspective de proposer en tout temps et circonstance une QoS satisfaisante aux utilisateurs dans les réseaux IP. Chacun de ces travaux essaie tour à tour d'améliorer le plus de métriques rattachées à cette QoS. Etant donné que nous évoluons dans la même logique dans cette thèse, nous présenterons dans ce qui suit les métriques de QoS et quelques modèles basiques de QoS existants.

I.4.1. Paramètres de QoS

La notion de qualité de service, ou QoS, concerne certaines caractéristiques d'une connexion réseau relevant de la seule responsabilité du fournisseur du service réseau. Le FS doit en tout temps se rassurer que chaque valeur de QoS est la même sur les extrémités d'une connexion réseau, indépendamment du nombre de sous-réseaux pris en charge et différents services proposés. Les principales métriques qui définissent la QoS dans les réseaux IP en général sont les suivantes :

- **le débit du transfert des données** ([Chimento & Ishac, 2008](#)) : c'est le nombre d'octets transportés sur une connexion réseau dans un temps raisonnable (quelques minutes, quelques heures ou quelques jours). Ce paramètre est généralement difficile à évaluer ; ceci à cause de l'asynchronisme du transport des paquets IP. En effet, pour obtenir une valeur crédible, il faut observer le réseau sur une suite de plusieurs paquets et considérer le nombre d'octets de données transportés en tenant compte du temps écoulé depuis la demande ou l'indication de transfert des données.
- **Le temps de transit lors du transfert des données** ([Almes, Kalidindi, & Zekauskas, 1999a, 1999c](#)) : il correspond au temps écoulé entre le transfert

de données d'une source vers une destination. Ce temps de transit est aussi difficile à calculer, du fait de la distribution géographique des extrémités.

- **Le taux d'erreur résiduelle** (Pujolle, 2014) : c'est la probabilité que les données n'atteignent pas correctement leur destination. Il se calcule à partir du nombre de paquets qui arrivent erronés, perdus ou en double sur le nombre total de paquets émis.
- **Le taux de perte de paquets** (Almes, Kalidindi, & Zekauskas, 1999b) : elle désigne la probabilité que les données n'atteignent pas leur destination, c'est à dire le ratio du nombre de paquets perdus par rapport au nombre de paquets envoyés. Il est souvent utilisé pour déduire le taux d'erreur résiduelle.
- **La gigue** (Demichelis & Chimento, 2002) : il s'agit de la variation de latence des paquets. C'est un paramètre dont la valeur doit en tout temps être raisonnable sous peine de détériorer considérablement des services tel que la vidéo-conférence, le streaming ou encore la VOIP ; car il est assez pénible de voir des communications entrecoupées ou des images en déphasage avec le son lors d'une communication vidéo. Ce paramètre doit donc être très surveillé pour les services multimédia.

Les réseaux virtuels étant bâtis sur la base des réseaux physiques, la gestion de la QoS dans ces réseaux intègre également les paramètres sus cités. Nos travaux dans le cadre de cette thèse visent à améliorer ces paramètres en situation de perturbation du réseau.

I.4.2. Modèles de QoS

L'IETF (Internet Engineering Task Force), l'organisme de normalisation du monde Internet, a fait de nombreuses propositions ces dernières années pour introduire la qualité de service dans les réseaux IP ; même si concernant les nouveaux paradigmes (virtualisation, SDN), il reste encore beaucoup à faire. Les deux plus importantes propositions sont le modèle des *services intégrés IntServ (Integrated Services)* et le modèle *Differentiated Services (DiffServ)* (Pujolle, 2014).

Le modèle IntServ définit une architecture capable de prendre en charge la Qualité de Service en définissant des mécanismes de contrôle complémentaires sans toucher au fonctionnement d'IP. C'est un modèle basé sur un protocole de signification RSVP (Resource Reservation Protocol) qui permet d'acheminer les informations des routeurs en fonction des exigences de flux (Zhang, Berson, Herzog, & Jamin, 1997). RSVP est le protocole qu'utilisent les applications pour réserver des ressources du réseau. À grande échelle, IntServ admet beaucoup de limitations. En effet, l'utilisation du protocole RSVP par IntServ implique le maintien des états de

réservation de chaque flux traversant les routeurs. Le nombre d'états à gérer devient complexe pour IntServ avec l'augmentation du nombre de flux. Il en découle une détérioration des performances dans le réseau. De plus, le grand nombre de messages de rafraîchissement échangés dans le soucis de tenir compte des nouvelles exigences de l'utilisateur, entraîne un accroissement de la charge du réseau et par conséquent une augmentation de la probabilité de suppression des paquets. IntServ souffre donc du problème de passage à l'échelle et n'est pas adapté aux réseaux de grande taille.

DiffServ a été conçu pour pallier les difficultés de mise à l'échelle d'IntServ. DiffServ sépare l'architecture en deux composantes majeures : la technique de transfert et la configuration des paramètres utilisés lors du transfert. Cela concerne aussi bien le traitement reçu par les paquets lors de leur transfert dans un nœud que la gestion des files d'attente et la discipline de service. L'objectif est de fournir, d'un côté, une QoS par classe de paquets IP et non pas par flux. D'un autre côté, il pousse la surcharge du traitement aux extrémités du réseau cœur. Autrement dit les routeurs aux frontières d'un réseau, s'occupent de la classification des paquets en fonction du type de trafic et les routeurs au cœur du réseau appliquent les comportements prévu pour chaque classe de flux. Cette façon de faire a permis pendant longtemps aux fournisseurs de services, de proposer différentes classes de services (Gold, Platinum, Silver...) pour divers tarifs, engendrant ainsi beaucoup de revenus. Ce mécanisme et ses variantes comme le MPLS (Multi-Protocol Label Switching) DiffServ-aware Traffic Engineering, permettent de garantir la QoS au sein d'un même domaine. Mais le problème de gestion de la QoS persiste lorsque les flux doivent transiter par d'autres domaines avant d'atteindre l'utilisateur final, ou alors en présence de panne sévère d'une liaison ou d'un nœud.

I.5. Pannes de nœuds et de liaisons

Les pannes sont des phénomènes récurrents dans la plupart des systèmes implantés par l'homme. Les réseaux informatiques en général n'y échappent pas. Ces pannes sont la majeure partie du temps liées à l'hétérogénéité des utilisateurs du réseau, les natures différentes des équipements ayant chacune leur caractéristiques propres (résistivité, fiabilité, durée de vie...) et les facteurs environnementaux qui influencent le système (variations d'énergie électrique, catastrophe naturelle, climat ...) ([Servin & Arnaud, 2003](#)). Dans cette section, nous définissons formellement la notion de panne aussi bien dans un réseau physique que virtuel, et passons

en revue quelques pistes de solutions déjà proposées jusqu'ici dans la perspective de sa résolution.

I.5.1. Notion de panne

Dans les réseaux informatiques, une liaison désigne une connexion physique (fibre optique, câble coaxial, câbles à paires torsadées, ...) ou virtuelle (tunnels MPLS) entre deux équipements. La liaison est donc maintenue stable par le biais des équipements qui la définissent de bout en bout. Un nœud quant à lui, est tout point du réseau par lequel transite un flux et au sein duquel il est possible d'effectuer des traitements sur ce flux. Un nœud est aussi le point d'interconnexion de plusieurs liaisons du réseau.

Une panne désigne un arrêt complet ou partiel d'un fonctionnement. Dans le cadre des réseaux, cet arrêt est souvent relatif à la défaillance des liaisons ou des nœuds et aux congestions. Une panne de liaison désigne l'incapacité d'une liaison à transporter un flux dans l'un ou l'autre de ses sens de transmission. En effet, une liaison peut transmettre une information soit dans une direction (liaison unidirectionnelle) ou alors dans deux directions (liaison bidirectionnelle (voir figure 4c)). Les différents sens de circulation des informations au sein d'une liaison constituent les arcs de cette dernière. Pour une liaison (A-B) par exemple, on distingue l'arc (A→B) (voir figure 4a), l'arc (B→A) (voir figure 4b) et les deux sens (voir figure 4c). On parlera ainsi de *panne partielle* d'une liaison bidirectionnelle lorsque les informations ne peuvent plus circuler que dans un seul sens, et de *panne complète ou totale* lorsque cette circulation devient impossible dans les deux sens. Ces définitions sont les mêmes dans les réseaux conventionnels et dans les réseaux virtuels (Son, 2014).

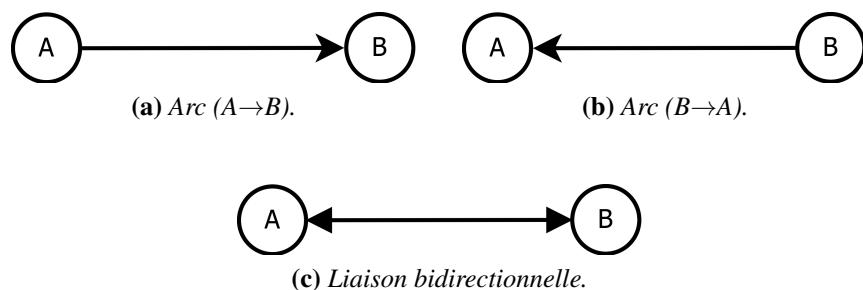


Figure 4 – Les différents arcs d'une liaison (A-B).

Nous classifions les pannes de liaisons et de nœuds en fonction de deux critères :

- *Le nombre de nœuds ou de liaisons impliqués* : suivant ce critère, on parle de panne simple lorsqu'on a à faire à la panne d'un seul nœud ou d'une

seule liaison, et de panne multiple lorsque plusieurs nœuds ou liaisons sont en pannes.

- *La durée de la panne* : lorsqu'une panne dure moins d'un certain seuil, elle est qualifiée de *panne temporaire*, tandis qu'au-delà de ce seuil, elle est qualifiée de *panne persistante*. Le seuil est défini par le FS qui souhaite implémenter des stratégies de tolérance aux pannes en fonction de ses objectifs propres.

Une panne de nœud désigne quant à elle, l'incapacité d'un flux à transiter d'une interface d'entrée de ce nœud à une interface de sortie du même nœud. En fonction du degré du nœud considéré, une panne de nœud peut induire une panne de liaison multiple.

A cause des effets néfastes des pannes sur la qualité de service (perte de données, latence) la plupart des systèmes actuels dits à *tolérance de panne* (*fault tolerant en anglais*), implémentent des stratégies de prévention et de prise en main de ces types de pannes. C'est ainsi que pour les centres de données par exemple, des stratégies de *mirroring*¹, de *duplexing*² et de RAID (Redundant Array of Independent Disk) (Servin & Arnaud, 2003) sont souvent utilisés pour gérer les pannes d'équipements de stockage. Le point commun de ces techniques réside sur l'utilisation d'un équipement de repli pour continuer à garantir le fonctionnement du système. Dans le cas des pannes de liaisons, ces éléments de repli sont souvent les liaisons voisines aux liaisons en pannes ; la ressource généralement utilisée dans ces liaisons pour supporter le flux ainsi rerouté, est la capacité résiduelle de la liaison (Son, 2014 ; Veerasamy et al., 1994).

I.5.2. Notions de capacité résiduelle et additionnelle

Dans un réseau informatique, chaque liaison utilisée pour le transport des données est caractérisée par deux éléments principaux : la bande passante et le débit. La bande passante désigne la quantité d'information pouvant être transmise par unité de temps sur la liaison sans perte, alors que le débit désigne la quantité de données effectivement transmise sur la liaison par unité de temps. Ainsi, pour une liaison à 512 Mbits par seconde (Mbits/s) (bande passante ou capacité dédiée de la liaison), le débit (aussi appelé capacité active) peut varier de 0 à 512Mbits/s au

1. **Le mirroring** est une technique dans laquelle le système de secours est maintenu en permanence dans le même état que le système actif (miroir). Le mirroring disques consiste à écrire simultanément les données sur deux disques distincts. En cas de défaillance de l'un, l'autre continue d'assurer les services disques. La panne est transparente pour l'utilisateur. Après remplacement du disque défectueux, le système reconstruit automatiquement le disque miroir.

2. **La duplexing** consiste à avoir un équipement disponible qui prend automatiquement le relais du système défaillant. C'est une évolution de la technique de mirroring.

maximum. Dans les cas où la valeur maximale de la bande passante n'est pas atteinte, la valeur restante est appelée la capacité résiduelle de la liaison (Veerasamy et al., 1994).

Lorsqu'un support de transmission physique est fabriqué, sa bande passante n'est généralement pas modifiable. De ce fait, en cas de besoin d'une bande passante supplémentaire, il faudrait changer le support physique de transmission. Cependant, la bande passante des liaisons virtuelles étant paramétrable autant que possible dans les limites des liaisons physiques dont ces dernières sont issues, il est alors possible de redéfinir dynamiquement, la bande passante de ces liaisons virtuelle ; cela se fait en augmentant ou diminuant de la valeur (appelée capacité additionnelle) à la bande passante initiale de la liaison virtuelle. La figure 5 donne une illustration de ces différentes capacités.

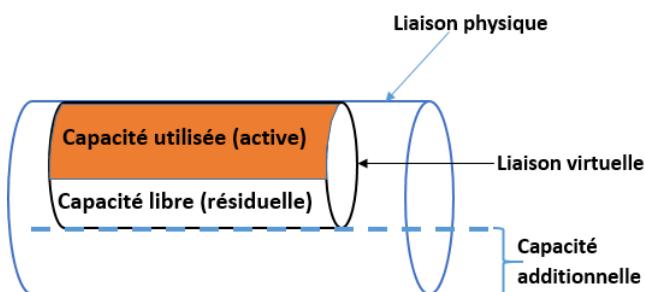


Figure 5 – Différents types de capacité.

I.5.3. Mécanismes de routage pour la tolérance aux pannes de nœuds et de liaisons

La tolérance aux pannes est un problème qui a été largement étudié dans les réseaux informatiques depuis leur implémentation. Plusieurs approches de rerouting ont été jusqu'ici proposé dans les réseaux conventionnels. Toutefois, parlant des environnements virtuels intégrant le SDN/OpenFlow, beaucoup reste encore à faire. Dans ce type d'environnement, les équipements dépourvus de plan de control ne peuvent plus s'identifier mutuellement et prendre des décisions de rerouting majeures de façon autonome ; ils doivent donc en référer à un contrôleur qui supervise l'intégration des stratégies de gestion du réseau.

Les approches de tolérance aux pannes effectuées jusqu'ici se regroupent en plusieurs catégories : les approches proactives (Dong, Shen, & Sun, 2017 ; Khan, Shahriar, Ahmed, & Boutaba, 2016), les approches réactives (Dong et al., 2017) et les approches à chemins précalculés hybrides ou approches hybrides (Kamamura et al., 2013 ; Son, 2014).

I.5.3.1. Approches de reroutage proactives

Les approches proactives essaient d'anticiper la situation de panne en réservant des ressources dans le réseau afin de faire face à une éventuelle panne de liaison (Khan et al., 2016 ; Rahman, Aib, & Boutaba, 2010) ou de nœud (Shahriar et al., 2016). Ce type d'approche en général essaie de proposer une stratégie de mappage des nœuds et liaisons du réseau virtuel où se produit la panne, avec les ressources du réseau physique : il s'agit des approches d'approvisionnement des réseaux virtuels (Virtual Network Embedding VNE). Ce problème d'approvisionnement est à la base NP-difficile en raison des contraintes sur les nœuds et les liaisons qu'il faut prendre en compte. Mais, les solutions existantes proposent des heuristiques qui s'appuient sur des modèles mathématiques linéaires à base d'entiers (Integer Linear Program ILP) pour proposer des formulations et des solutions approchées lorsque la panne survient. La limite majeure de plusieurs de ces approches (Jiang, Wang, Gong, & Zhu, 2015 ; Rahman et al., 2010) réside dans le fait que la réservation de ressource est fixée à un quota de départ et la possibilité d'une réservation dynamique n'est pas envisagée. Cette possibilité de réservation dynamique est très difficile à intégrer dans les réseaux virtuels (Khan et al., 2016).

D'autres approches proactives essaient de déterminer à l'avance les ports de repli (backup ports) potentiels à travers lesquels il faudrait rerouter les flux issus des pannes. Dans ce sens, Xi et Chao (2007) proposent la méthode IPFRR (Internet Protocol Fast ReRoute), qui utilise deux types de ports au sein d'un commutateur : les ports primaires et les ports secondaires. Dans cette approche, le trafic migre d'un port primaire vers un port secondaire lorsqu'il y a une panne sur le port primaire ou lorsque la panne provient de ce port. Cette politique de transfert implique que dans un réseau implémentant cette approche, le trafic suit le chemin inverse à celui ayant mené au port primaire en panne ; cette remontée de chemin se poursuit jusqu'à trouver un nœud à partir duquel il peut se connecter au port primaire d'un autre nœud ; la liaison ainsi utilisée pour se connecter au port primaire de ce nœud est appelé *pont*. Le nombre de ports secondaires traversés pour atteindre le pont est minimisé. Sous le respect du principe d'optimalité, lorsque plusieurs chemins de destination commune ont un nœud en commun, ces chemins doivent être identiques après ce nœud jusqu'à la destination : cela permet de garantir l'absence de conflit de routage entre les chemins. En effet, deux chemins de routage sont en conflits lorsque pour une même destination et ayant un arc en commun, ils ne sont pas identiques après cet arc jusqu'à cette destination (Son, 2014). La figure 6 donne une illustration du conflit. Les chemins suivis par les trafics T1 et T2 ont en commun l'arc (B→C) pour la destination E, mais pourtant ils sont en conflit dans

la figure 6b et pas en conflit dans la figure 6a parce qu'ils suivent le même chemin C→E après l'arc (B→C) jusqu'à la destination E.

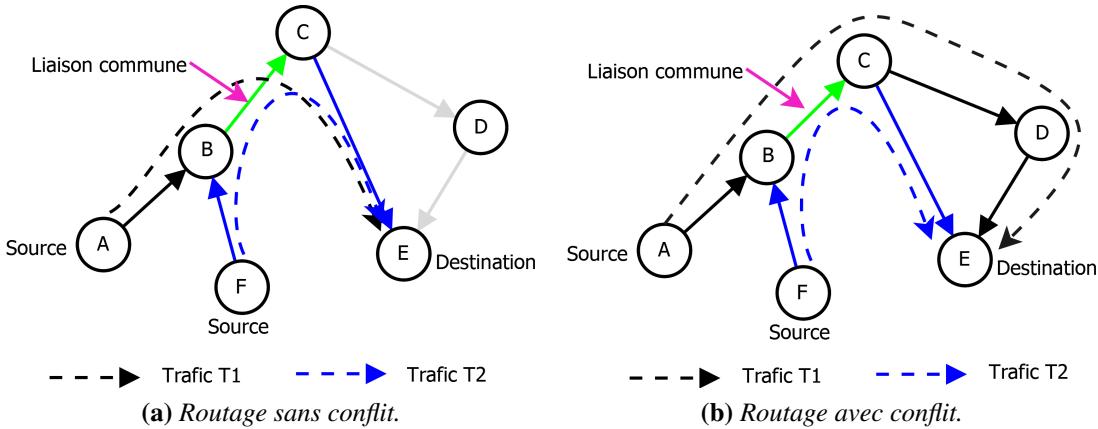


Figure 6 – Illustration du conflit de routage.

Par ailleurs, la méthode de [Xi et Chao \(2007\)](#) ne minimise pas la capacité ajoutée à chaque liaison. L'intérêt de la minimisation de la capacité additionnelle réside dans le fait qu'elle réduit les coûts de dimensionnement du réseau ; cela participe à relever le challenge de l'allocation des ressources qui est l'un des problèmes majeurs dont nous faisons face dans les réseaux virtuels ([Chowdhury & Boutaba, 2009](#)). De plus, le modèle ILP proposé n'intègre pas la prise en compte des conflits potentiels. Plusieurs autres méthodes ([Atlas & Zinin, 2008](#) ; [S. Bryant & Shand, 2013](#) ; [Wang & Nelakuditi, 2007](#)) ont été proposés sur le modèle IPFRR avec comme avantage la réduction du taux de perte de paquets dans les réseaux IP, sans pour autant être compatible avec les réseaux virtuels.

I.5.3.2. Approches de reroutage réactives

Elles consistent à prendre des mesures de reroutage au moment où la panne survient ([Sharma, Staessens, Colle, Pickavet, & Demeester, 2011](#)). [Sharma et al. \(2011\)](#) proposent un système de reroutage pour les réseaux bâtis sur le protocole OpenFlow. Lorsqu'une panne de liaison survient dans le réseau, le nœud en amont du flux contacte le contrôleur grâce à un message spécifique appelé *packet-in message*. Un *packet-in message* est un message envoyé par un switch vers un contrôleur, tandis qu'un *packet-out message* est envoyé par le contrôleur vers le switch. Une fois le *packet-in message* reçu, le contrôleur détermine le chemin de reroutage à utiliser et transmet les informations de routage au switch via un *packet-out message*.

Cette stratégie permet d'apporter une solution aux pannes de liaisons dans les réseaux virtuels OpenFlow, mais souffre d'un temps de latence assez important par

rapport à d'autres travaux (Kamamura et al., 2013 ; Sharma, Staessens, Colle, Piickavet, & Demeester, 2013 ; Son, 2014) qui précalculent les chemins de reroutage avant la mise en marche du réseau.

I.5.3.3. Approches à chemins précalculés hybrides

Les approches à chemins précalculés hybrides consistent à calculer avant la mise en marche du réseau, tous les chemins de reroutage possibles pour chaque possibilité de panne de liaison ou de nœud et l'insérer dans les nœuds ; mais lorsque certaines conditions particulières l'exigent (panne persistante, congestion de trafic, changement de topologie, ...), les chemins précédents peuvent être recalculés.

Kamamura et al. (2013) proposent un système de reroutage qui précalcule les chemins de reroutage pour les cas possibles de pannes dans les réseaux IP utilisant OpenFlow. Lorsqu'une panne survient, le switch OpenFlow la détecte à travers un changement d'état du port impliqué, puis applique les règles précalculées. À la réception d'un flux à rerouter, le commutateur recherche dans sa table de routage des flux, l'entrée correspondante au flux à rerouter sur la base des informations d'en-tête des paquets d'entrée. Si une telle entrée de flux existe, le commutateur transmet les paquets à travers l'interface de sortie prévue à cet effet; sinon, le commutateur transmet les paquets au contrôleur. La limite de cette approche est également la non prise en charge des conflits potentiels dans le reroutage.

Dans le souci de prendre en compte cette contrainte de conflit, des travaux de reroutage exploitant des arbres de routage peuvent aussi servir. Dans cette optique, Son (2014) propose une méthode préemptive permettant de déterminer à l'avance les chemins de reroutage pour toute situation de panne simple et multiple non simultanée dans le cas de réseaux virtuels avec protocole OpenFlow. Les chemins sont précalculés par un contrôleur SDN. La recherche du chemin de reroutage proposée combine deux stratégies : la recherche de chemins par IPFRR (Xi & Chao, 2007) et la minimisation de la capacité additionnelle nécessaire au niveau des liaisons. Cette approche considère que pour une destination donnée, les flux sont acheminés dans le réseau à travers un ensemble de chemins constituant un *arbre de routage nominal*. Cet arbre est construit en déterminant les plus courts chemins de chaque nœud vers la destination considérée, et peut faire l'objet d'une reconstruction lorsque cela s'avère nécessaire. La figure 7b donne (en bleu) les chemins constituant l'arbre de routage nominal pour la destination D.

Lorsqu'une panne de liaison (S-D) par exemple se produit dans le réseau (voir figure 7c), elle entraîne la subdivision logique de l'arbre de routage nominal en deux sous-graphes : le graphe rouge G_r qui contient l'ensemble des nœuds et liai-

sons susceptibles de déboucher sur la liaison en panne ; et le graphe bleu G_b contenant l'ensemble des nœuds et liaisons débouchant sur la destination (ici le nœud D) de l'arbre de routage nominal (voir figure 7c). Selon la stratégie IPFRR, le reroutage est initié par un seul des nœuds composant la liaison (S-D) ; dans le cas d'espèce, ce nœud est le nœud S. Ce nœud S initie immédiatement le reroutage lorsqu'il détecte en interne un changement d'état du port primaire composant la liaison (S-D). Les autres nœuds ne sont pas au courant de la panne et appliquent leurs règles de routage normalement. Le reroutage consiste à trouver un pont pour relier les deux sous graphes G_r et G_b . Tout le trafic à destination du nœud D qui transitait par le nœud S ne pourra plus passer par la liaison (S-D) et sera donc redirigé suivant le chemin $S \rightarrow T \rightarrow U \rightarrow V \rightarrow K \rightarrow D$ et le pont ($U \rightarrow V$) en destination de D. Nous rappelons que ce chemin a été calculé à l'avance au niveau du contrôleur et intégré au sein du nœud S. Ainsi, le trafic acheminé via l'arc ($T \rightarrow S$) en destination de D est rerouté à travers l'arc ($S \rightarrow T$) et cela ne cause pas de problème de boucle à cause de la programmation des filtres. Grâce aux filtres intégrés dans les équipements OpenFlow, lorsque le noeud T reçoit le trafic venant de U, sachant que sa destination est D, T transfère ce trafic vers le nœud S. En cas de panne de liaison, S transfère le trafic vers T. Sachant que le trafic provient de S à destination de D, T dirige tout ce trafic vers U, qui le transfère à son tour vers V. Ensuite ce même trafic sera transféré de V vers K, ainsi de suite jusqu'à atteindre la destination D.

La limite majeure de l'approche de Son (2014) réside dans le fait qu'en cas d'impossibilité de trouver une liaison offrant assez de capacité résiduelle pour rerouter le trafic, la capacité additionnelle est utilisée directement. Pourtant, dans le cas d'espèce, la combinaison de ressources de plusieurs liaisons voisines pourrait permettre de ne pas recourir à cette capacité additionnelle ou alors le cas échéant, minimiser sa quantité. La stratégie de séparation de flux proposée par Veerasamy et al. (1994) permet d'atteindre cet objectif d'exploitation judicieuse de la capacité résiduelle par le reroutage des flux à travers plusieurs liaisons. Cette méthode de séparation de flux est également reprise par Khan et al. (2016) pour élaborer une approche de protection des liaisons contre les pannes dans les réseaux virtuels. Mais, ces méthodes ne tiennent pas compte des conflits potentiels dans les chemins de reroutage. Nous exploitons cette séparation de flux dans la gestion de la panne d'une liaison, tout en intégrant la gestion des conflits et en minimisant la capacité additionnelle prélevée sur le réseau physique.

Les méthodes de routage et reroutage présentées dans cette section traitent des pannes temporaires de liaisons ; dans le cas des méthodes utilisant des chemins précalculés, ces chemins deviennent obsolètes lorsque la panne persiste dans le

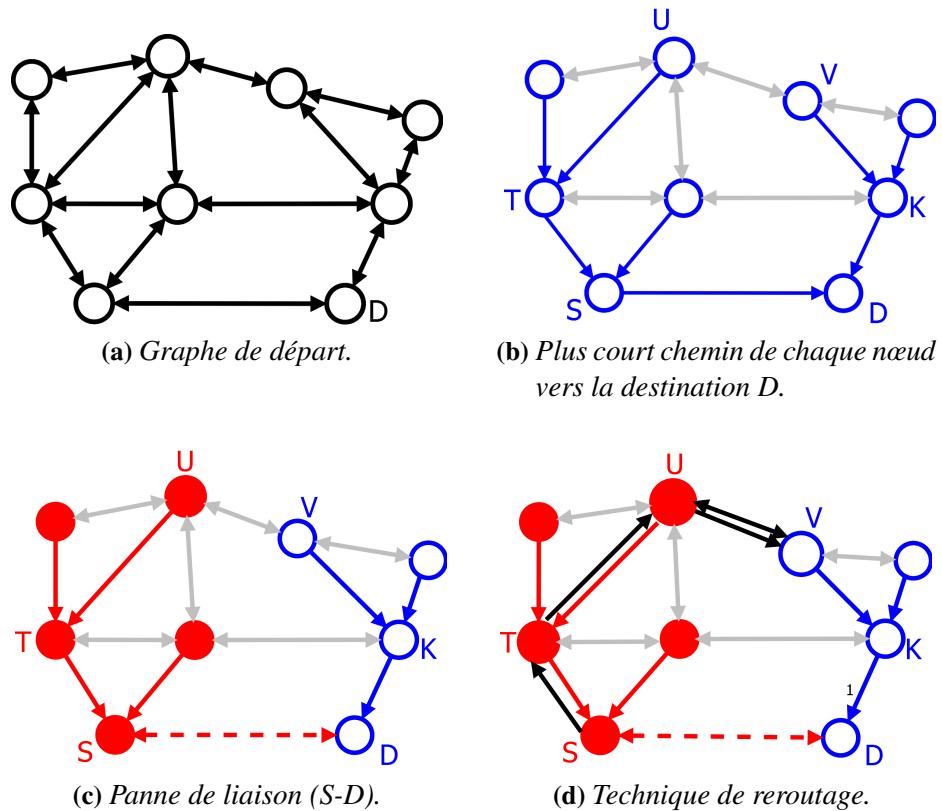


Figure 7 – Méthode de reroutage de Son (2014) : Cas de pannes simples (Son, 2014).

temps. Il faut alors penser dans le contexte du SDN, à recalculer les chemins de reroutage et mettre à jour les nœuds concernés par la panne.

I.6. Mise à jour des tables de routage pour la tolérance aux pannes persistantes

Afin de mieux appréhender le problème de mise à jour des tables de routage dans les réseaux virtuels, nous présentons dans cette section une formulation de celui-ci en passant par les motivations d'une telle étude, puis nous passons en revue quelques travaux effectués dans ce sens.

I.6.1. Formulation du problème de mise à jour

Le problème de mise à jour des tables de routage des différents nœuds d'un réseau, consiste à ajouter, supprimer ou remplacer partiellement ou totalement les informations présentes dans ces tables ; cette mise à jour est faite dans l'objectif de communiquer de nouveaux comportements au réseau face à certaines situations bien particulières. Les équipements réseau pouvant être concernés par cette mise

à jour sont le switch et le routeur, parce que ceux-ci maintiennent en leur sein une table de routes affectant une ou plusieurs sorties potentielles à tout flux entrant. Ainsi, un switch fera correspondre à tout port recevant un flux entrant, un ou plusieurs ports de sorties ; un routeur quant à lui, proposera une ou plusieurs interfaces de sorties à tout flux provenant d'une de ses interfaces entrante. De ce fait, une étude réalisée dans un environnement comportant des switchs serait également valable dans un autre intégrant des routeurs. Par ailleurs, dans un environnement réseau adoptant une architecture SDN, les routeurs qui sont alors dépourvus de leur fonction intelligente, fonctionnent comme des switchs.

Une illustration de ce problème de mise à jour est présentée par la figure 8. Cette figure montre à travers l'arbre de routage nominal (arbre constitué des plus

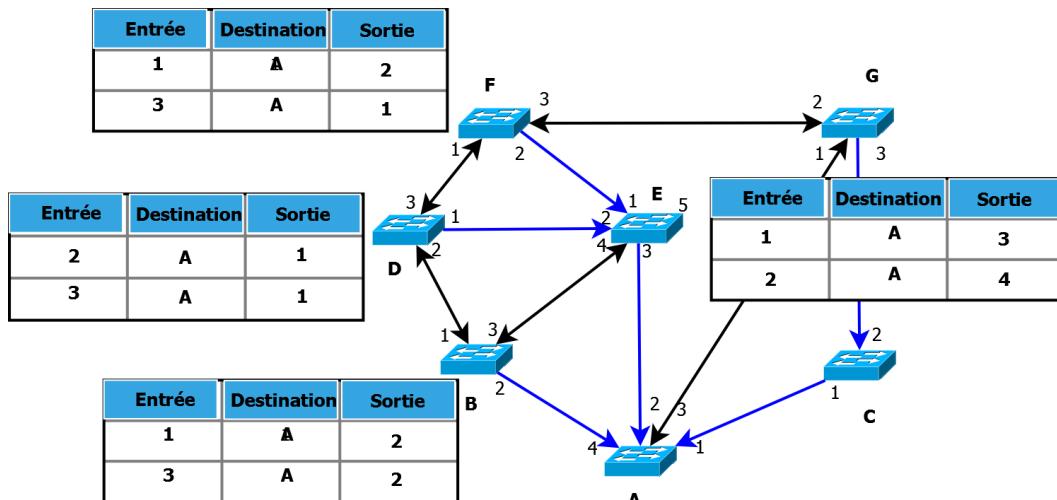


Figure 8 – Le problème de mise à jour des tables de routage.

courts chemins partant de chaque nœud vers une destination commune) représenté par les flèches colorées en bleue, la circulation des flux dans le réseau ainsi pris pour exemple. Chacun des nœuds maintient une table de routage permettant l'acheminement des flux de proche en proche vers une destination précise, ici le nœud étiqueté A. Plusieurs problèmes dus à une mise à jour hasardeuse ou non efficiente des tables de routage peuvent s'y dégager :

La redirection vers des sorties non connectées

A propos de ce type de problème, la mise à jour d'une table de routage entraîne une redirection des flux vers un ou plusieurs ports du switch non connectés à d'autres noeuds du réseau. Par exemple, une modification de la table de routage du nœud E au niveau de la première ligne afin de réorienter les flux entrant par le port numéro 1 vers le port de sorti numéro 5, entraîne

la perte de tout flux entrant par le port 1 de ce nœud, car aucun nœud n'est relié au nœud E par l'intermédiaire du port 5.

La génération des cycles dans le réseau

Il s'agit ici de l'emprisonnement d'une ou de la totalité du trafic réseau dans une boucle. Considérons la figure 9 dans laquelle nous décidons d'effectuer les mises à jours suivantes dans les tables de routage de certains nœuds :

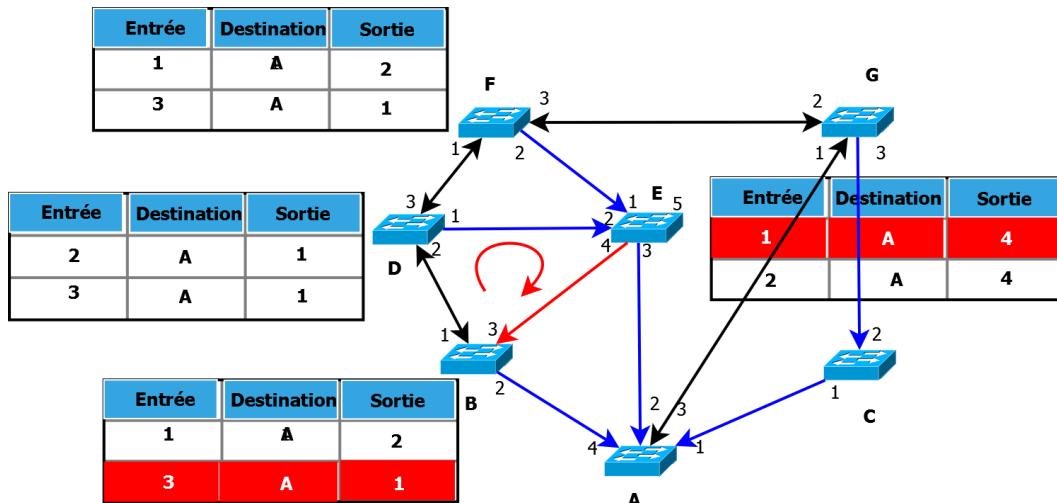


Figure 9 – Génération de cycle dans la mise à jour des tables de routage.

Au niveau du nœud E, l'on décide de mettre à jour la table de routage au niveau de la première ligne en réorientant les flux entrant par le port 1 vers le port de sortie numéro 4 et non celui de numéro 3 comme initialement ; l'on apporte également une modification au niveau de la table de routage du nœud B en réorientant tout flux entrant par le port 3 vers le port de sorti numéro 1 et non celui de numéro 2. La configuration que nous obtenons à la suite de ces mises à jour crée le cycle D-E-B-D. En effet, tout flux sortant par le port 1 du nœud D sera acheminé vers l'entrée 1 du nœud E qui va l'acheminer à son tour vers la sortie 2 ; ensuite le nœud B recevant ce flux par l'entrée 3 va l'acheminer vers la sortie 1 en direction du nœud D initial, et le processus recommencera indéfiniment. Ces cycles augmentent considérablement le *délai de transit* et le *taux de perte des paquets*, qui sont font partie des principaux critères de la qualité de service dans les réseaux IP (Almes et al., 1999a ; Seddiki, 2015).

Les conflits dans le routage

La mise à jour des tables de routage peut également engendrer des entrées multiples pour la même destination, dans un contexte de rerouting de flux issus de pannes de liaison multiples non simultanées, à l'aide d'un seul chemin

(Pham et al., 2012). On parle donc de conflit. Ce conflit est la matérialisation dans les tables de routage du type de conflit décrit à la section I.5.3.1.

Ainsi, l'implantation d'une stratégie de mise à jour efficiente des tables de routage dans notre environnement de réseaux virtuels doit pouvoir répondre aux questions suivantes :

- *quels sont les nœuds qui vont être concerné par la mise à jour ?* Répondre à cette question est primordiale dans la mesure où elle permet d'éviter les mises à jour inutiles et des opérations supplémentaires inadaptées pouvant engendrer les problèmes cités ci-dessus ;
- *qui va déclencher le processus de mise à jour des tables de routage ?* Il s'agit de préciser quel sera le nœud initiateur du processus. Dans un environnement centralisé où il existe un contrôleur de réseau (SDN), c'est le contrôleur qui initie ce processus. Par contre, dans les environnements décentralisés, il faut choisir si le processus de mise à jour sera distribué ou pas et le risque encouru avec le distribué c'est la forte probabilité des cycles lorsque la distribution n'est pas bien gérée ;
- *comment vont s'effectuer les mises à jour jusqu'à la résolution du problème ?* Il faut savoir ici si la mise à jour va se faire de proche en proche et de façon séquentielle, ou alors si elle sera distribuée.

Les trois problèmes ainsi présentés et les questions posées constituent des problématiques importantes liées à la qualité de service dans les réseaux virtuels, raison pour laquelle la mise à jour des tables de routage doit être justifiée.

I.6.2. Motivations de la mise à jour des tables de routage

Plusieurs raisons peuvent justifier la nécessité de mise à jour des tables de routage dans les réseaux informatiques en général, et dans les réseaux virtuels en particulier.

La tolérance aux pannes

Elle consiste à garantir la circulation des flux dans le réseau même en cas de pannes de nœuds ou de liaisons. Pour réussir à garantir cette circulation, en général on détermine des chemins alternatifs pouvant supporter le trafic à acheminer jusqu'à la destination considérée ; les différents nœuds intervenant dans ces chemins doivent alors se charger de la transmission de ces flux de proche en proche suivant l'itinéraire choisi ; et puisque cet itinéraire n'est pas toujours celui qui est considéré en l'absence de pannes, il faut donc redéfinir totalement ou partiellement les tables de routage au niveau de ces différents

nœuds. Cela permet d'acheminer efficacement les flux aussi bien en cas de pannes qu'en leur absence.

La saturation des nœuds

Certains nœuds d'un réseau peuvent être énormément sollicités à un moment donné, cela étant dû parfois au nombre élevé des utilisateurs ou alors l'exploitation excessive de ces nœuds lors des mécanismes de reroutage. Quelle que soit la raison, les nœuds ainsi saturés ne peuvent plus fournir une qualité de service acceptable ; au pire des cas, ces nœuds ne permettent plus le transit des flux en leur sein. Dans cette situation, il y a donc nécessité de modifier les tables de routage de ces nœuds avec pour principal objectif de réduire de façon efficiente la quantité de flux qu'ils contrôlent, en envoyant cet excédant vers les nœuds voisins. Dans le contexte des réseaux virtuels, cette décongestion doit tenir compte de l'hétérogénéité des différents plans virtuels tant au niveau de la topologie que des services offerts.

La sécurité

La sécurité dans les réseaux désigne l'ensemble des techniques permettant aux réseaux de faire face aux menaces de diverses natures. Ces menaces peuvent aller du simple piratage d'un nœud du réseau à sa mise hors service complète. Dans le cas de piratage, les nœuds attaqués doivent être isolés du réseau afin de protéger les flux qui ne sont pas encore victimes de ces attaques. Une solution consisterait à dévier tout flux allant vers ces nœuds attaqués, vers des nœuds fiables ; pour y arriver, il faut procéder à la mise à jour des tables de routage dans le réseau.

Ces motivations nous permettent de situer l'importance du problème de mise à jour, raison pour laquelle plusieurs travaux s'y sont déjà penchés. L'état de l'art que nous proposons dans la suite revisite les stratégies de reroutage dans les réseaux IP conventionnels et dans les réseaux à architecture centralisée tel que le SDN (Software Defined Networking). Les tables de routage étant le plus souvent sollicitées lors de l'aiguillage des paquets par les équipements d'interconnexion (routeurs, switch), nous nous intéresserons donc aux différentes méthodes de gestion des tables de routage utilisées par les protocoles fonctionnant en environnement distribué ou centralisé.

I.6.3. Routage en environnement distribué

Dans ce type d'environnement, le routage des paquets est effectué simultanément par plusieurs équipements qui sont indépendants aussi bien en termes de

choix des routes que de réactivité en cas de pannes (liaisons et nœuds). Cette réactivité est assurée par des protocoles de routage distribués se classant en protocoles de routage interne et externe.

I.6.3.1. Protocoles de routage interne

Les protocoles de routage interne visent à faciliter la gestion d'un système autonome en automatisant au maximum la construction des tables de routage. Les routeurs échangent des informations concernant leur vision du réseau par l'intermédiaire d'un protocole de routage ; sur la base de ces informations, ils choisissent le meilleur itinéraire entre deux machines (le plus rapide, le moins cher, le plus fiable, etc.). Plusieurs protocoles de routage interne ont été standardisés :

Le protocole RIP (Routing Information Protocol)

C'est un protocole de routage IP de type Vecteur-Distance (Vector Distance) basé sur l'algorithme de routage décentralisé de Bellman-Ford comme plusieurs travaux existants ([Albrightson, Garcia-Luna-Aceves, & Boyle, 1994](#) ; [Perkins, Belding-Royer, & Das, 2003](#)). Il permet à chaque routeur de communiquer par message aux autres routeurs, la distance (le nombre de sauts *hops* en anglais) ([Hedrick, 1988](#) ; [G. S. Malkin, 1993](#)) qui les sépare dans le réseau IP. Ainsi, lorsqu'un routeur reçoit un de ces messages, il incrémente cette distance de 1 et communique le message aux routeurs directement accessibles. Les routeurs peuvent donc conserver de cette façon la route optimale d'un message en stockant l'adresse du routeur suivant dans la table de routage de telle sorte que le nombre de sauts pour atteindre un réseau soit minimal. La mise à jour des tables de routage consiste donc à incrémenter le nombre de sauts nécessaires pour atteindre les différentes destinations et fixer les prochains sauts. Les routes sont mises à jour toutes les 30 secondes, chaque routeur envoyant à ses voisins ses informations de routage (les réseaux qu'il sait router et les métriques associées). Les routes apprises ne sont ainsi disponibles que pendant 30 secondes au plus.

Dans cette technique de routage, les messages de mise à jour des tables sont portés par des vecteurs de distance à une ou plusieurs entrées qui précisent les distances de tout voisin à un nœud donné, à une destination précise. Un message RIP comprend donc une en-tête suivie de 1 à 25 enregistrement(s) de route (24 si un message d'authentification est requis) ([G. Malkin, 1998](#)). Les structures globale et détaillée d'un message RIP sont respectivement illustrées par les figures 10 et 11.

Dans ces figures :

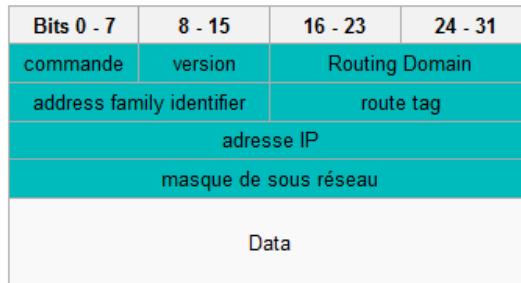


Figure 10 – Structure d'un message RIP.

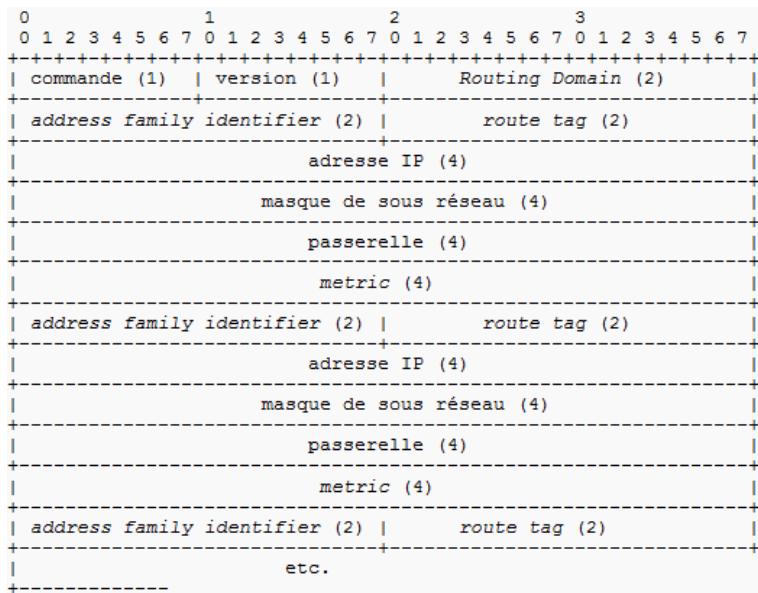


Figure 11 – Exemple de message de routage RIP pour plusieurs destinations (G. Malkin, 1998).

- **commande** représente une requête/réponse ou diffusion ;
- **routing domain** : permet de découper le réseau en sous-réseaux logiques ;
- **route tag** : est un marqueur qui peut être utilisé pour distinguer les routes apprises en interne par RIP de celles apprises par d'autres protocoles (par exemple OSPF) ;
- **metric** : est la distance de la route, comprise entre 1 et 15 ;
- **version** : indique la version du protocole utilisé : version 1 ou 2.

Cependant, l'un des problèmes de performance majeur de cette stratégie est qu'elle met beaucoup de temps pour procéder à la mise à jour des tables de routage des différents nœuds en cas de pannes de nœuds. Ce problème de performance de l'algorithme DBF (Distributed Bellman-Ford) découle du fait qu'il n'a aucun mécanisme inhérent pour déterminer quand un nœud du réseau doit cesser d'augmenter sa distance par rapport à une destination donnée (connu sous le nom de counting-to-infinity problem (Chawda & Gorana, 2015 ; Dugaev, Zinov, Siemens, & Shuvalov, 2015 ; Murthy & Garcia-Luna-Aceves, 1996 ; Woo, Tong, & Culler,

2003)). De plus, cet algorithme augmente les cas de congestion dans le réseau pour les mêmes raisons (Murthy & Garcia-Luna-Aceves, 1996). Néanmoins, nous retenons l'idée de vecteur de distance qui nous sera utile dans notre solution pour transmettre les informations détaillées de mise à jour aux divers noeuds.

Pour éviter les boucles de routage, le nombre de sauts dans le protocole RIP est limité à 15. Au-delà, les paquets sont supprimés. De plus, RIP ne prend en compte que la distance entre deux machines en termes de saut, mais il ne considère pas l'état de la liaison ; ceci est fait dans le but de choisir la meilleure bande passante possible. Si on considère un réseau composé de cinq routeurs A, B, C, D et E reliés tel qu'indiqué à la figure figure 12, alors RIP préférera passer par la liaison directe (A-B) pour joindre C, même si la bande passante n'est que de 52 kbps, alors qu'elle est de 60 kbps sur (A-D) et de 15Mbps sur (D-E) et (E-C).

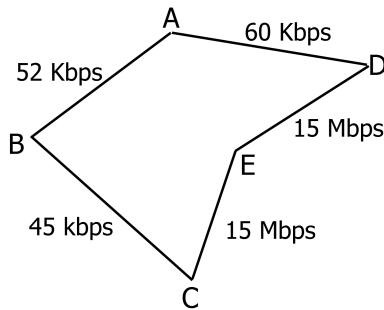


Figure 12 – Mauvaise politique de choix de chemin par RIP.

Ces limitations sont corrigées dans le protocole OSPF.

Le protocole OSPF (Open Shortest Path First)

Open Shortest Path First (OSPF) est un protocole de routage IP interne à un système autonome (Autonomous System) de type protocole à état de liens (link-state protocol) ; c'est à dire qu'il attribue un coût à chaque liaison (appelée lien dans le jargon OSPF) afin de privilégier l'élection de certaines routes. Plus le coût est faible, plus le lien est intéressant. Par défaut, les coûts suivants sont utilisés en fonction de la bande passante du lien.

Dans OSPF, chaque routeur établit des relations d'adjacence avec ses voisins immédiats en envoyant des messages *hello* à intervalle régulier. Chaque routeur communique ensuite la liste des réseaux auxquels il est directement connecté par des messages *Link-State Advertisements (LSA)* propagés de proche en proche à tous les routeurs du réseau. L'ensemble des LSA forme la base de données topologique des liens Link-State Database (LSDB), qui est identique pour tous les routeurs participants. Chaque routeur utilise ensuite l'algorithme de Dijkstra, Shortest Path

First (SPF) pour déterminer la route la plus courte vers chacun des réseaux connus dans la LSDB.

Le bon fonctionnement d'OSPF requiert une complète cohérence dans le calcul SPF. Il n'est donc par exemple pas possible de filtrer des routes ou de les résumer à l'intérieur d'une aire. Chaque entrée dans la table de routage construite par OSPF est indexée par une destination et contient le coût de la destination et un ensemble de chemins à utiliser lors de la transmission des paquets pour atteindre cette destination. Un chemin est décrit par son type (intra-domaine ou extra-domaine) et le prochain saut ; d'où la structure de la figure 13 pour une entrée de la table de routage.

Type de destination	Destination	Aire	Type de chemin	Coût	Prochain(s) saut(s)	Routeur d'information
---------------------	-------------	------	----------------	------	---------------------	-----------------------

Figure 13 – Structure d'une entrée de table de routage OSPF.

Pour la mise à jour des informations de routage, lorsqu'il y a changement d'un état de lien, les routeurs OSPF utilisent un processus d'inondation (flooding) pour avertir les autres routeurs. L'intervalle de mort (dead interval) du protocole *Hello* fournit un mécanisme simple pour déclarer un lien rompu. Quand une interface n'a plus de nouvelles d'un lien après cette période (habituellement 40 secondes ([J. Moy, 1997](#))), le lien est réputé *down* (au sens OSPF). Le routeur qui a constaté le lien *down* envoie par multicast un message LSU (Link State Update) avec les nouvelles informations d'état de lien sur l'adresse Multicast 224.0.0.5 ou 224.0.0.6. Chaque routeur ayant reçu un LSU répond par un LSAck (Link State Acknowledgement). La structure d'un message de mise à jour LSU ([J. Moy, 1997](#) ; [J. T. Moy, 1998](#)) est donnée à la figure 14.

Dès qu'un routeur reçoit un LSU, il met à jour sa link-state database (LSD) et met en œuvre l'algorithme SPF pour calculer les nouvelles routes à inscrire dans sa table de routage. Après l'expiration du compteur SPF, la route est inscrite dans la table de routage. Sur les routeurs Cisco, une vieille route est toujours utilisée pendant que l'algorithme SPF calcule la nouvelle route et jusqu'au moment où le calcul sera achevé.

Le protocole EIGRP (Enhanced Interior Gateway Routing Protocol)

C'est un protocole de routage développé par Cisco à partir de leur protocole original IGRP (Interior Gateway Routing Protocol). EIGRP est un protocole de routage hybride IP (car combinant les protocoles à états de lien et à vecteurs de distance), avec une optimisation permettant de minimiser l'instabilité de routage

0	1	2	3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Version # 4 Packet length			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Router ID			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Area ID			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Checksum AuType			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Authentication			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
Authentication			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
# LSAs			
+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+	+-----+-----+-----+-----+
++		++	
LSAs			
++		++	
...			

Figure 14 – Structure d'un LSU (J. Moy, 1997).

due aussi bien au changement de topologie qu'à l'utilisation de la bande passante et la puissance du processeur du routeur (Albrightson et al., 1994 ; Thorenoor, 2010). Certaines de ces optimisations sont basées sur le Diffusing Update Algorithm (DUAL) développé par SRI (Stanford Research Institute), qui garantit l'absence de boucle. En particulier, DUAL évite les *sauts à l'infini* en les limitant à 224. Pour ce faire, elle associe cinq différentes métriques à chaque route :

- **le délai** : intervalle de temps nécessaire à la transmission d'un paquet de données depuis la source jusqu'à la destination ;
- **la bande passante** : intervalle de fréquence de transmission de données par unité de temps dans un support de communication ;
- **la fiabilité** : désigne le taux de perte des paquets, c'est à dire la probabilité pour laquelle les données n'arrivent pas à destination ;
- **la charge** : représente la quantité de travail effectuée dans le réseau durant la période considérée ;
- **le MTU (Maximum Transfert Unit)** : qui est la taille maximale d'un paquet pouvant être transmis en une seule fois (sans fragmentation) sur une interface.

Le fonctionnement du protocole EIGRP repose sur les éléments tels que :

- **une table de voisins** : Chaque routeur conserve les informations d'état sur les voisins adjacents grâce à l'envoi régulier de messages *Hello*. Lorsque les voisins nouvellement découverts sont atteints, l'adresse et l'interface du voisin sont enregistrées ;

- **une table de topologie** : Il contient toutes les destinations fournies par les routeurs voisins. A chaque entrée de la table, est associée l'adresse de destination et la liste des voisins qui ont annoncé la destination ;
- **la liste des successeurs potentiels** pour une destination donnée ;
- **l'état des routes** : une route peut être active ou passive ;
- **les paquets** : EIGRP distingue cinq types de paquets : le paquet Hello/Acks³, le paquet Update⁴, les paquets Query et reply⁵, le paquet Request.⁶

I.6.3.2. Protocoles de routage externe

Ce sont des protocoles qui échangent des informations de routage entre systèmes autonomes (Autonomous Systems (AS) en anglais)⁷. L'un des plus utilisés est BGP (Border Gateway Protocol). C'est un protocole d'échange de route utilisé notamment sur le réseau Internet. Son objectif est d'échanger des informations d'accessibilité de réseaux (appelés préfixes) entre systèmes autonomes, car il a été conçu pour prendre en charge de très grands volumes de données et dispose de possibilités étendues de sélection. Contrairement aux protocoles de routage interne, BGP n'utilise pas de métrique classique mais base les décisions de routage sur les chemins parcourus, les attributs des préfixes et un ensemble de règles de sélection définies par l'administrateur de l'AS. On le qualifie de protocole à vecteur de chemin (path vector protocol).

BGP prend en charge le routage sans classe et utilise l'agrégation de routes afin de limiter la taille de la table de routage. Depuis 1994, la version 4 (Rekhter et al., 2006a ; Rekhter, Li, & Hares, 2006b) du protocole est utilisée sur Internet, les précédentes étant considérées comme obsolètes.

Les connexions entre deux voisins BGP (neighbours ou peers) sont configurées explicitement entre deux routeurs. Ils communiquent alors entre eux via une session TCP sur le port 179 initiée par l'un des deux routeurs. BGP est le seul protocole de routage à utiliser TCP comme protocole de transport. Une fois la connexion

3. **Le paquet "Hello/Acks"** : paquet d'initiation de communication (Hello) ou d'acquittement (Acks).

4. **Le paquet "Update"** : paquet de mise à jour utilisé pour transmettre l'accessibilité des destinations.

Lorsqu'un nouveau voisin est découvert, les paquets de mise à jour sont envoyés afin que le voisin puisse construire sa table de topologie.

5. **Les paquets "Query" et "Reply"** : sont envoyés lorsque les destinations passent à l'état actif.

6. **Les paquets "Request"** : sont utilisés pour obtenir des informations spécifiques d'un ou plusieurs voisins. Les paquets de requêtes sont utilisés dans les applications de serveur de route. Ils peuvent être multicast ou unicast. Les demandes sont transmises sans relâche.

7. **Un Autonomous System**, abrégé AS, ou Système Autonome, est un ensemble de réseaux informatiques IP intégrés à Internet et dont la politique de routage interne (routes à choisir en priorité, filtrage des annonces) est cohérente. Un AS est généralement sous le contrôle d'une entité unique, typiquement un fournisseur d'accès à Internet. Au sein d'un AS, le protocole de routage est qualifié d'« interne » (par exemple, OSPF). Entre deux systèmes autonomes, le routage est « externe » (par exemple Border Gateway Protocol, abrégé en BGP).

entre deux routeurs établie, ceux-ci s'échangent des informations sur les réseaux qu'ils connaissent et pour lesquels ils proposent du transit. Ils s'échangent aussi un certain nombre d'attributs associés à ces réseaux, et qui vont permettre d'éviter des boucles et garantir une sélection fine de la meilleure route.

Le protocole BGP fonctionne également sous la base de plusieurs messages parmi lesquels le *UPDATE Message* dont le but est d'annoncer de nouveaux itinéraires réalisables qui partagent des attributs de chemin communs à un pair ou de retirer plusieurs itinéraires non réalisables du service (Rekhter et al., 2006a). La structure de ce type de message est présenté à la figure 15.

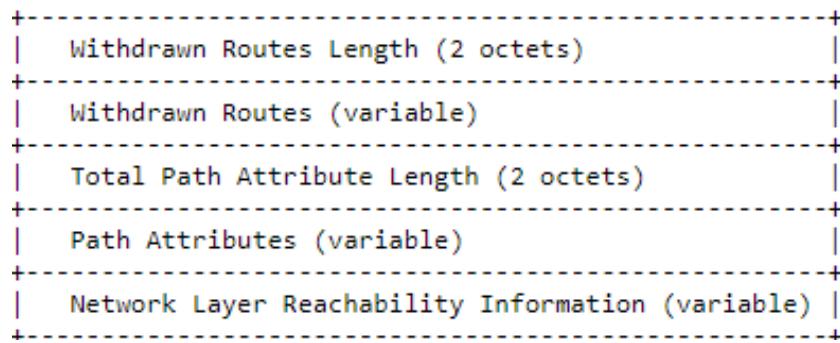


Figure 15 – Structure d'un UPDATE message du protocole BGP (Rekhter et al., 2006a).

Dans cette structure de paquet :

- le champ *Withdrawn Routes Length* : représente la longueur totale des routes retirées du service ;
- *Withdrawn Routes* : désigne un champ de longueur variable qui contient une liste de préfixes d'adresse IP pour les routes retirées du service. Chaque route retirée est donnée sous forme de tuple <longueur (sur 1 octet), prefixe (variable)> ;
- *Total Path Attribute Length* : répertorie les attributs du chemin pour une route possible vers une destination ;
- *Path Attributes* : il s'agit d'une séquence de longueur variable des attributs de chemin qui est présente dans chaque message UPDATE, à l'exception de ceux ne comportant que les routes retirées. Chaque attribut de chemin est un triplet (type d'attribut, longueur d'attribut, valeur d'attribut) de longueur variable (Rekhter et al., 2006a) ;
- *Network Layer Reachability Information* : préfixes d'adresses IP des itinéraires réalisables à informer par le message de mise à jour.

Un des problèmes auquel BGP doit faire face sur Internet est la croissance de la table de routage des routeurs de la *default-free zone* ; c'est-à-dire ceux qui disposent d'une table de routage complète et n'utilisent pas de route par défaut. Avec

le temps, les routeurs plus anciens n'ont plus les ressources mémoire ou CPU nécessaires au maintien d'une table complète. D'autre part, la taille de la table nuit à la vitesse de convergence⁸, le CPU étant particulièrement sollicité lors de changements importants (établissement de nouvelles connexions ou changements importants de topologie) (Bu, Gao, & Towsley, 2004).

Les stratégies de routage et mise à jour décentralisées, utilisées dans les réseaux conventionnels intègrent généralement les étapes de découverte de la topologie et de prise de décision par les noeuds intervenant dans le processus ; mais, dans les réseaux SDN, le contrôleur a une vue globale sur le réseau et peut prendre les décisions de gestion du trafic. Les protocoles de routage décentralisés ne sont donc pas conseillés dans ce type d'environnement.

I.6.4. Routage en environnement centralisé

Dans le standard IEEE 802.11s définissant les spécifications pour l'implémentation des réseaux numériques locaux à liaison sans fil (Hiertz et al., 2010), un protocole de routage basé sur une topologie d'arbre (en anglais Tree-Based Routing abrégé TBR) est adopté en tant que protocole de routage proactif viable pour un réseau sans fil maillé. Dans le protocole TBR, chaque noeud maintient à jour sa table de routage et il existe un noeud central qui supervise le routage des paquets. Lorsqu'un noeud source souhaite envoyer du trafic vers un noeud de destination qui se trouve à l'intérieur du réseau maillé, le noeud source transfère le trafic vers la racine si aucun chemin actif n'existe dans sa table de routage, et le noeud destinataire reçoit le trafic transmis par la racine. Lorsqu'un noeud veut effectuer la mise à jour de sa table de routage, il collecte les informations de ses voisins pendant un temps prédéfini et utilise ces informations pour mettre à jour les informations telles que le prochain saut dans l'arbre de routage (Next Hop NH), la métrique de la liaison (Link Metric LM), le numéro de séquence (Sequence Number SN), etc. Ces messages sont appelés messages RANN (Root ANNouncement) et leur structure est donnée par la figure 16.

Type	Length	Flag	TTL	Lifetime	Metric	Root Address	Root Sequence Number
1	1	1	1	4	4	4	4

Figure 16 – Structure d'un message RANN (Lim et al., 2008).

8. **La vitesse de convergence** : désigne le temps que mettront l'ensemble des routeurs du réseau à faire l'apprentissage de leur environnement, ou à réagir à un changement du réseau, pour le mettre à jour dans leurs tables de routage.

Les inconvénients du protocole TBR liés à la mise à jour des tables de routage sont les suivants :

- le délai d'attente des messages des voisins ne permet pas d'avoir forcément des informations de mise à jour optimales ; un nœud peut transmettre des informations qui n'arrivent pas avant l'épuisement du délai de réception des informations des voisins ;
- beaucoup de messages doivent être échangé avec les voisins pour mettre en place le procédé de mise à jour ;
- cette stratégie considère que chaque noeud a la possibilité de mettre en place des politiques de contrôle personnalisé. Dans l'architecture que nous utilisons, le SDN, les nœuds n'ont pas cette capacité.

Par ailleurs, [Lim et al. \(2008\)](#) proposent une approche hybride (pseudo-centralisée) de routage semblable à celle précédente en mettant en place une meilleure approche de détermination des chemins optimaux de routage des paquets et de mise à jour des nœuds. Dans cette approche, la mise à jour au niveau d'un nœud peut être déclenchée par le nœud lui-même ou le nœud racine. Lorsqu'un nœud veut mettre sa table de routage à jour à la suite d'une panne ou d'une perte de connectivité avec son nœud racine direct dans l'arbre de routage, il envoie un Route Request (RREQ) message à destination du nœud racine principal de l'arbre ; lorsque ce dernier reçoit le RREQ message, il détermine le meilleur chemin qui permettra de reconnecter ce nœud à son nœud racine direct ; ensuite un Route SET (RSET) message est envoyé à destination du nœud à mettre à jour. Le RSET contient la liste des nœuds intermédiaires à traverser, avec les informations à propos des nœuds à suivre pour atteindre la destination des paquets envoyés par le nœud source. Le RSET message a la structure présentée par la figure 17. Ce protocole permet certes de fournir à chaque fois les chemins de routage de flux optimaux, mais il ne peut mettre à jour qu'un nœud à la fois. De plus, la mise à jour d'un nœud est précédée d'un ensemble de messages RREQ envoyé par le nœud racine pour la découverte de la topologie du réseau, ce qui contribue à encombrer le réseau.

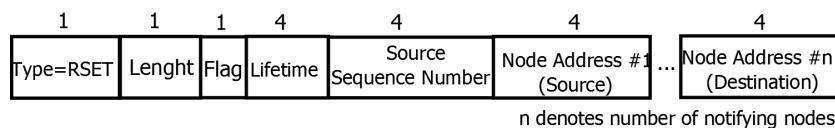


Figure 17 – Structure d'un message RSET ([Lim et al., 2008](#)).

Concernant la question de mise à jour des tables de routage dans un contexte de virtualisation réseau intégrant la technologie SDN en présence de panne persistante d'une liaison, [Son \(2014\)](#) propose de faire les mises à jours des tables de routage de

la destination vers la source de l'arbre de routage, inspiré des travaux de [Lambert, Buob, et Uhlig \(2009\)](#); cela permet de conserver les flux entrants et sortants d'un nœud et d'éviter de perturber le trafic en cours; la figure 18 illustre ce procédé de mise à jour. En effet, lorsque la panne de liaison (S-D) se produit, le nœud S reroute les flux suivant le chemin $S \rightarrow T \rightarrow U \rightarrow K \rightarrow M \rightarrow N$ pour reconnecter le réseau. Lorsque la durée de la panne (S-D) atteint 10 min, elle est donc persistante; [Son \(2014\)](#) propose de mettre d'abord à jour les nœuds W, Y et M dans cet ordre, puis suivront les nœuds du chemin de reroutage présenté par la figure 7c dans l'ordre $K \rightarrow U \rightarrow T \rightarrow S$. Si le nœud W n'est pas mis à jour, les paquets seront transférés à S via l'ancien arbre de routage, risquant de créer des conflits.

Mais, cette stratégie de mise à jour de [Son \(2014\)](#) met à jour des nœuds dont la mise à jour n'est pas absolument nécessaire. Par exemple, pour la figure 18c, on aurait pu parmi les deux nœuds Y et W, mettre à jour uniquement le nœud W puisqu'il est le père des nœuds X et Y dans l'arbre nominal rouge; de ce fait, les nœuds fils du nœud W seraient alors redirigés suivant la liaison (W-R). De plus, cette stratégie surcharge davantage le chemin de reroutage par les flux de l'arbre nominal rouge.

Les pannes liaison et de nœud ainsi présentés dans cette section engendre des variations dans le trafic réseau, ce qui peut conduire à des surcharges. La mobilité des utilisateurs peut davantage accentuer cette surcharge, surtout quand on sait que les services sont généralement proposés par des serveurs dont les ressources sont limitées.

I.7. Congestion du trafic

La congestion est la condition dans laquelle une augmentation du trafic (flux) provoque un ralentissement global de celui-ci. Les trames entrantes dans les buffers des commutateurs sont rejetées dans ce cas. Elle apparaît lorsque le débit du flux d'entrée (ou la somme des débits des flux d'entrée) du réseau est supérieure au débit utile de sortie. Ce phénomène a été observé dès la première phase de croissance d'internet en 1980. L'une des principales causes de la congestion dans les réseaux IP en général est la nature sporadique des trafics qui évolue au rythme de la mobilité des utilisateurs dans le réseau et les évolutions de la topologie.

Dans chaque réseau virtuel, plusieurs types de services peuvent être fournis aux utilisateurs. Chaque service est fourni par l'intermédiaire d'un serveur virtuel qui gère la charge du trafic lié à ce service ([Horiuchi & Tachibana, 2018](#)). La qualité du service est affectée par la quantité de trafic sur chaque réseau virtuel. Lorsque

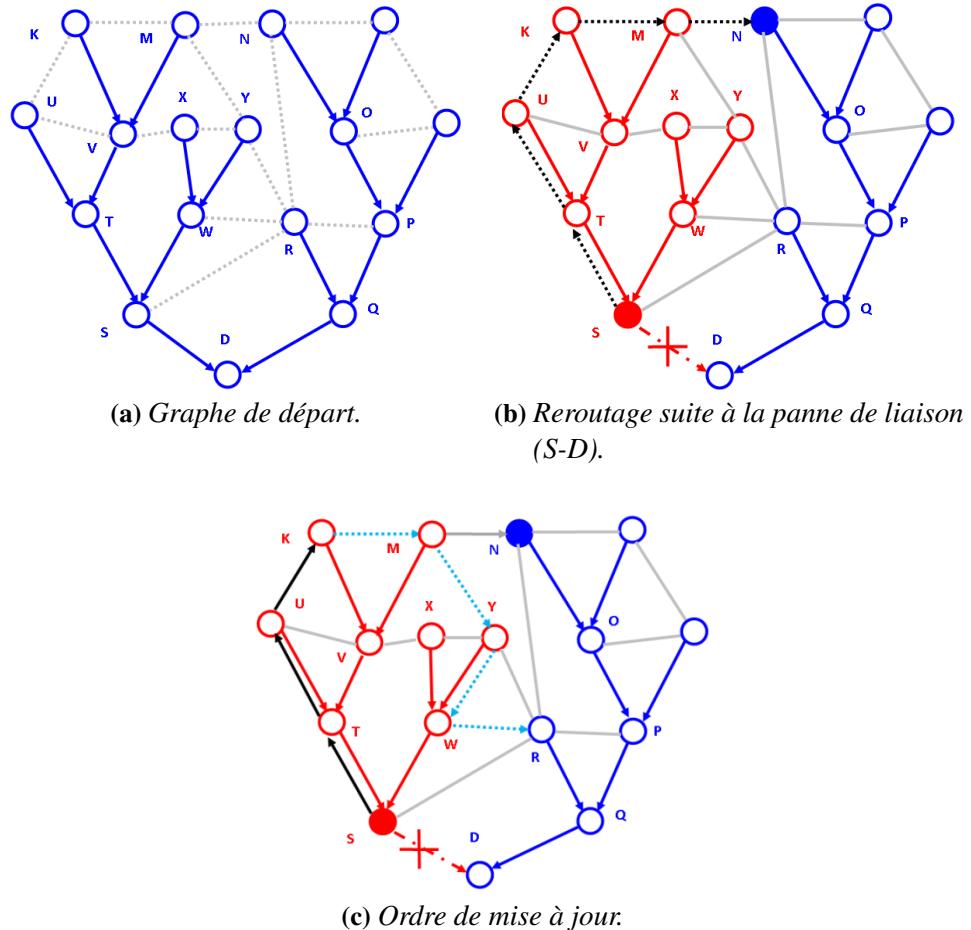


Figure 18 – Technique de mise à jour proposée par Son (2014) : Cas de pannes simples (Son, 2014).

la quantité de trafic est importante ou petite dans le réseau virtuel, la qualité du service fourni est élevée ou faible. De plus, si de nouveaux nœuds sont ajoutés ou si des nœuds existants sont supprimés du réseau virtuel, la qualité du service fourni change de manière plus significative. Avec la propagation des terminaux mobiles, les réseaux virtuels sont également utilisés par les utilisateurs mobiles ; ce qui implique que la quantité de trafic et la topologie du réseau peuvent être modifiées dynamiquement si certains utilisateurs passent fréquemment d'un point d'accès à un autre point d'accès. Plusieurs travaux ont essayé dans la littérature de résoudre ce problème au fil des années.

I.7.1. Techniques d'approvisionnement en ressources

L'approvisionnement consiste à mettre en place une approche de protection du réseau virtuel contre les congestions et pannes en réservant à l'avance des ressources du réseau physique. Ainsi, lorsqu'un serveur est congestionné, les res-

sources réservées sont mobilisées pour fournir à ce nœud les ressources nécessaires à la prise en main de l'excédent de trafic au niveau du serveur. Cette piste se ramène à un pur problème d'allocation de ressources du réseau physique aux réseaux virtuels. L'idée est en général de trouver la meilleure répartition de ressources qui intègre toutes les contraintes liées aux réseaux virtuels et les utilisateurs en présence. C'est un problème connu comme NP-difficile (Haider, Potter, & Nakao, 2009). Néanmoins plusieurs travaux (Pham et al., 2012 ; Seddiki, 2015 ; Shahriar et al., 2017 ; Yankam & Kengne Tchendji, 2020) ont déjà proposé des modèles ILP (Integer Linear Program) et des heuristiques avec réduction du nombre de contraintes, permettant de le modéliser et d'avoir quelques solutions approchées.

L'inconvénient commun des méthodes proposées dans les travaux liés à l'approvisionnement en ressources des réseaux virtuels est qu'ils réduisent davantage les ressources physiques disponibles au profit des réseaux virtuels ; de plus, une stratégie de gestion de trafic par réservation de ressources crée à la longue une mauvaise répartition des ressources réseau, même si le modèle d'allocation de ressource initial est efficace (Doumith, 2007).

I.7.2. Repositionnement des serveurs de services virtuels

Cette méthode de gestion des congestions consiste à faire migrer un service virtuel de son serveur source vers un autre nœud offrant des ressources appropriées pour gérer l'excédent de trafic. Cette approche n'exige pas de ressource supplémentaire de la part du réseau physique de base. Contrairement à la méthode d'approvisionnement, cette solution a été peu explorée jusqu'ici.

Dans le domaine des réseaux virtuels à topologie arborescente, Koyanagi et Tachibana (2014) proposent une technique de repositionnement dynamique des ressources du réseau virtuel pour satisfaire la QoS fournie lorsque le changement de trafic est dû à l'ajout d'un nouveau nœud dans le réseau. L'idée est de faire migrer successivement d'un saut, le serveur virtuel de son nœud de départ vers un autre qui a suffisamment de ressources, de manière à rapprocher le plus possible le service du nouveau nœud ajouté dans le réseau. À chaque migration vers un nœud, il est déterminé si la qualité de service est satisfaisante avant de passer au nœud suivant. Le nouveau nœud hôte du service virtuel est celui situé à un saut du nouveau nœud ajouté dans le réseau. La figure 19 illustre cette méthode de repositionnement. Lorsque le nouveau nœud (voir figure 19a) est ajouté dans la topologie, la ressource de service virtuel qui est initialement située dans le nœud n2 est repositionné dans le nœud n6 à un saut du nœud n7 à travers le chemin le

plus court (voir figure 19b). Le principal inconvénient de cette méthode c'est le temps de repositionnement du service de réseau virtuel.

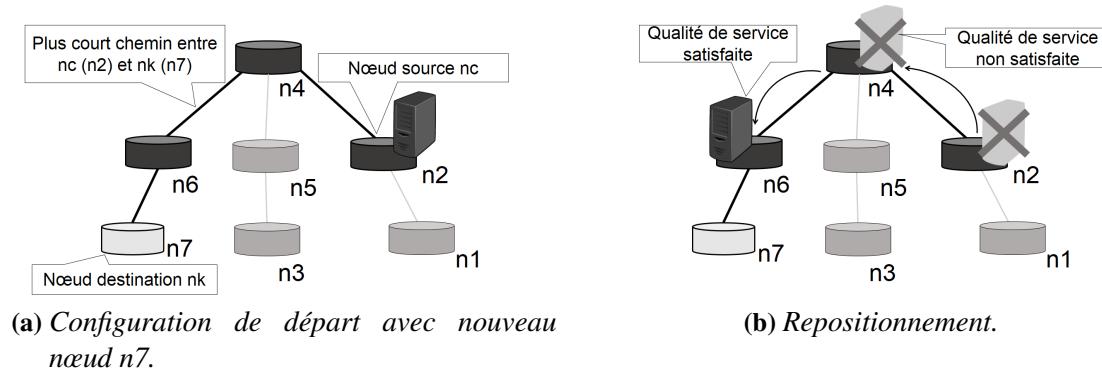


Figure 19 – Repositionnement de serveur virtuel selon Koyanagi et Tachibana (2014).

Afin d'améliorer l'approche proposée par Koyanagi et Tachibana (2014), Horiuchi et Tachibana (2018) proposent une nouvelle méthode de repositionnement dynamique de ressources de services virtuel. Cette méthode ne vérifie pas si la qualité de service est satisfaite à chaque nœud entre n_c et n_k , mais uniquement à la fin du processus. Cela implique un temps de repositionnement de ressources de services virtuel plus court. L'idée est de sélectionner dans l'arbre, le nœud qui permet de contrôler le plus de trafic possible. L'algorithme de recherche commence par sélectionner parmi les nœuds feuilles, celui qui a la quantité γ_i minimale de trafic encours ; ensuite, ce nœud est retiré du processus et son poids est ajouté à celui de son nœud père dans l'arborescence. Le procédé continu jusqu'à ce que l'arbre ne contienne plus que deux nœuds et dans ce cas le nouveau nœud devant servir de serveur est celui dont le poids de trafic est maximal. Une illustration de cette approche de sélection est présentée par la figure 20.

A l'issue du processus de recherche pour le cas de la figure 20, les deux nœuds restants sont n_1 et n_2 . Le nœud n_1 est donc sélectionné comme nouveau serveur virtuel, puisqu'il peut supporter une quantité de trafic égale à 40 par rapport à n_2 qui ne peut en supporter que 35. Par ailleurs, Horiuchi et Tachibana proposent également deux autres améliorations importantes :

- une restructuration du réseau sous forme de topologie arborescente lorsque la topologie initiale change après l'ajout ou la suppression d'un nœud dans un réseau virtuel ;
- le repositionnement de 2^n serveurs virtuels.

Néanmoins, la méthode de Horiuchi et Tachibana (2018) présente plusieurs manquements importants :

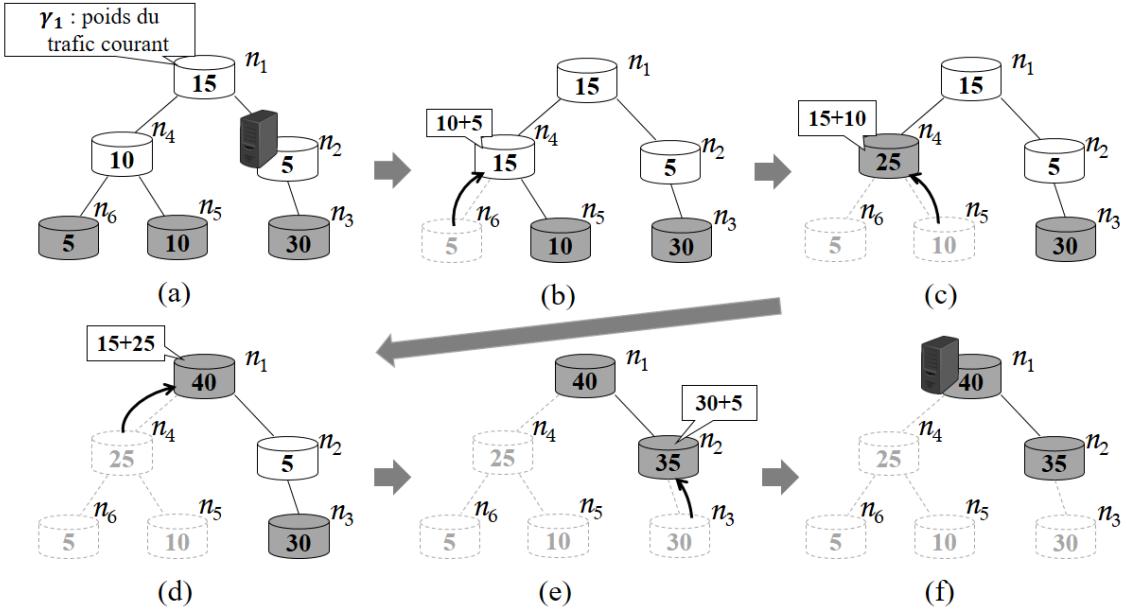


Figure 20 – Repositionnement de serveur virtuel selon Horiuchi et Tachibana (2018).

1. Absence de solution de repositionnement lorsque tous les nœuds feuilles de l'arbre ont le même poids de trafic

Dans ce cas de figure, l'algorithme est buté dès le début de son exécution et le processus de recherche ne produit rien en sortie. La figure 21 illustre cette limite. Pour le cas illustré, les nœuds feuilles F, E et C ont tous un poids de trafic courant égal à 12. Lequel faudrait-il donc sélectionner ? L'approche de Horiuchi et Tachibana (2018) ne permet pas de répondre à cette question.

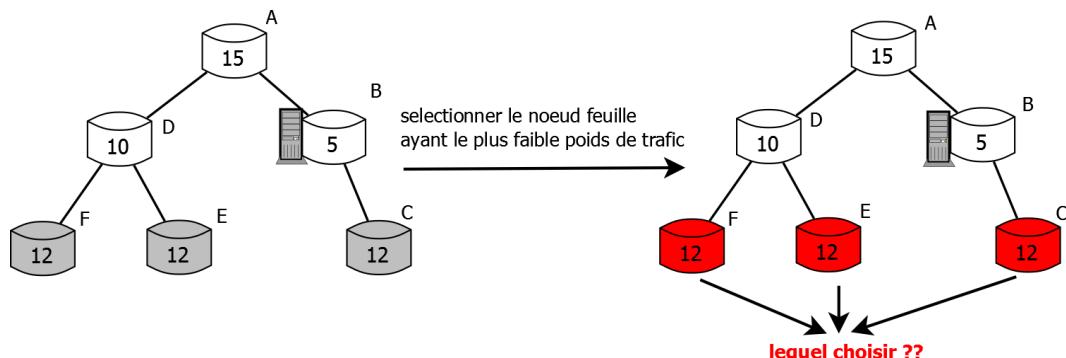


Figure 21 – Problème d'égalité de poids de trafic dans la méthode de Horiuchi et Tachibana (2018).

2. trop de mobilités de services en fonction des variations de trafic

Chaque fois qu'une charge de trafic importante est détectée sur un serveur, un repositionnement est effectué. Ces multiples repositionnements créent une

instabilité du service réseau offert ainsi qu'un déséquilibre de charge au niveau des nœuds du réseau ;

3. le service n'est pas repositionné lorsque le trafic diminue ou redevient normal

Le repositionnement du service virtuel dans un autre nœud entraîne l'utilisation des ressources de ce nœud qui sont prévu en principe pour un service bien précis. L'exploitation de ces ressources de façon inattendue, est donc susceptible de créer des perturbations dans le service proposé par le nœud ; il y a donc nécessité d'utiliser temporairement les ressources pour gérer la surcharge du trafic et de repositionner le service ensuite dans son nœud de départ. Ceci permettrait d'éviter la surutilisation des ressources de certains nœuds. [Horiuchi et Tachibana \(2018\)](#) n'envisagent pas le retour du service virtuel à son nœud de départ après repositionnement de ce dernier ;

4. la stratégie de migration proprement dite du service d'un nœud à un autre (pré-copie, post-copie, hybridation) n'est pas prise en compte

Le repositionnement du serveur de service virtuel met en jeu le transfert d'un ensemble de données d'une machine virtuelle vers une autre ; ces données sont caractéristiques de l'état du service qui est transféré, afin qu'il soit restauré plus efficacement dans le nouveau nœud physique ou virtuel qui le recevra. Ces caractéristiques peuvent notamment être ([Kherbache, 2016](#)) : la quantité de mémoire volatile nécessaire, la mémoire non-volatile, l'état des CPUs virtuels, l'état des périphériques connectés et l'état des connections actives. Ces éléments qui caractérisent le service prennent en réalité du temps pour être acheminés d'un nœud virtuel à un autre et demandent la mise en place d'une approche de migration adaptée pour conserver la cohérence du service rendu aux utilisateurs finaux. Selon les cas, on peut envisager une migration des fichiers du service ou alors une migration du système d'exploitation hébergeant le service. Dans les deux cas, le temps de migration influence la disponibilité du service ; d'où la nécessité de prendre en compte ce temps de migration et de le minimiser.

5. la structure d'arbre utilisée ne permet pas d'explorer plusieurs autres possibilités de repositionnement

Dans la structuration sous forme d'arbre, les données de migration ne peuvent être acheminées que du nœud fils vers le nœud père et vice versa ; ce qui entraîne une mauvaise utilisation des ressources du réseau.

Au chapitre IV, nous proposons des protocoles de repositionnement de serveurs virtuels qui résolvent les limites ci-dessus évoquées.

I.8. Conclusion

Dans ce chapitre, il était question pour nous de fixer les bases terminologiques et techniques nécessaires à la compréhension de notre travail. C'est ainsi que nous avons passé en revue de façon détaillée, quelques travaux importants liés à la gestion de la qualité de service dans les réseaux virtuels face aux pannes de diverses sortes. Nous avons présenté les limites des solutions existantes, qui n'offrent pas une bonne QoS aux utilisateurs finaux lorsque le réseau est perturbé. Il en ressort principalement que les stratégies de reroutage pour la tolérance aux pannes dans les réseaux virtuels se classent en deux grandes catégories : d'une part nous avons celles qui essaient d'anticiper la panne en réservant à l'avance les ressources nécessaires auprès du réseau physique (méthodes proactives), ce qui permet de déclencher le reroutage des flux plus rapidement ; d'autre part, nous distinguons celles qui se mettent en place uniquement lorsque la panne survient (les méthodes réactives). Néanmoins, ces deux approches mettent un accent particulier sur la minimisation des ressources physiques mobilisées pour le reroutage ; ceci en raison du partage des ressources de l'infrastructure physique entre plusieurs réseaux virtuels hétérogènes.

Tout au long de ce travail, nous apporterons des solutions aux divers manquements des méthodes de gestion de trafic présentés dans ce chapitre. Nous commençons par le problème de pannes de liaisons et de noeuds dont nous abordons les questions de la détection et de la prise en main en environnement SDN, en fonction de leur nature tout d'abord temporaire puis persistante. Le chapitre II présentera notre solution de reroutage pour le cas de la panne temporaire d'une liaison.

II

CHAPITRE

NOTRE MÉTHODE DE REROUTAGE PAR SÉPARATION DE TRAFIC

SOMMAIRE

II.1 - Introduction	49
II.2 - Reroutage multichemins sans conflit dans les réseaux à base de com- mutateurs	50
II.3 - Modèle mathématique	57
II.4 - Analyse et interprétation des résultats de simulations	66
II.5 - Conclusion	72

II.1. Introduction

Dans ce chapitre, nous présentons notre approche de reroutage par séparation de flux pour la tolérance aux pannes de liaisons dans les réseaux virtuels. La séparation de flux consiste à utiliser deux ou plusieurs chemins alternatifs pour rediriger le trafic d'une liaison en panne (Veerasamy et al., 1994). Le flux peut être subdivisé en deux parties (on parle de *dichotomie de flux*) ou en plusieurs parties suivant le nombre de liaisons alternatives offrant un total de capacité résiduelle suffisante pour supporter ce trafic. Dans l'objectif d'améliorer l'approche de Son (2014), la technique de séparation de flux que nous proposons ici vise non seulement à minimiser davantage la capacité additionnelle requise pour les liaisons, mais également d'utiliser autant que possible la capacité résiduelle disponible au niveau des arcs concernés. L'intérêt de cette minimisation est la réduction de l'impact de la stratégie de tolérance aux pannes sur le réseau physique.

Nous commençons par présenter la technique de séparation de flux que nous illustrons via un exemple concret, puis nous formalisons cette solution dans un

contexte de panne de liaison simple ; ensuite, nous proposons un modèle mathématique permettant de valider l'existence de chemins pour une stratégie de rerouting sans conflits dans un cas de panne ponctuelle d'une liaison.

II.2. Rerouting multicemins sans conflit dans les réseaux à base de commutateurs

Dans cette section, nous décrivons notre approche de rerouting avec séparation de flux pour la tolérance aux pannes d'une seule liaison. Nous supposons que le réseau est représenté par un graphe orienté $G(N, L)$, où $|N|$ désigne le nombre de nœuds et $|L|$ le nombre de liaisons. Pour fixer les bases de cette partie de notre travail, nous émettons les hypothèses suivantes :

- Il existe au moins deux chemins disjoints entre toute paire de nœuds du réseau ;
- le trafic est assuré d'un point à un autre via un arbre appelé arbre de routage nominal utilisant le plus court chemin déterminé grâce à l'algorithme de Dijkstra ;
- le calcul des chemins est fait à l'avance pour chaque situation de panne et la stratégie de rerouting est implémentée dans le contrôleur SDN, permettant ainsi d'éviter les calculs et les reconfigurations à l'intérieur du réseau au cours de son fonctionnement ;
- toute panne de liaison est détectée par le nœud en amont du flux dans cette liaison, et le rerouting est initié à partir de ce nœud (routage local), afin de limiter le nombre de nœuds impliqués dans le rerouting ;
- une seule panne de liaison se produit à un moment donné.

II.2.1. Illustration de notre solution utilisant la séparation de flux

La séparation de flux utilisée dans l'objectif d'améliorer l'approche IPFRR (IP Fast ReRoute) de [Son \(2014\)](#) induit l'exploitation de plusieurs ponts (liaisons permettant de relier le sous arbre rouge au sous arbre bleu) pour rerouter le trafic. Nous allons d'abord illustrer la séparation de flux basée uniquement sur l'utilisation de la capacité résiduelle, ensuite nous la présenterons telle que nous la proposons avec minimisation de la capacité additionnelle.

II.2.1.1. Exemple illustratif de la séparation de flux sans utilisation de capacité additionnelle

Considérons la figure 22 représentant un réseau de 13 nœuds et 25 liaisons. La destination de l'arbre de routage nominal est le nœud 1. Supposons que le chemin $11 \rightarrow 8 \rightarrow 5 \rightarrow 4 \rightarrow 1$ transporte un poids de trafic égal à 8. En cas de panne de la liaison (5-4) telle qu'indiqué sur la figure 22, on doit pouvoir rerouter le trafic de poids égal à 8.

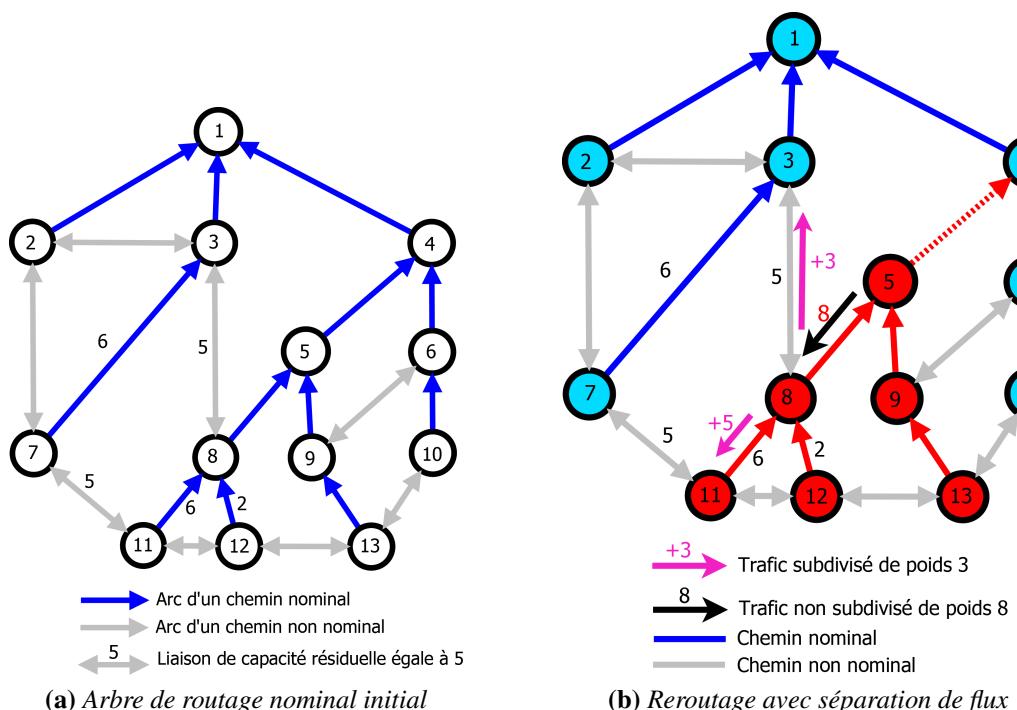


Figure 22 – Rerouting sans capacité additionnelle.

D'après Son (2014), il faut déterminer un pont possédant une capacité résiduelle suffisante pour contenir ce poids. Ainsi, on choisirait comme pont la liaison (8-3) ou (8-11). Cependant, aucun de ces ponts n'offre une capacité résiduelle suffisante pour supporter un trafic de poids 8. Pour que la rerouting de ce trafic soit donc possible, au niveau du nœud 8, on divise ce trafic en deux parties de poids 3 et 5. Le trafic de poids 3 est acheminé à travers la liaison (8-3) et celui de poids 5 sur la liaison (8-11).

La grande partie du trafic est envoyée sur le pont qui offre la plus grande capacité résiduelle (8-11). La clause des conflits que nous voulons éviter empêche la sélection de l'arc (8→12) comme pont, puisqu'elle conduit à un nœud du graphe G_r . De plus, chaque nœud accueillant une partie du trafic va suivre le chemin nominal partant de ce nœud jusqu'au nœud de destination. Ainsi, lorsque le nœud 3

reçoit le trafic de poids 3, il suit son chemin nominal partant du nœud 3 au nœud 1 ; il en est de même pour le trafic de poids 5.

Afin d'éviter les cycles dans le reroutage, chaque partie du trafic subdivisé ne doit emprunter aucun arc du chemin nominal précédemment emprunté par le trafic originel ; c'est-à-dire par exemple que lorsque la partie de trafic de poids 5 arrive sur l'arc $(8 \rightarrow 11)$, il ne doit pas emprunter l'arc $(11 \rightarrow 8)$ qui est un arc du chemin nominal, mais doit emprunter le chemin $11 \rightarrow 7 \rightarrow 3 \rightarrow 1$. Nous pouvons garantir l'acheminement du trafic subdivisé jusqu'à destination grâce aux contraintes de continuité du trafic présentés dans (Son, 2014).

L'exemple de la figure 22 montre que la séparation de flux utilisant uniquement la capacité résiduelle peut permettre de faire face au problème d'insuffisance de cette capacité résiduelle lors d'un reroutage unicemin. Cependant, on peut se retrouver bloqué au niveau de certaines liaisons devant reconstituer les flux subdivisés ou alors faire face à de multiples subdivisions, ce qui diminuerait davantage les chances de trouver un chemin de reroutage. Dans le cas de l'exemple de la figure 22, il peut arriver que l'arc $(3 \rightarrow 1)$ n'ait pas la capacité résiduelle nécessaire pour supporter les deux flux provenant des arcs $(7 \rightarrow 3)$ et $(8 \rightarrow 3)$ résultants de la subdivision du flux de la liaison $(5-8)$; dans ce cas, la séparation du flux reposant exclusivement sur la capacité résiduelle des liaisons, ne résout que temporairement le problème. Avec notre séparation de flux améliorée, nous pouvons faire face à ce genre de situation, ceci en utilisant la capacité additionnelle au niveau de l'arc $(3 \rightarrow 1)$ si cela s'avère nécessaire.

II.2.1.2. Exemple illustratif de notre méthode de séparation de flux avec utilisation conjointe de capacité additionnelle et capacité résiduelle

Dans cette section, nous illustrons notre méthode de séparation de flux via un exemple. Pour ce faire, considérons la figure 23 représentant un réseau avec 6 nœuds et 8 liaisons.

La figure 23a présente le graphe originel dont la capacité résiduelle de chaque liaison est représentée. Supposons que le nœud source soit de numéro 6 et le nœud destination de numéro 1. Nous supposons ici qu'une panne de la liaison $(4-1)$ engendre un flux de 15 à rerouter. La figure 23b représente le poids de chaque liaison et la capacité additionnelle dont on aurait besoin au niveau de chaque arc si le flux rerouté venait à utiliser cet arc. En utilisant l'heuristique de Son (2014) (voir figure 24a), on aboutirait au choix du chemin $4 \rightarrow 6 \rightarrow 3 \rightarrow 1$ comme chemin de reroutage ;

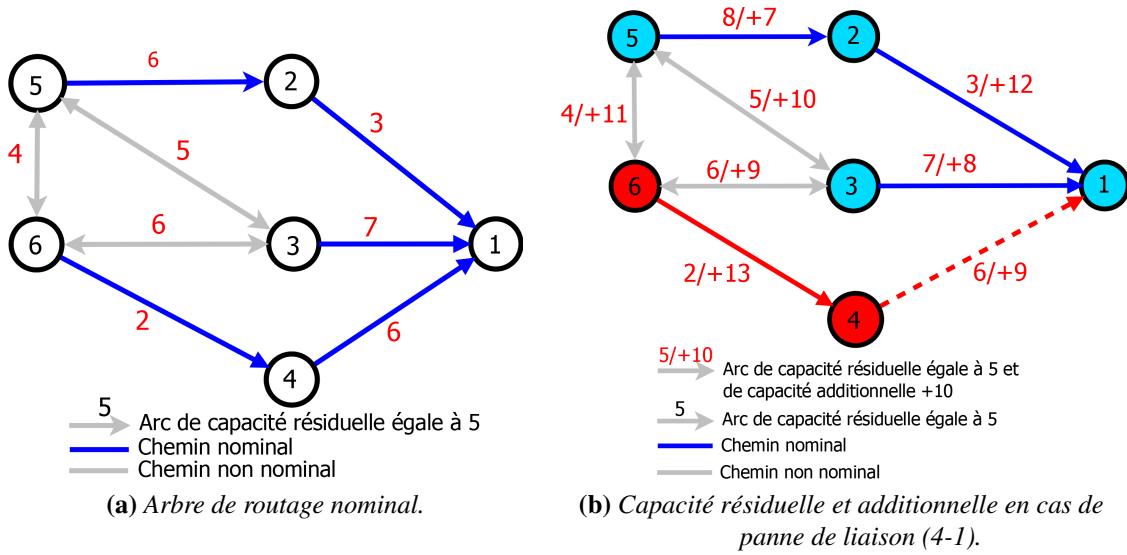


Figure 23 – Réseau avec capacité résiduelle et additionnelle.

ce chemin équivaut à une utilisation d'un total $T = 13 + 9 + 8 = 30$ de capacité additionnelle.

En appliquant notre séparation de flux (voir figure 24b), la partie $4 \rightarrow 6$ du chemin précédent sera conservée. A partir du nœud 2, nous pouvons effectuer la séparation du flux en deux parties puisqu'il y a deux arcs qui partent du nœud 6 en direction du nœud 1. Le critère qui régira notre séparation de flux c'est la capacité résiduelle disponible au niveau des liaisons. Ainsi, on va envoyer plus de flux sur la liaison qui offre la plus grande capacité résiduelle et l'autre partie de ce flux sera envoyée sur l'autre liaison. De ce fait, on enverra un flux égal à 6 sur l'arc $(6 \rightarrow 3)$ (qui est un pont) offrant une capacité résiduelle égale à 6, et 9 sur l'arc $(6 \rightarrow 5)$ (un autre pont) qui offre une capacité résiduelle de 4. Ceci induit l'utilisation sur l'arc $(6 \rightarrow 5)$ d'une capacité additionnelle égale à 5 en plus de sa capacité résiduelle existante. Les heuristiques permettant d'effectuer ces séparations sont présentées à la section II.3.3.

En fin de compte, pour notre séparation de flux, les chemins de reroutage sont : 4-6-3-1 utilisant un total de $13 + 0 + 0 = 13$ de capacité additionnelle, et le chemin $4 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 1$ utilisant $13 + 5 + 1 + 6 = 25$ de capacité additionnelle. Puisque l'arc $(4 \rightarrow 6)$ appartient aux deux chemins de reroutage et induit une capacité additionnelle de 13, la capacité additionnelle totale nécessaire pour ces deux chemins de reroutage est de 25, soit une économie de 17% sur la capacité additionnelle ajoutée par rapport à la méthode de Son (2014). Cet exemple montre ainsi qu'en utilisant notre méthode de séparation de flux, nous pouvons minimiser davantage la capacité additionnelle au niveau des liaisons de reroutage.

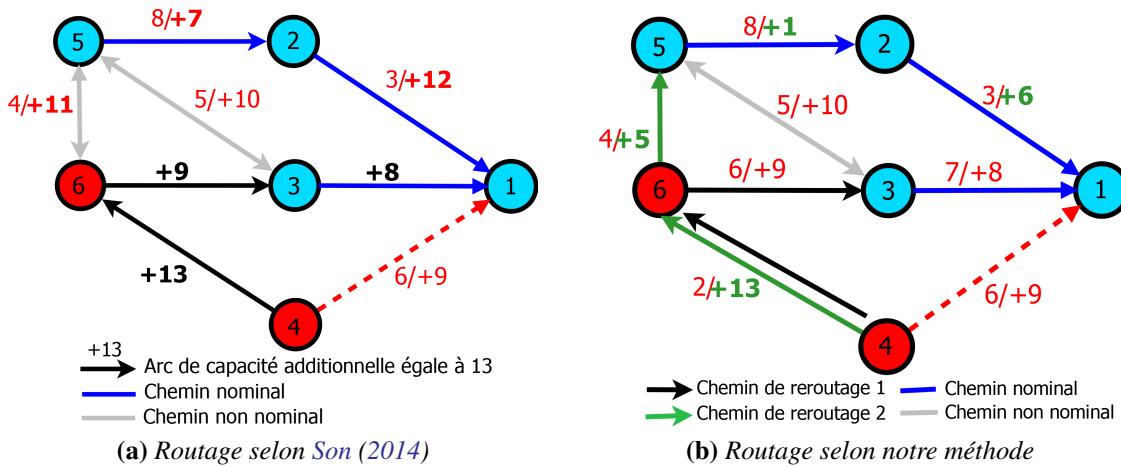


Figure 24 – Rerouting de Son (2014) vs notre méthode.

II.2.2. Existence des chemins de reroutage

Dans cette section, nous étudions l'existence d'une réelle solution de rerouting multicamins sans conflit de rerouting. Sous les hypothèses précédentes, nous émettons le théorème suivant pour valider l'existence de cette solution.

Théorème 1 *Pour toute destination d , il existe un plan de rerouting sans conflit utilisant au moins deux chemins.*

Preuve. Considérons un arbre de routage Rt vers une destination d comme indiqué sur la figure 25 ; d est la racine de R_t . Les liaisons en pointillés représentent l'existence possible de nœuds entre les nœuds connectés via ces liaisons. Soit $(p_1 \rightarrow q_1)$ un arc de Rt . Supposons que cet arc tombe en panne et que nous devons trouver un schéma de rerouting sans conflit. Notons que $(q_1 \rightarrow p_1)$ n'appartient pas à Rt , ce qui signifie que pour une destination donnée, la panne de liaison peut être traitée comme une panne d'arc. Sans nuire à la généralité, nous considérerons le problème de rupture de l'arc $(p_1 \rightarrow q_1)$. p_1 est la racine d'un sous-arbre Rt_1 dont les nœuds sont colorés en rouge, c'est à dire G_r . Les autres sommets de l'arbre sont colorés en bleu (G_b). Nous supposons que tous les sommets font partie de l'arbre Rt et que cette hypothèse est vraie pour toute destination d .

D'après Son (2014), il existe un chemin de rerouting reliant G_r à G_b . À l'intérieur de la partie rouge du sous-arbre enraciné en p_1 , sous l'hypothèse d'une connectivité à deux liaisons, il y a au moins deux chemins allant de n'importe quel nœud de cette partie rouge à un arc reliant à la fois les parties rouge et bleue. Ainsi, si la séparation du trafic est effectuée sur l'un de ces nœuds, il sera possible de rerouter le flux à travers au moins deux chemins différents jusqu'à ce qu'il atteigne la destination d . En d'autres termes, il existe au moins deux chemins μ et ν

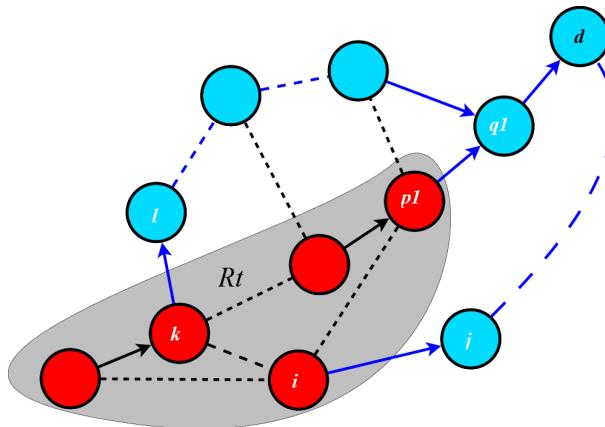


Figure 25 – Preuve d’existence de plusieurs chemins de reroutage.

partant de $p1$, parcourant les sommets de Rt_1 et reliant un sommet de Rt_1 qui est rouge à un sommet bleu. Cette connexion peut être effectuée via des arcs ($i \rightarrow j$) et ($k \rightarrow l$) offrant une capacité résiduelle suffisante ou une capacité additionnelle adéquate. Ces arcs sont des ponts entre la partie rouge et la partie bleue. Pour un sommet rouge de Rt_1 ou G_r affecté par la panne, le trafic suivra d’abord les trajets μ et ν à l’intérieur de la partie rouge jusqu’à $p1$; puis, il suivra ensuite l’arbre de routage nominal de $p1$ à i ou k et utilisera l’arc ($i \rightarrow j$) ou ($k \rightarrow l$) comme pont pour atteindre la destination d par des sommets bleus situés dans G_b . En raison de problèmes de dimensionnement, le trafic peut également être rerouté dans G_b sur au moins deux chemins alternatifs jusqu’à la destination d . Selon les choix de reroutage, les chemins de reroutage associés à la défaillance de la liaison ($p1 - q1$) dans G_r et G_b sont sans conflit.

Considérons les pannes de chacun des $n - 1$ autres arcs de l’arbre ; nous choisissons différents arcs ($i \rightarrow j$) pour connecter la partie rouge à la partie bleue. Tout d’abord, nous prouvons l’existence des ponts reliant les deux parties ; ensuite, nous démontrons l’absence de conflit entre les différents chemins de reroutage. Les arcs de l’arbre sont numérotés en ordre décroissant à l’approche de la racine d . Nous choisissons les arcs ($i \rightarrow j$) et ($k \rightarrow l$) par ordre croissant des index des nœuds les constituant. Soit $(p_r \rightarrow q_r)$ un arc en panne. Supposons que nous choisissons des arcs $((i_1 \rightarrow j_1), (k_1 \rightarrow l_1)), ((i_2 \rightarrow j_2), (k_1 \rightarrow l_1)), \dots, ((i_{r-1} \rightarrow j_{r-1}), (k_{r-1} \rightarrow l_{r-1}))$ pour le reroutage. p_r est la racine de l’arbre Rt_r . Nous considérons deux cas. Le premier cas est celui du choix de l’arc $(p_s \rightarrow q_s)$ comme panne, avec s strictement plus petit que r ($s \ll r$). Dans ce cas, les arcs $(i_s \rightarrow j_s)$ et $(k_s \rightarrow l_s)$ ont leurs extrémités i_s et k_s à l’intérieur de l’arbre Rt_r , et les extrémités j_s et l_s hors de Rt_s et donc de Rt_r qui est inclus dans Rt_s (voir figure 26). L’arc $(i_s \rightarrow j_s)$ est donc l’arc (i, j) et $(k_s \rightarrow l_s)$ est l’arc $(k \rightarrow l)$ pour l’arbre Rt_r ; notons ainsi qu’il existe

donc au moins deux ponts à utiliser, ce qui est contradictoire avec le fait que nous supposons avoir utilisé $((i_1 \rightarrow j_1), (k_1 \rightarrow l_1)), ((i_2 \rightarrow j_2), (k_1 \rightarrow l_1)), \dots, ((i_{r-1} \rightarrow j_{r-1}), (k_{r-1} \rightarrow l_{r-1}))$ comme ponts pour le rerouting. Dans le second cas, nous ne fixons pas de contrainte sur les nœuds i_s et k_s . Nous choisissons tous les arcs $(i_r \rightarrow j_r)$ et $(k_r \rightarrow l_r)$ qui relient Rt_r à G_b . Chacun de ces cas offre au moins deux possibilités de chemins dans la partie rouge du réseau.

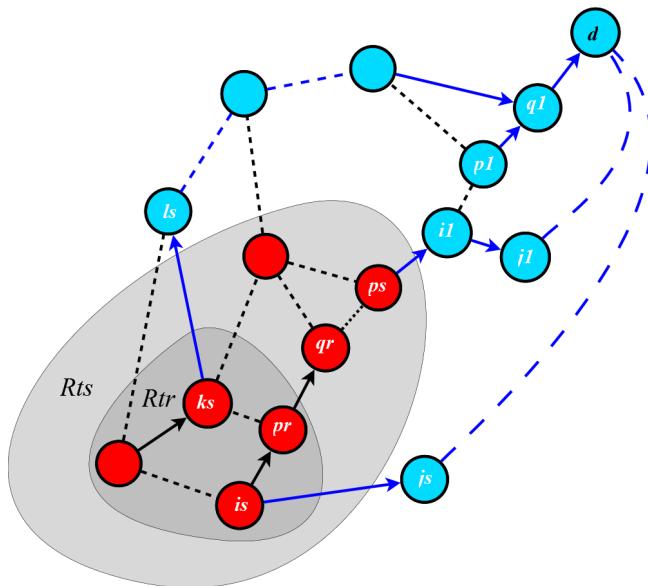


Figure 26 – Preuve par récurrence

Intéressons-nous maintenant à l'absence de conflit dans notre schéma de rerouting. Par récurrence sur le nombre d'arcs reroutés, supposons que nous avons déjà rerouté $n - 1$ arcs dans l'arbre. Chaque arc rerouté a généré au moins deux chemins de rerouting. Selon l'hypothèse de récurrence, il n'y a pas de conflit pour les $n - 1$ premiers reroutages. Nous devons vérifier que le n^{ieme} rerouting n'a pas non plus de conflit avec les $n - 1$ premiers reroutages.

Il n'y a pas de conflit par construction concernant le rerouting de la partie extérieure à l'arbre Rt_r , c'est à dire la partie bleue G_b . Bien que ce n^{ieme} rerouting utilise le même arc que les précédents, dans cette partie, le rerouting suivra le même chemin jusqu'à la destination d ; il n'y a donc pas de conflit. Un conflit peut se produire si la séparation de flux a lieu dans G_b . On vérifie qu'il n'y a pas de conflit pour la partie dans laquelle le flux va dans le sens opposé de l'arbre, ce qui vérifie que dans les deux cas cités ci-dessus, il n'y a pas de conflit. En effet, dans le premier cas, lorsque nous avons choisi les arcs $(i_s \rightarrow j_s)$ et $(k_s \rightarrow l_s)$ dans l'arbre Rt , il n'y avait pas de conflit dans cette partie de l'arbre car Rt_r utiliserait les mêmes arcs $(i_s \rightarrow j_s)$ et $(k_s \rightarrow l_s)$ comme ponts entre sa partie rouge et sa partie bleue (voir figure 26). Dans le second cas, le flux ascendant dans l'arbre Rt_r peut ne rien à avoir de com-

mun avec les autres arcs de reroutage, auquel cas il existerait $(i_s \rightarrow j_s)$ et $(k_s \rightarrow l_s)$. Par conséquent, il n'y a pas de conflit dans ce cas. Nous pouvons ainsi conclure que l'absence de conflit est aussi vérifiée à l'ordre n . Ce qui signifie que l'absence de conflit dans notre stratégie de reroutage a été démontrée par la récurrence. \square

II.3. Modèle mathématique

La stratégie de reroutage que nous proposons dans ce chapitre exploite plusieurs chemins pour rerouter les flux en cas de panne d'une liaison en minimisant la quantité de ressources utilisée. En dehors de la criticité des ressources disponibles, la contrainte de l'absence des conflits dans le reroutage rend encore plus difficile la modélisation et la résolution d'un tel problème. Dans cette section, nous proposons une modélisation formelle du problème d'optimisation lié au reroutage multichemins avec utilisation de capacité résiduelle et additionnelle dans les liens virtuels.

II.3.1. Description du modèle mathématique

Sous réserve des hypothèses fixées dans la section II.2, les notations du tableau I seront utilisées dans la modélisation de notre solution.

Notre objectif étant de minimiser la capacité additionnelle utilisée sur chaque arc lors du reroutage, notre fonction objective est donnée par l'équation II.1 :

$$\min \sum_{(a,b) \in Arc} Ac_{ab} \quad (\text{II.1})$$

sous les contraintes qui suivent.

Contraintes de flux et de reroutage

Ces contraintes aident à construire un ou plusieurs chemins de reroutage à partir d'un nœud fictif F_0 (un des nœuds de la liaison en panne) de G_r vers d . Tout chemin de reroutage part de ce nœud fictif F_0 , passe par le nœud n qui détecte la panne et reroute une portion de trafic. Cette contrainte est exprimée à travers l'équation II.2 et permet de contrôler le nombre de chemins de reroutage dans G_r à travers leurs numéros e .

$$\sum_{\substack{h=h_s \\ (n,h_s) \notin Rt^d, h \text{ voisin de } n, h=h_1}}^{h=h_s} y_{eF_0nh_s}^{dn} = 1, n \in N, d \in N, s = 1, 2, \dots, e = 1, 2, \dots \quad (\text{II.2})$$

Symbole	Description
Rt^d	Ensemble des arcs de l'arbre nominal de routage de destination d
Ac_{ab}	Capacité additionnelle attribuée à l'arc (a, b)
Tr_n^d	Trafic total en destination de d et passant par le nœud n . n est le nœud qui détecte la panne. Une panne est caractérisée par une source n et une destination d , puisque le routage nominal est assuré à travers un arbre de routage. En cas de panne, seul le trafic nominal dans cet arbre doit être rerouté. Pour une destination d donnée, l'arc en panne est celui acheminant le trafic vers d , et en provenance de n par le routage nominal ;
F_i	Indique des nœuds fictifs utilisés pour détourner le trafic en cas de panne. Nous introduisons les nœuds fictifs F_i qui seront utilisés pour toutes les pannes. Pour une panne donnée $(n - d)$, le trafic vers d sera redirigé par les chemins i de F_i vers d et en commençant par l'arc $(F_i \rightarrow n)$, $i = 1, 2, \dots$
$SRed_n^d$	Sous-arbre de racine n . Rappelons qu'en cas de panne, l'arbre est divisé en deux parties, la partie isolée, c'est-à-dire la partie rouge G_r et la partie bleue G_b . Des chemins alternatifs redirigeront le trafic de la partie rouge vers la partie bleue
$SBlue_n^d$	Sous-arbre de racine d , avec $SBlue_n^d = Rt^d - SRed_{nd}$
y_{efgh}^{dn}	Variable binaire indiquant si le $e^{i\text{eme}}$ chemin alternatif vers la destination d pour une panne donnée, contient des arcs $(f \rightarrow g)$ et $(g \rightarrow h)$, le nœud n étant le nœud qui détecte la panne
x_{efgh}^d	Cette variable binaire indique le schéma de rerouting vers la destination d . Il prend la valeur 1 s'il existe une panne dont le $e^{i\text{eme}}$ chemin alternatif vers la destination d contient les arcs $(f \rightarrow g)$ et $(g \rightarrow h)$. Cela signifie que cette variable prend la valeur 1 s'il existe n et e tel que y_{efgh}^{dn} soit égal à 1
α_{eab}^{dn}	Coefficient binaire égal à 1 si l'arc (a, b) appartient à l'un des chemins du routage nominal de n à d , en dehors de l'arc en panne
Quadruplet	Ensemble de quadruplets (e, f, g, h) où e est le numéro d'un chemin de rerouting, f , g et h sont des nœuds du graphe ; f peut être le nœud fictif ; $(f \rightarrow g)$ et $(g \rightarrow h)$ sont deux arcs adjacents, avec $f \neq h$
Arc	Ensemble des arcs du graphe
L	Ensemble des liaisons du graphe
N	Ensemble des nœuds du graphe

Table I – Différentes notations utilisées dans notre modélisation.

Puisqu'un chemin de rerouting ne doit pas passer par un arc en panne, il faut donc que $(n - h_s)$ ne soit pas la liaison en panne.

Pour éviter les boucles et les problèmes de conflit, les chemins alternatifs ne doivent contenir aucun arc de routage nominal dans la partie rouge du réseau. En effet, lorsqu'un flux emprunte un arc de l'arbre nominal dans G_r , il doit suivre les arcs nominaux jusqu'à la destination ; dans la partie rouge G_r , si un flux rerouté suit ces arcs nominaux, alors cela peut conduire à nouveau à la liaison en panne, et entraîner une boucle de reroutage. L'équation II.3 garantit cette condition.

$$y_{efgh}^{dn} = 0, d \in N, n \in N, f \in SRed_n^d, (f, g) \in Rt^d, (e, f, g, h) \in Quadruplet, e = 1, 2, \dots \quad (\text{II.3})$$

Dans la même optique d'éviter les boucles, tout nœud d'un chemin de reroutage doit apparaître au plus une fois dans le chemin. La contrainte de l'équation II.4 permet de s'en rassurer.

$$\sum_{\substack{f \in N | (e, f, g_s, h_s) \in Quadruplet}} y_{efg_s h_s}^{dn} \leq 1, d \in N, n \in N, (g_s, h_s) \in Arc, g_s \in SRed_n^d, h_s \in SRed_n^d, e = 1, 2, \dots \quad (\text{II.4})$$

Par ailleurs, les contraintes qui garantissent la continuité des chemins alternatifs jusqu'à destination sont les suivants :

$$\sum_{f_s \in \text{voisin de } f} y_{ef_s fg}^{dn} = \sum_{h_s \in \text{voisin de } g} y_{efgh_s}^{dn}, d \in N, (f, g) \in Arc, f \neq f_s, f \neq n, g \in N - d, s = 1, 2, \dots, e = 1, 2, \dots \quad (\text{II.5})$$

$$\sum_{e=1}^{e=k} \sum_{h_s \in \text{voisin de } g} y_{engh_s}^{dn} = y_{F_0 ng}^{dn}, d \in N, n \in N, (n, g) \in Arc, k = 1, 2, \dots, e = 1, 2, \dots \quad (\text{II.6})$$

$$y_{eghh_1}^{dn} - y_{efgh}^{dn} \geq 0, h \in SBlue_n^d - d, (h_1, h) \in Rt^d, (g, h) \in Rt^d, \forall d \in N, \forall (e, f, g, h) \in Quadruplet, e = 1, 2, \dots \quad (\text{II.7})$$

L'équation II.5 est la contrainte de la conservation du débit de données. La quantité de données entrant sur le nœud i de la liaison $(f - g)$ du e^{ieme} chemin de reroutage est la même qui sort du nœud g . Puisque cette contrainte de conservation

de flux ne s'applique pas au nœud n qui détecte la panne de liaison, nous avons donc la contrainte de flux II.6 pour ce cas particulier. L'équation II.7 garantit que dans la partie bleue, si un chemin utilise un arc de l'arbre de routage nominal, il doit continuer jusqu'à la destination d .

Contraintes relatives aux conflits

Pour s'assurer de l'absence de conflits entre les chemins de reroutage, nous fixons les contraintes II.8, II.9, II.10 et II.11.

$$\sum_{n \in N} \sum_{e=1}^{e=k} y_{engh}^{dn} \geq x_{efgh}^d \geq \frac{\sum_{n \in N} y_{engh}^{dn}}{\text{cardinal}(N)}, (e, f, g, h) \in \text{Quadruplet}, n \in N, \\ k = 1, 2, \dots, e = 1, 2, \dots \quad (\text{II.8})$$

$$\sum_{\substack{h=h_s \\ h_s \in \text{voisin de } g, h=h_1}} x_{efgh}^d \leq 1, (f, g) \in \text{Arc}, d \in N, s = 1, 2, \dots, e = 1, 2, \dots \quad (\text{II.9})$$

$$x_{efgh}^d \in \{0, 1\}, \forall (e, f, g, h) \in \text{Quadruplet}, \forall d \in N, e = 1, 2, \dots \quad (\text{II.10})$$

$$y_{efgh}^{dn} \in \{0, 1\}, \forall (e, f, g, h) \in \text{Quadruplet}, \forall d \in N, \forall n \in N, e = 1, 2, \dots \quad (\text{II.11})$$

Les contraintes II.10 et II.11 dépendent par la définition des variables x et y , permettent de se rassurer de l'existence ou pas d'arcs en commun entre les chemins de reroutage de pannes différentes pour une même destination. En outre, II.9 exprime l'absence de conflit, puisque si nous utilisons un arc de reroutage ($f \rightarrow g$), alors nous devons utiliser au plus un arc ($g \rightarrow h$) dans la partie bleue pour les e chemins ayant l'arc ($f \rightarrow g$) en commun. La contrainte II.8 exprime le fait que pour une panne donnée, il existe au plus un chemin de reroutage dans la partie bleue contenant le même arc ($f \rightarrow g$) que le chemin numéro e dans la partie rouge.

Contraintes de capacité

Il s'agit des contraintes liées à la capacité additionnelle ajoutée aux liaisons pour supporter le trafic rerouté. Ces contraintes sont celles des équations II.12, II.13, II.14.

$$\begin{aligned}
& \sum_{d \in N | (n,m) \in Rt^d} \sum_{f \in \text{voisin de } g, f \neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn} \cdot Tr_n^d + \sum_{d \in N | (m,n) \in Rt^d} \sum_{f \in \text{voisin de } g, f \neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn} \cdot Tr_m^d \\
& \leq Ac_{gh} + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{egh}^{dn} \cdot Tr_n^d + \sum_{d \in N | (m,n) \in Rt^d} \alpha_{egh}^{dm} \cdot Tr_m^d, \\
& (g,h) \in Arc, g \neq n, g \neq m, (n,m) \in L, k = 2, 3, \dots, e = 1, 2, \dots
\end{aligned} \tag{II.12}$$

$$\begin{aligned}
& \sum_{d \in N | (n,m) \in Rt^d} \sum_{e=1}^{k=k} y_{eF_0nh}^{dn} \cdot Tr_n^d \leq Ac_{nh} + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{nh}^{dn} \cdot Tr_n^d, \\
& (n,h) \in Arc, h \neq m, (n,m) \in L, k = 2, 3, \dots, e = 1, 2, \dots
\end{aligned} \tag{II.13}$$

$$\begin{aligned}
& \sum_{d \in N | (n,m) \in Rt^d} \sum_{e=1}^{e=k} y_{eF_0mh}^{dn} \cdot Tr_m^d \leq Ac_{nh} + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{mh}^{dm} \cdot Tr_m^d, \\
& (m,h) \in Arc, h \neq n, (n,m) \in L, e = 1, 2, \dots
\end{aligned} \tag{II.14}$$

La contrainte II.12 exprime le fait que le total de trafic rerouté sur les e chemins de rerouting est au plus égale au total de la capacité additionnelle sur chaque liaison de ces chemins combinée à leur capacité initiale. Elle définit ainsi les limites de la capacité additionnelle utilisée sur l'arc $(g \rightarrow h)$ pour rerouter le flux issus de la panne de la liaison $(n - m)$. Le terme Ac_{gh} représente la capacité additionnelle ajoutée sur l'arc $(g \rightarrow h)$; les termes $\sum_{d \in N | (n,m) \in Rt^d} \alpha_{egh}^{dn} \cdot Tr_n^d$ et $\sum_{d \in N | (m,n) \in Rt^d} \alpha_{egh}^{dm} \cdot Tr_m^d$ quant à eux représentent le flux à rerouter provenant des arcs $(n \rightarrow m)$ et $(m \rightarrow n)$ de la liaison en panne $(n \leftrightarrow m)$. II.13 et II.14 sont des cas particuliers de II.12 concernant le noeud qui détecte la panne de l'arc $(n \rightarrow m)$ ou $(m \rightarrow n)$.

II.3.2. Convexité du modèle

La formulation ILP (Integer Linear Program) de notre stratégie de rerouting est soumise à plusieurs contraintes portant sur des métriques différentes. Il faut donc se rassurer que ces contraintes soient bien exprimées et permettent de trouver formellement des chemins de rerouting pour une panne de liaison. Pour ce faire, le modèle proposé doit être convexe. Dans cette section, nous étudions cette convexité.

La fonction objective de notre modèle a la forme générale de l'équation II.15 :

$$\min_{s.t. x \in C} f(x) \tag{II.15}$$

où C est l'ensemble des arcs noté Arc et f est une fonction sur C donnant la capacité additionnelle nécessaire sur un arc donné pour rerouter un flux. De part sa définition, le problème décrit par l'équation II.15 est convexe dans l'ensemble C et la fonction f est convexe (Boyd & Vandenberghe, 2004).

En effet, sous l'hypothèse de l'existence d'au moins deux chemins disjoints entre toute paire de noeuds du réseau et considérant tout arc de l'ensemble Arc comme un segment, l'ensemble C est convexe comme intersection de sous-ensembles (les segments) convexes. La fonction f étant affine, elle est donc convexe et le problème décrit par l'équation II.15 est implicitement convexe (Boyd & Vandenberghe, 2004). Cette convexité implicite est due au fait qu'il existe des formulations plus explicites de problèmes convexes tels que les problèmes d'optimisation convexes sous forme fonctionnelle, qui sont des problèmes convexes de la forme :

$$\min_{S.t. g_i(x) \leq 0, i=1,2,\dots,m, h_j(x)=0, j=1,2,\dots,p} f(x) \quad (\text{II.16})$$

où $f, g_1, g_2, \dots, g_m : \mathbb{R}^n \mapsto \mathbb{R}$ sont des fonctions convexes et $h_1, h_2, \dots, h_p : \mathbb{R}^n \mapsto \mathbb{R}$ sont des fonctions affines. Chacune des contraintes de notre modèle pouvant s'écrire sous n'importe laquelle des formes de contraintes de l'équation II.16, ceci montre donc que notre modèle est implicitement convexe.

II.3.3. Heuristiques

Le modèle mathématique présenté précédemment impose formellement le respect d'un certain nombre de contraintes de diverses natures qui rendent très difficile la détermination de solutions malgré leur existence prouvée, même pour un réseau de petite taille. Cela justifie le besoin de solutions heuristiques pour la détermination progressive de chemins de reroutages qui minimisent la capacité additionnelle utilisée dans le réseau. Nous rappelons que le graphe du réseau est orienté avec des liaisons bidirectionnelles, une seule panne de liaison se produit à la fois et il existe un arbre de routage nominal prédéfini pour chaque destination. Sur la base de ces hypothèses et du théorème 1, nous proposons deux heuristiques : une qui ne tient pas compte des conflits et une autre qui intègre la résolution de conflits de routage. Les notations du tableau II seront utilisées pour décrire ces heuristiques.

II.3.3.1. Heuristique 1 : sans gestion des conflits

Le but de l'algorithme est de fournir pour chaque situation de panne et chaque destination dans le réseau, au moins un chemin de reroutage utilisable. Un seul chemin de reroutage est utilisé lorsque la capacité résiduelle disponible au niveau des liaisons est suffisante pour rerouter la totalité du flux ; dans tous les autres cas,

Symbol	Description
$G(N, L)$	Graphe possédant un ensemble N de nœuds et un ensemble L de liaisons
p	Panne d'un arc de liaison
P	Ensemble des pannes d'arcs de liaisons
d	Destination du flux à rerouter dû à la panne p
$M[p][a]$	Capacité résiduelle nécessaire sur l'arc de reroutage a en cas de panne p
$M[p'][a]$	Capacité résiduelle nécessaire sur l'arc de reroutage a en cas de panne p' (le symétrique de p)
$R[a]$	Capacité disponible (résiduelle et additionnelle) sur l'arc a
$capRes[w]$	Capacité résiduelle sur un arc w
$T[d][p]$	Poids du flux à rerouter
μ	Ensemble des arcs d'un chemin de reroutage valide
k	Nombre de sous-flux sortant d'un nœud donné
E	Liste des chemins de reroutage
$arcsList$	Liste des arcs sortants d'un nœud donné. $Card(arcsList)$ désigne le nombre d'éléments de $arcsList$

Table II – Différentes notations utilisées dans les heuristiques.

le flux est rerouté à travers au moins deux chemins. Pour chaque situation de panne, on récupère tout d'abord les capacités résiduelles et additionnelles nécessaires au niveau de chaque liaison potentiellement sujette à une panne p , que l'on stocke dans la variable $M[p][a]$.

Pour une destination d donnée ainsi qu'une panne p , l'algorithme construit arc après arc, les chemins de reroutages en cherchant à minimiser le total de capacité additionnelle nécessaire pour réaliser le reroutage du flux engendré par la panne p . Au niveau de chaque nœud intervenant dans le reroutage et ayant au moins deux liaisons sortantes, on divise si nécessaire le flux entrant en k parties selon le besoin des sorties. Sur la liaison sortante offrant la plus grande capacité résiduelle, nous envoyons une partie du flux égale à cette capacité résiduelle ; l'autre partie du flux est envoyée sur l'autre liaison offrant la plus grande capacité résiduelle. Afin d'optimiser l'usage de la capacité additionnelle, les flux reroutés suivent les chemins qui minimisent le coût $S = \sum_{a \in \mu} \delta(R[a], M[p][a] + T[d][p] + M[p'][a])$ où δ est une fonction telle que $\delta(x, y) = y - x$ si $y > x$ et 0 sinon. Lorsque $\delta(x, y) = y - x$, cela signifie que la capacité résiduelle disponible ($R[a]$) n'est pas suffisante pour supporter le flux à rerouter ; dans ce cas, la séparation de flux est envisagée si cet arc a doit être utilisé dans le reroutage.

L'algorithme 1 présente notre protocole de construction des chemins de reroutage avec un nombre de subdivisions $k \geq 2$ à chaque nœud, et ne prend pas en

compte la gestion des conflits. La gestion des conflits dans les n chemins générés sera intégrée dans la suite.

Algorithme 1 : Séparation de flux avec utilisation de capacité additionnelle sans gestion de conflit.

Entrées : Un graphe $G(N, L)$; un arbre de routage A; un ensemble de destinations D et de pannes P ;

Sorties : Un système de reroutage

```

1 Pour chaque destination d dans la liste des destinations D faire
2   Pour chaque panne p dans la liste des pannes P faire
3     Pour chaque arc a dans G – Gr ∪ Gb faire
4       Si  $(M[p][a] + M[p'][a] + T[d][p]) - R(a) > 0$  alors
5         Initialiser le poids des arcs à la valeur
6          $M[p][a] + M[p'][a] + T[d][p]$  ;
7       Si  $(M[p][a] + M[p'][a] + T[d][p]) - R(a) \leq 0$  alors
8         Initialiser le poids des arcs à la valeur 0 ;
9       /* On trouve d'abord le meilleur chemin de référence */ */
10       $\mu \leftarrow$  plus court chemin vers  $d$  avec dijkstra;
11      /* On construit progressivement à partir de la référence, les autres
12        chemins */ */
13      Pour chaque nœud v dans  $\mu$  faire
14         $arcsList \leftarrow$  liste des arcs sortants de  $v$ ;
15        Pour t allant de 1 à Card(arcsList) faire
16           $l \leftarrow$  arc de plus grande capacité résiduelle;
17          Si  $M[p][l] \geq$  trafic entrant  $T$  alors
18            envoyer tout le trafic  $T$  sur  $l$ 
19          Sinon
20            /* Subdivision du flux */ */
21            Si  $t < Card(arcsList)$  alors
22              envoyer un trafic  $T$  égal à  $capRes[l]$ ;
23               $T \leftarrow T - capRes[l]$ ;
24            Si  $t = Card(arcsList)$  et  $T \neq 0$  alors
25              /* La capacité additionnelle sera utilisée sur l */ */
26              envoyer le reste de trafic  $T$  sur  $l$ 
27             $path \leftarrow$  plus court chemin vers  $d$  avec dijkstra passant par  $l$ ;
28            Ajouter  $path$  à la liste des chemins  $E$ 
```

La subdivision est faite d'abord selon la valeur de la plus grande capacité résiduelle des liaisons sortantes du nœud portant le flux, ensuite le reste de flux est subdivisé selon la valeur de la deuxième plus grande capacité résiduelle, puis sui-

vant la valeur de la troisième plus grande capacité résiduelle, ainsi de suite jusqu'à n subdivisions.

II.3.3.2. Heuristique 2 : avec gestion de conflit

Puisqu' avec la subdivision de flux on peut avoir plusieurs chemins de reroutage pour une panne donnée, il faut vérifier l'absence de conflit à la fois entre les chemins de reroutage obtenus pour la panne courante et ceux obtenus pour les pannes traitées précédemment. Dans le cas d'une dichotomie de flux, la vérification de l'absence de conflits a lieu entre les couples de chemins obtenus pour le reroutage. Ainsi, si deux chemins ont un arc en commun, alors ils doivent être identiques après cet arc jusqu'à la destination. Lors de la vérification, tout arc donnant lieu à un conflit avec un autre arc d'un chemin précédemment sélectionné, doit être abandonné à la faveur de l'arc précédemment choisi. Pour illustrer cette gestion de conflits, considérons le graphe orienté de la figure 27.

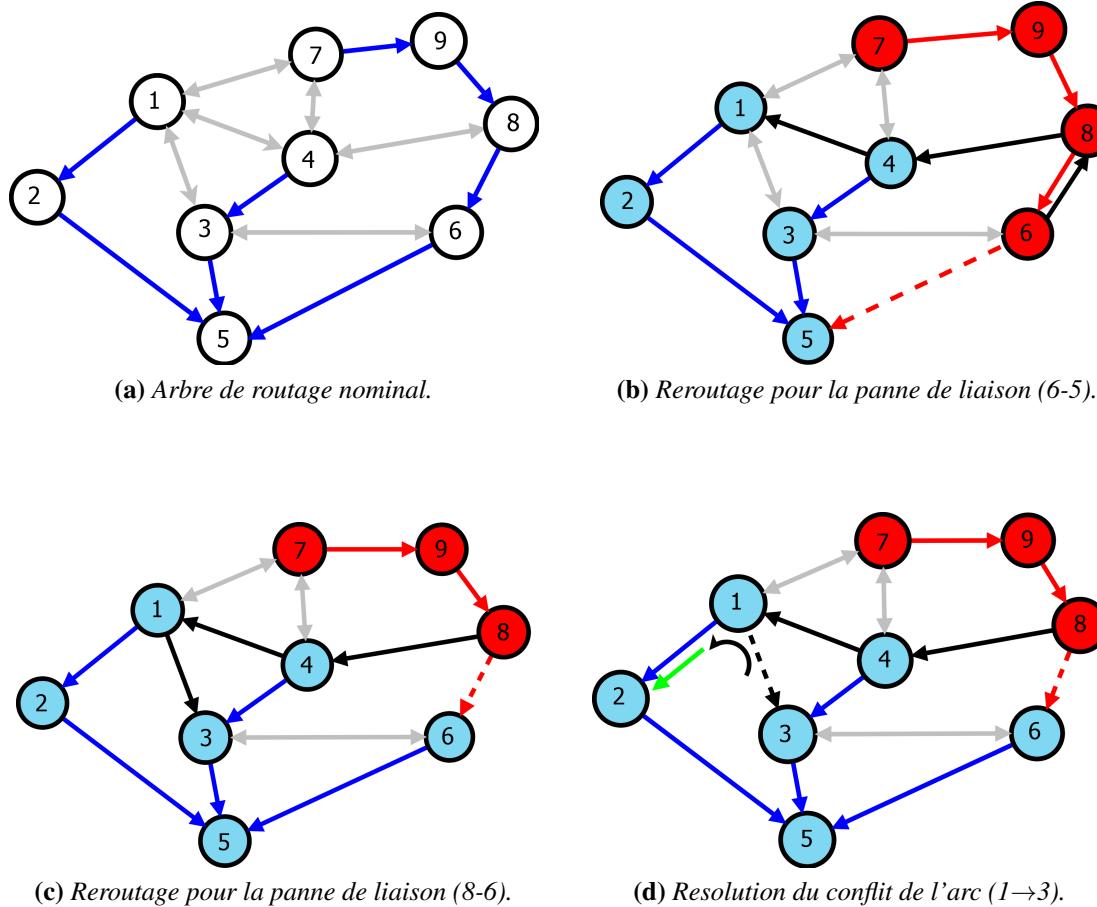


Figure 27 – Résolution de conflit.

En cas de panne de la liaison (6-5), les deux chemins $6 \rightarrow 8 \rightarrow 4 \rightarrow 3 \rightarrow 5$ et $6 \rightarrow 8 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 5$ sont choisis pour rerouter la totalité du flux (voir figure 27b). Quand

la liaison (8-6) tombe en panne, les chemins $8 \rightarrow 4 \rightarrow 3 \rightarrow 5$ et $8 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 5$ sont choisis pour le reroutage (voir figure 27c). Si la panne de la liaison (6-5) est résolue avant celle de la liaison (8-6), on aura un conflit entre les chemins $8 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 5$ et $6 \rightarrow 8 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 5$ après l'arc (4→1). En effet, après cet arc qu'ils ont en commun, ces deux chemins ne sont pas identiques jusqu'à destination ; la fusion consistera donc à éliminer l'arc qui crée le conflit sur le chemin $8 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 5$ (c'est à dire l'arc (1→3)) et le remplacer par l'arc (1→2) (voir figure 27d). Les chemins de reroutage que nous obtenons après élimination du conflit pour la panne de liaison (8-6) sont $6 \rightarrow 8 \rightarrow 4 \rightarrow 3 \rightarrow 5$ et $6 \rightarrow 8 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 5$.

L'algorithme 2 présente notre méthode avec gestion des conflits.

II.4. Analyse et interprétation des résultats de simulations

Afin d'évaluer les performances de notre modèle, des simulations ont été effectuées dans l'environnement de simulation de réseau OMNeT++ version 5.0, fonctionnant dans un ordinateur doté des caractéristiques suivantes : Core i5 2.40GHz, 4.00 Go de RAM et 12Mo de cache. Dans cette section, nous présentons les résultats obtenus.

Pour tenir compte de la scalabilité, les simulations ont été réalisées sur 4 réseaux de différentes tailles conformes à nos hypothèses : Réseau1 (5 nœuds, 7 liaisons : Figure 45), Réseau2 (10 nœuds, 18 liaisons : Figure 50), Réseau3 (20 nœuds, 31 liaisons) et Réseau4 (60 nœuds, 81 liaisons).

L'objectif des simulations est d'apprécier dans un premier temps le taux de restauration des pannes de notre stratégie de reroutage par rapport à celle de Son (2014). Il s'agit de déterminer quelle stratégie permet la restauration d'un nombre maximal de liaisons en panne. Ensuite, il est question d'apprécier le temps de restauration des pannes de liaison selon les différents réseaux, afin de mesurer la résistance de notre stratégie de reroutage face à la scalabilité. L'appréciation de ces différentes métriques est faite sur la base de comparaisons entre l'approche de Son (2014) et les deux versions de notre stratégie de reroutage par séparation de flux : la dichotomie de flux et la multiséparation (séparation d'un flux en plus de deux parties).

II.4.1. Taux de restauration des pannes

Les résultats numériques de notre protocole avec gestion des conflits sont résumés dans le tableau III. La colonne "Restauration par DF" représente le nombre de pannes restaurées par la dichotomie de flux ; la colonne "Restauration par MSF"

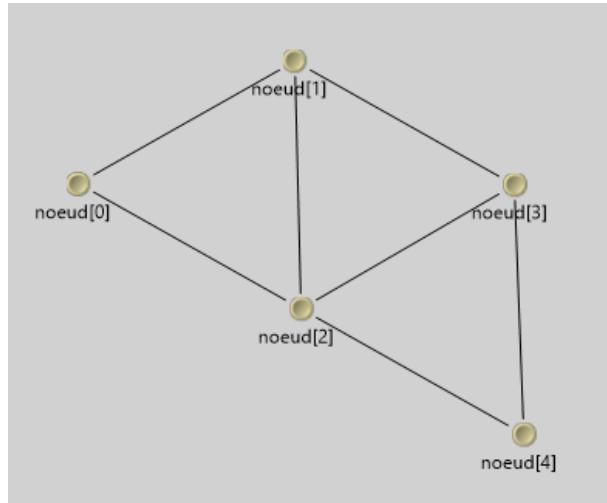
Algorithme 2 : Séparation de flux avec utilisation de capacité additionnelle avec gestion de conflit.

Entrées : Un graphe $G(N, L)$; un arbre de routage A; un ensemble de destinations D et de pannes P ;

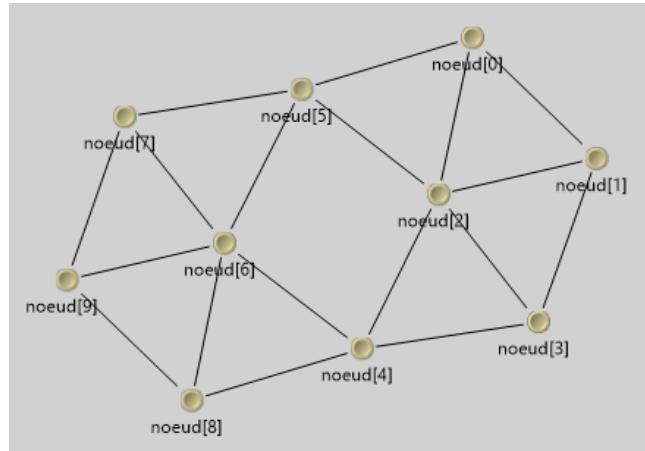
Sorties : Un système de reroutage sans conflit

```

1 Pour chaque destination  $d$  dans la liste des destinations  $D$  faire
2   Pour chaque panne  $p$  dans la liste des pannes  $P$  faire
3     Pour chaque arc  $a$  dans  $G - G_r \cup G_b$  faire
4       Si  $(M[p][a] + M[p'][a] + T[d][p]) - R(a) > 0$  alors
5         Initialiser le poids des arcs à la valeur
6          $M[p][a] + M[p'][a] + T[d][p]$  ;
7       Si  $(M[p][a] + M[p'][a] + T[d][p]) - R(a) \leq 0$  alors
8         Initialiser le poids des arcs à la valeur 0 ;
9
10    /* On trouve d'abord le meilleur chemin de référence */ /
11     $\mu \leftarrow$  plus court chemin vers  $d$  avec dijkstra;
12    /* On construit progressivement à partir de la référence, les autres
13    chemins */ /
14
15    Pour chaque nœud  $v$  dans  $\mu$  faire
16       $arcsList \leftarrow$  liste des arcs sortants de  $v$ ;
17      Pour  $t$  allant de  $l$  à  $Card(arcsList)$  faire
18         $l \leftarrow$  arc de plus grande capacité résiduelle;
19        Si  $M[p][l] \geq$  trafic entrant  $T$  alors
20          envoyer tout le trafic  $T$  sur  $l$ 
21
22        Sinon
23          /* Subdivision du flux */ /
24          Si  $t < Card(arcsList)$  alors
25            envoyer un trafic  $T$  égal à  $capRes[l]$ ;
26             $T \leftarrow T - capRes[l]$ ;
27
28          Si  $t = Card(arcsList)$  et  $T \neq 0$  alors
29            /* La capacité additionnelle sera utilisée sur  $l$  */ /
30            envoyer le reste de trafic  $T$  sur  $l$ 
31
32
33     $path \leftarrow$  plus court chemin vers  $d$  avec dijkstra passant par  $l$ ;
34    Ajouter  $path$  à la liste des chemins  $E$ ;
35    Si (conflit entre path et les autres chemins obtenus) alors
36      Fusionner les chemins obtenus ;
37
38    Si (conflit entre path et les chemins précédemment obtenus) alors
39      Fusionner les chemins obtenus ;
40
41
42    Mettre à jour  $R[a]$  et  $M[p][a]$ ;
```



(a) Réseau 1.



(b) Réseau 2.

Figure 28 – Quelques réseaux de tests.

représente le nombre de pannes restaurées par la séparation de flux en plus de deux parties. Nous pouvons constater une restauration presque complète des différentes configurations de pannes de liaisons simples par les deux versions de séparation de flux utilisées. Les cas de panne non traités sont dus à des problèmes de cycles qui n'ont pas pu être résolu.

En éliminant la gestion des conflits dans le mécanisme de restauration, nous aboutissons aux résultats du tableau IV.

Nous pouvons constater que la négligence des conflits permet de restaurer quelques pannes en plus, mais le nombre de pannes restaurées en plus est de l'ordre de 2% à 3% ; cela signifie que la contrainte de gestion des conflits n'a pas une très grande influence sur l'efficacité de notre méthode, tout comme dans le cas de la stratégie de restauration de Son (2014). Néanmoins, la séparation de flux permet de restaurer

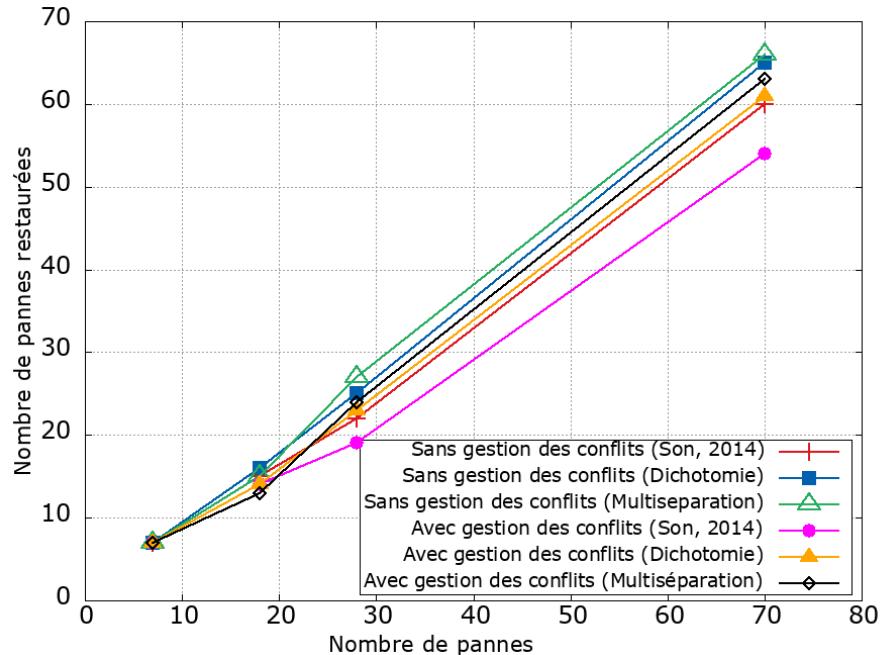
Réseaux	Nbre de noeuds	Nbre de liaisons	Cas de pannes	Restauration par Son (2014)	Restauration par DF	Restauration par MSF
Reseau1	5	7	7	7	7	7
Reseau2	10	18	18	14	14	13
Reseau3	20	31	28	19	23	26
Reseau4	60	81	70	54	61	63

Table III – Taux de restauration des pannes avec gestion de conflits.

Réseaux	Nbre de noeuds	Nbre de liaisons	Cas de pannes	Restauration par Son (2014)	Restauration par DF	Restauration par MSF
Reseau1	5	7	7	7	7	7
Reseau2	10	18	18	15	16	15
Reseau3	20	31	28	22	25	27
Reseau4	60	81	70	60	65	66

Table IV – Taux de restauration des pannes sans gestion de conflits.

plus de pannes que la méthode de Son (2014). A partir des données des tableaux III et IV, nous obtenons les courbes de la figure 29.

**Figure 29 – Courbe comparative des restaurations avec conflits et sans conflits.**

D'après ces courbes, le nombre de pannes restaurées croît avec le nombre de pannes auxquelles nous faisons face, que nous soyons dans un cadre de restauration avec gestion des conflits entre chemins de reroutage ou pas. L'écart entre les deux courbes se creuse au fur et à mesure que le nombre de pannes augmente, ce qui

signifie que plus l'infrastructure réseau est de grande taille, plus il devient difficile d'assurer la gestion des conflits entre les chemins de reroutage. Par ailleurs, on peut observer un meilleur taux de restauration de panne avec la technique de séparation de flux en plus de deux parties lorsque le réseau est assez grand, contrairement à celle séparant le flux en deux parties (Dichotomie de flux) ; ceci laisse suggérer que la stratégie de séparation de flux en plus de deux parties, est plus appropriée pour les réseaux de grande taille.

II.4.2. Coûts de restauration des pannes

L'évaluation des coûts moyens de restauration pour les quatres (04) réseaux précédents nous a fourni les données du tableau V. Dans ce tableau, la colonne "Dichotomie AC" donne les résultats de notre stratégie de restauration par dichotomie de flux avec gestion des conflits ; "Multiséparation AC" donne les résultats de notre stratégie de restauration par séparation de flux en plus de deux parties avec gestion des conflits. La colonne "Méthode de Son" quant à elle donne les résultats de la méthode de [Son \(2014\)](#) pour les différents réseaux considérés. Ces résultats sont exprimés en secondes.

Réseaux	Dichotomie AC	Multiséparation AC	Méthode de Son (2014)
Reseau1	0.703	0.79047	0.597
Reseau2	0.951	1.06810	0.897
Reseau3	2.512	2.10632	1.825
Reseau4	4.382	3.99104	3.085

Table V – Coût de restauration pour pannes simples.

Les données du tableau V permettent d'obtenir les courbes de la figure 30.

Nous pouvons remarquer une différence significative des coûts de restauration des deux méthodes de séparation de flux par rapport à celle de Son ; celà est tout d'abord dû à la contrainte de gestion des conflits qui garantit l'élémentarité des chemins de reroutage obtenus. Dans le cadre de la séparation de flux, cette contrainte implique la vérification et la fusion éventuelle de plus de chemins de reroutage par rapport à la stratégie de [Son \(2014\)](#). Par ailleurs, la taille du réseau participe aussi à augmenter les coûts de gestion des conflits. En effet, plus la taille du réseau est grande, plus la séparation de flux peut engendrer de chemins de reroutage et par conséquent, augmenter le nombre de chemins à vérifier pour gérer les conflits.

Les courbes de la figure 30 permettent également de constater pour les réseaux de petite taille (moins de 10 nœuds), que notre stratégie de reroutage par séparation de flux est proche de celle de [Son \(2014\)](#) en terme de temps de restauration ; cela

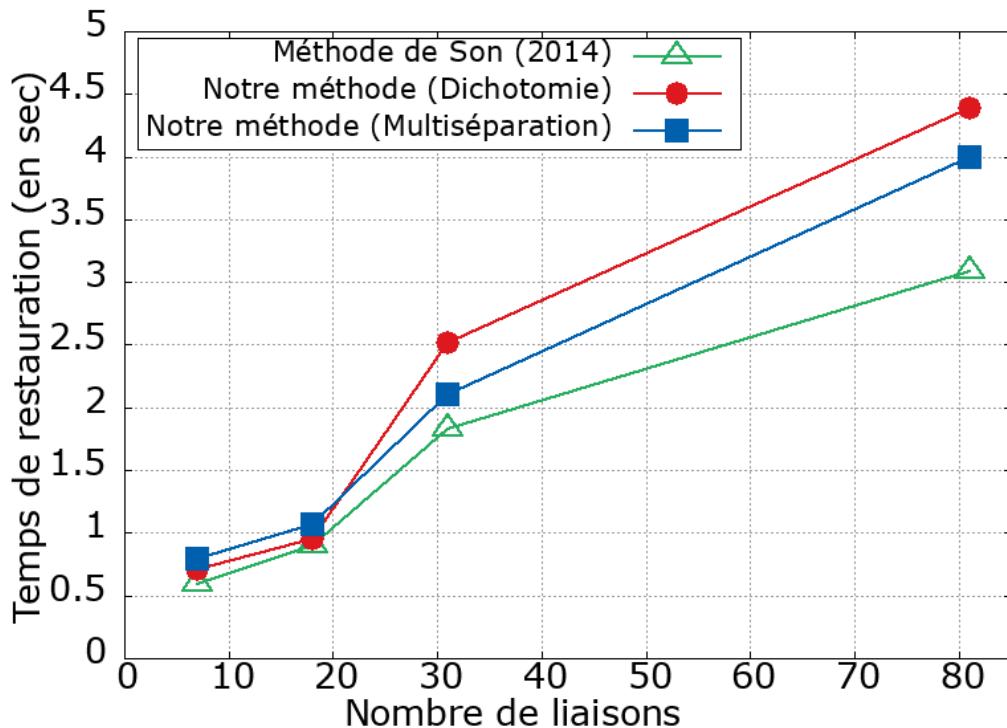


Figure 30 – Coûts de restauration de différentes méthodes.

s'explique par le fait que le nombre de liaisons participant à la gestion des conflits ne soit pas grand, du fait de la petite taille du réseau.

La taille du réseau influence également la capacité additionnelle nécessaire pour assurer le rerouting des flux en cas de panne, comme l'illustre le tableau VI. La colonne "CA disponible" représente la capacité additionnelle disponible dans le réseau ; la colonne "CA par dichotomie" représente le total de capacité additionnelle utilisée par notre dichotomie de flux dans le réseau pour la restauration des pannes de liaisons simples ; "CA par multiséparation" représente le total de capacité additionnelle utilisée par la séparation de flux en plus de deux parties. "CA par Son" est la capacité additionnelle utilisée dans le réseau par la méthode de Son (2014). Ce tableau dresse une étude comparative de notre stratégie de restauration par rapport à celle de Son (2014) en termes de capacité additionnelle utilisée.

Réseaux	CA disponible	CA par dichotomie	CA par multi-separation	CA par Son (2014)
Reseau1	3342	1012	1012	1624
Reseau2	7216	2433	2509	2741
Reseau3	14000	3802	3125	4524
Reseau4	12052	4110	3906	5021

Table VI – Comparaison de capacité additionnelle utilisée pour différentes techniques.

Nous pouvons constater que notre stratégie de reroutage minimise davantage l'usage de la capacité additionnelle que celle de Son (2014). Cette différence significative est due à notre séparation de flux qui diminue le poids du flux acheminé sur certaines liaisons. Les courbes de la figure 31 montrent que cette différence ressort davantage lorsque le réseau grandit. Nous y observons par exemple un écart de presque 1000 unités sur les capacités additionnelles utilisées entre la stratégie de restauration de Son (2014) et la nôtre pour le réseau de 60 nœuds ; ceci laisse penser que pour cette topologie de 60 nœuds, notre stratégie de restauration par séparation de flux permet d'économiser environ 18% de capacité additionnelle comparée à celle de Son (2014). Cela peut s'expliquer par le fait que plus le réseau est de grande taille, plus les flux peuvent être sujet à des subdivisions pouvant conduire à des économies. De plus, d'après ces mêmes courbes, nous pouvons estimer à environ 27% la consommation en capacité additionnelle de notre stratégie de reroutage contre 32% pour celle de Son (2014).

Par ailleurs, nous observons que la multiséparation consomme moins de ressources additionnelles que la dichotomie de flux. Cette observation se vérifie davantage lorsque le réseau est assez grand (plus de 50 nœuds). Par contre, dans les réseaux de petite taille (20 nœuds et 31 liaisons), on peut observer à certains moments que la capacité additionnelle utilisée par la multiséparation est supérieure à celle de la dichotomie. Cette situation revèle que la multiséparation est mieux adaptée au réseaux de grande taille.

II.5. Conclusion

Dans ce chapitre, nous avons présenté notre solution de reroutage par séparation de flux, pour la résolution de la panne d'une liaison dans les réseaux virtuels. Cette stratégie est implémentée avant la mise en marche du réseau dans un contrôleur SDN qui l'installe au sein des différents nœuds du réseau. Lorsqu'une panne de liaison se produit, le nœud en amont de la panne initie un reroutage local sur la base des chemins préalablement définis par le contrôleur ; cela permet un reroutage rapide de la panne. Un modèle ILP (Integer Linear Program) a été proposé pour valider formellement notre solution. La complexité de ce modèle a conduit à la mise en place de deux heuristiques dont l'usage dans les simulations réalisées, ont montré un impact faible de notre méthode sur les ressources du réseau physique ainsi qu'un taux de restauration de pannes plus important que ceux de la littérature. De plus, nous avons constaté qu'un nombre assez important de subdivisions du flux original induit dans certains cas, l'utilisation de davantage de ressources

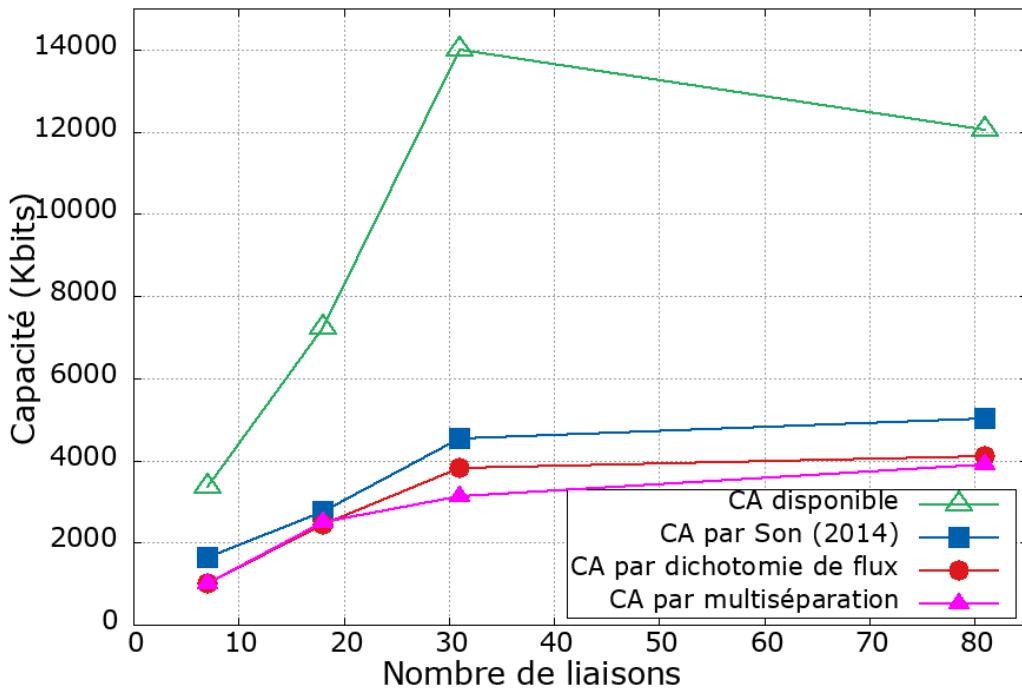


Figure 31 – Capacité additionnelle utilisée par les différentes méthodes.

additionnelles, même si le taux de restauration de pannes est meilleur. Cependant, lorsque la panne de liaison persiste dans le temps, les chemins précalculés par notre méthode deviennent rapidement obsolètes et ne permettent plus de router les flux suivant des chemins optimaux. Le chapitre III est consacré à la mise en place d'une méthode de recalcul des chemins de routage optimaux et la mise à jour des tables de routage des nœuds lorsqu'une ou plusieurs pannes sont déclarées persistantes, l'une des difficultés majeures étant d'effectuer ces mises à jour sans perturber le trafic ambiant.

III

CHAPITRE

GESTION DE LA QOS PAR MISE À JOUR DES TABLES DE ROUTAGE DANS LE CAS DES PANNEES PERSISTANTES

SOMMAIRE

III.1 - Introduction	74
III.2 - Réflexions sur quelques mécanismes de détection des nœuds à mettre à jour	75
III.3 - Notre stratégie de mise à jour de tables de routage en environnement centralisé	83
III.4 - Extension à la gestion des pannes multiples persistantes	91
III.5 - Résultats de simulations	99
III.6 - Conclusion	109

III.1. Introduction

Dans ce chapitre, nous présentons notre contribution à la résolution du problème de mise à jour des tables de routage pour la gestion des cas de pannes persistantes d'une ou plusieurs liaisons dans les réseaux virtuels. Cette contribution sera présentée en deux versions : une version centralisée où un équipement en particulier déclenche le processus de mise à jour, et une version distribuée pour laquelle les mises à jour s'effectuent simultanément à plusieurs endroits différents. Pour simplifier la présentation de notre stratégie, nous supposons initialement que le reroutage est unicemin. Ceci peut être aisément étendu au reroutage multichemins comme celui que nous avons présenté au chapitre II. Dans ce cas, de mise à jour

que nous mettons en place dans le cas d'un reroutage unichemin pourra être utilisée pour chacun des chemins obtenus par séparation de flux. Les hypothèses sont les mêmes qu'au chapitre II, section II.2 dont nous rappelons ici quelques-unes :

- le réseau est représenté par un graphe dont les liaisons sont bidirectionnelles.
- Chaque arc est doté d'un poids qui représente la bande passante ou le débit sur cette liaison ;
- entre toute paire de nœuds, il existe au moins deux chemins disjoints permettant de les relier ;
- pour toute situation de panne, un chemin de reroutage est déjà fourni pour acheminer les flux suivant la stratégie utilisée dans (Pham et al., 2012 ; Son, 2014) ;
- un seul plan virtuel est considéré dans un premier temps.

La mise à jour des nœuds devant se faire sans nuire au trafic courant, un choix judicieux des nœuds à impliquer dans le processus s'impose. Nous définissons deux principaux types de nœuds :

- *nœud frontière* : nœud de l'arbre nominal dans la partie rouge, relié directement à un nœud de la partie bleue dans la méthode de Son (2014) ;
- *nœud critique* : nœud frontière le plus proche du nœud qui détecte une panne de liaison passagère ;
- N_f : nœud qui détecte la panne de liaison lorsqu'elle se produit.

Parmi les contraintes à satisfaire par notre procédé de mise à jour, l'un des plus importants est le maintien d'une structure d'arbre après la mise à jour, afin de conserver toutes les propriétés du réseau et pouvoir y appliquer toutes les techniques basées sur l'arbre de routage nominal. L'incident direct d'une telle contrainte serait la proposition d'une qualité de service proche de celle du réseau en l'absence de panne. Ainsi, nous présentons d'abord différents mécanismes de détection de nœuds à mettre à jour, puis nous déroulons notre processus de mise à jour sur ces nœuds dans le cas d'un reroutage unichemin pour une panne de liaison simple et persistante. Nous étendons ensuite notre méthode à la gestion des pannes de liaisons multiples et persistantes, ainsi qu'aux pannes de nœuds.

III.2. Réflexions sur quelques mécanismes de détection des nœuds à mettre à jour

En l'absence de panne persistance, il n'y a pas de mise à jour à effectuer puisque chaque nœud applique directement la stratégie de rerouting qui est prévue dans sa table de routage en cas de panne d'un nœud ou d'une liaison.

Lorsqu'une panne persistante est détectée par le contrôleur dans le réseau, il détermine les nœuds dont les tables de routage doivent faire l'objet d'une mise à jour. Lorsqu'un flux atteint déjà G_b , la partie bleue du graphe G , il suit tout simplement l'arbre de routage nominal jusqu'à destination. A priori, les nœuds de G_r , la partie rouge du graphe G , qui appartiennent au chemin de rerouting auront besoin d'être mis à jour. Toutefois, en dehors de ces nœuds, il existe d'autres dans G_r qui auraient aussi besoin d'être mis à jour afin d'éviter des cycles ou des surcharges du pont choisi lors du rerouting, par des flux pouvant être acheminé à travers d'autres sorties potentielles de G_r : ces nœuds seront appelés *nœuds frontières*. Il s'agit des nœuds situés à un saut d'un nœud de G_b . Parmi ces nœuds de frontières, nous distinguons ceux qui sont le plus proche du nœud qui détecte la panne de liaison : ce sont les *nœuds critiques*. La mise sur pieds d'un bon mécanisme de détection de ces différents types de nœuds permettrait l'implantation d'une stratégie de mise à jour efficace des tables de routage.

De ce fait, nous proposons et menons dans la suite, une réflexion sur plusieurs mécanismes de détection inspirés de [Son \(2014\)](#) et visant à l'améliorer. En rasant tour à tour les limites de ces mécanismes, nous aboutissons à un mécanisme plus complet.

III.2.1. Mécanisme N°1 : Redirection de la totalité du flux sur le pont

Il est question dans cette stratégie de réorienter tous les flux reroutés sur le chemin de rerouting, et donc sur le pont principal. Les nœuds à mettre à jour sont ceux appartenant au chemin de rerouting utilisé dans la partie rouge pour rerouter le flux de la panne temporaire de liaison. Pour le cas présenté à la figure 32 par exemple, le chemin $2 \rightarrow 5 \rightarrow 10 \rightarrow 15 \rightarrow 17$ est celui utilisé dans G_r pour rerouter le flux issu de la panne temporaire de la liaison (2-1). Lorsque cette panne est déclarée persistante par le contrôleur, les nœuds à mettre à jour sont : 2, 5, 10, 15, 17. Ce mécanisme est celui utilisé dans ([Son, 2014](#)). Puisque le routage de flux repose sur une structure d'arbre nominal enraciné au nœud N_f (ici le nœud 17) à l'instant où la panne se produit, ce nœud est donc un point de sortie pour les flux de la partie rouge vers la partie bleue G_b ; par conséquent, tout flux contenu dans G_r passe par ce nœud N_f à destination du nœud 1 qui est la racine de l'arbre nominal de routage

global, contenant G_r et G_b . L'algorithme 3 décrit ce mécanisme de détection des nœuds à mettre à jour.

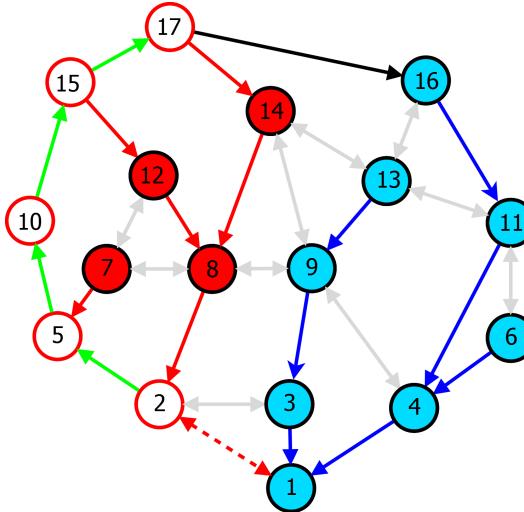


Figure 32 – Mécanisme N°1.

Algorithme 3 : Mécanisme N°1

Entrées : Un graphe orienté G , une panne de liaison l , le nœud N_f qui a détecté la panne temporaire, le chemin de reroutage R_p ;

Sorties : L'ensemble N_{upd} de nœuds à mettre à jour ;

```

1 début
2   Pour chaque nœud  $n \in R_p$  faire
3     ajouter  $n$  à  $N_{upd}$  ;
4   retourner  $N_{upd}$  ;

```

L'inconvénient majeur de cette stratégie c'est la surcharge de certaines liaisons qui peut conduire à une augmentation considérable du délai de transit et du taux de perte des paquets, nuisant ainsi à la QoS dans les réseaux virtuels. De plus, le poids de flux supporté par le pont peut devenir vraiment énorme face au traitement des pannes de liaison multiples.

III.2.2. Mécanisme N°2 : Recherche de ponts secondaires à partir des fils directs de l'extrémité du pont dans la partie rouge G_r

Afin d'améliorer le mécanisme N°1, nous pouvons chercher des ponts secondaires à partir des nœuds frontière. L'intérêt de ces ponts secondaires c'est la minimisation de la hauteur de l'arbre nominal en réduisant la distance entre les noeuds et la destination. Pour trouver les nœuds frontière, on peut commencer à parcourir

G_r à partir de l'extrémité du pont située dans celle-ci. On s'intéresse à cette extrémité parce qu'elle est celle qui offre la plus forte probabilité de trouver rapidement des nœuds frontière, car se trouvant à la frontière des deux parties. Ainsi, on explore les fils directs de cette extrémité à la recherche d'un nœud fils se trouvant dans G_r et située à un saut d'un nœud de G_b et appartenant à l'arbre nominal de routage. Lorsqu'un tel nœud est trouvé, on explore ses fils à la recherche d'autres nœuds comme lui, jusqu'à l'obtention d'un nœud critique (voir algorithme 4).

Algorithme 4 : Mécanisme N°2 de détection de nœuds à mettre à jour

Entrées : Un graphe orienté G , une panne de liaison l , le nœud N_f qui a détecté la panne temporaire, le chemin de reroutage R_p , deux nœuds $k, m \in G_r$, un nœud $p_i \in G_b$ avec $i \in (1, 2, \dots)$, nœud $j \in G_r \cap R_p$ telque l'arc $(j \rightarrow p_i) \in R_p$ existe;

Sorties : L'ensemble N_{upd} de nœuds à mettre à jour ;

1 début

```

2   |    $k \leftarrow j$ ;
3   |   Pour chaque nœud  $n \in R_p$  faire
4   |       |   ajouter  $n$  à  $N_{upd}$ ;
5   |   répéter
6   |       |    $m \leftarrow$  nœud fils direct de  $k \in G_r \setminus R_p$  telque  $(m \rightarrow p_i)$  existe;
7   |       |   ajouter  $m$  à  $N_{upd}$ ;
8   |       |    $k \leftarrow m$ ;
9   |   jusqu'à  $k = N_f$ ;
10  |   retourner  $N_{upd}$ ;
```

L'inconvénient de cette stratégie est qu'elle considère que tous les nœuds fils directs de nœuds frontières sont forcément des nœuds frontières. En effet, considérons la figure 33a. L'application d'une telle stratégie de recherche conduirait à la sélection des nœuds 8 et 14 comme nœuds frontières, justement parce que le nœud 8 donne un pont vers G_b et il est lié au nœud 14 qui fournit aussi un pont. Ce nœud 14 est directement lié au nœud 17 qui est l'extrémité du pont dans G_r . Cependant, en supposant l'existence d'un arc $(14 \rightarrow 12)$ à la place de l'arc $(14 \rightarrow 8)$ (voir 33b) dans l'arbre nominal, l'on ne pourra jamais atteindre le nœud 8 qui est pourtant un nœud frontière ; la recherche s'arrêtera au niveau du nœud 12 parce que n'étant pas un nœud frontière.

III.2.3. Mécanisme N°3 : Recherche de ponts secondaires à partir des fils de l'extrémité du pont dans la partie rouge G_r

Pour pallier au manque d'efficacité du mécanisme N°2, nous pouvons enlever la contrainte liée au choix de fils directs des nœuds impliqués dans la recherche ;

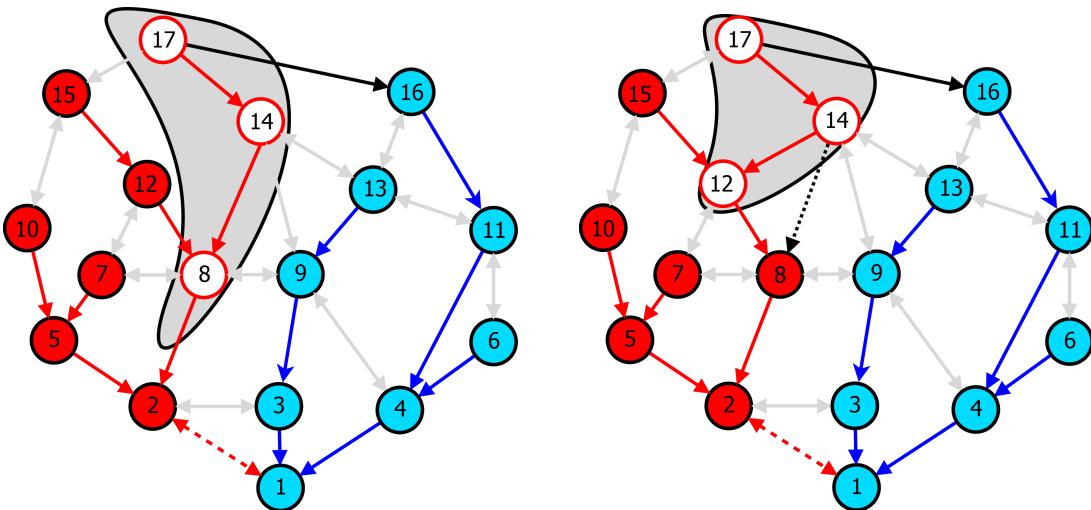


Figure 33 – Mécanisme de recherche N°2.

de ce fait, nous sélectionnons le nœud qui appartient à l’arbre nominal en cas de manque d’un fils direct respectant la condition de connectivité à G_b , et ensuite nous recherchons les nœuds frontière parmi ses descendants directs ou non. On arrête la recherche lorsqu’on rencontre le nœud N_f ou alors le nœud conduisant directement à ce dernier par l’arbre nominal (voir algorithme 5).

Algorithme 5 : Mécanisme N°3 de détection de nœuds à mettre à jour

Entrées : Un graphe orienté G , une panne de liaison l , le nœud N_f qui a détecté la panne temporaire, le chemin de reroutage R_p , deux nœuds $k, m \in G_r$, un nœud $p_i \in G_b$ avec $i \in (1, 2, \dots)$, nœud $j \in G_r \cap R_p$ telque l’arc $(j \rightarrow p_i) \in R_p$ existe, nœud $s \in G_r$ telque l’arc $(s \rightarrow N_f)$ existe;

Sorties : L’ensemble N_{upd} de nœuds à mettre à jour ;

1 **début**

```

2   |    $k \leftarrow j$ ;
3   |   Pour chaque nœud  $n \in R_p$  faire
4   |   |   ajouter  $n$  à  $N_{upd}$ ;
5   |   répéter
6   |   |    $m \leftarrow$  nœud fils de  $k \in G_r \setminus R_p$  telque l’arc  $(m \rightarrow p_i)$  existe ;
7   |   |   ajouter  $m$  à  $N_{upd}$ ;
8   |   |    $k \leftarrow m$ ;
9   |   jusqu’à ( $k = N_f || k = s$ );
10  |  return  $N_{upd}$ ;
```

Cette stratégie permet certes d’avoir des nœuds frontières, mais pas la totalité. En effet, considérons l’existence d’un nœud 18 possédant la configuration de la

figure 34. D'après cette configuration, le nœud 18 est un nœud frontière, mais ce dernier ne sera pas détecté en utilisant le mécanisme N°3.

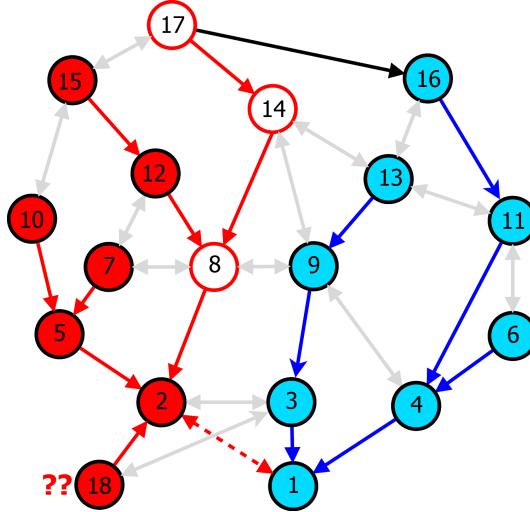


Figure 34 – Mécanisme N°3.

III.2.4. Mécanisme N°4 : Exploration à partir du nœud frontière N_f

Pour déterminer les nœuds frontière, on commence par regarder l’arbre enraciné au nœud N_f de G_r , initiateur du reroutage. Ensuite on détermine la liste de ses fils directs ou indirects à travers l’arbre nominal, situés à un saut d’un nœud de G_b . Chacun de ces nœuds frontière est une extrémité d’un pont secondaire permettant de relier les deux parties du réseau. Ces ponts secondaires sont alors utilisés comme points de sorties de certains flux de G_r à destination de G_b . La recherche de ces nœuds frontière se fera jusqu’à la rencontre de l’extrémité du pont principal situé dans G_r (voir algorithme 6).

Pour illustrer cette stratégie de détection des nœuds frontière, considérons la figure 35. L’arbre nominal de routage de départ est représenté par la figure 35a. La panne de la liaison (2-1) induit la subdivision du graphe de départ en deux parties suivant la stratégie de Son (2014). Le nœud 2 va initier le reroutage suivant le chemin $2 \rightarrow 5 \rightarrow 10 \rightarrow 15 \rightarrow 17$ dans G_r et choisir la liaison (17-16) comme pont pour atteindre G_b . Lorsque le contrôleur se rend compte de la persistance de la panne, il commencera par sélectionner le premier nœud fils direct du nœud 2 et situé à un saut d’un nœud de G_b comme nœud frontière : il s’agit ici du nœud 8 ; ensuite, parmi les nœuds fils du nœud 8 situés à un saut d’un nœud de G_b et se trouvant dans l’ancien chemin nominal, nous avons le nœud 14. Ce nœud est également sélectionné comme nœud frontière (voir figure 35b).

Algorithme 6 : Mécanisme N°4 de détection de nœuds à mettre à jour

Entrées : Un graphe orienté G , une panne de liaison l , le nœud N_f qui a détecté la panne temporaire, le chemin de reroutage R_p , deux nœuds $k, m \in G_r$, un nœud $p_i \in G_b$ avec $i \in (1, 2, \dots)$, nœud $j \in G_r \cap R_p$ tel que l'arc $(j \rightarrow p_i) \in R_p$ existe;

Sorties : L'ensemble N_{upd} de nœuds à mettre à jour ;

- 1 **début**
- 2 $k \leftarrow N_f$;
- 3 **Pour chaque nœud** $n \in R_p$ **faire**
- 4 ajouter n à N_{upd} ;
- 5 **répéter**
- 6 $m \leftarrow$ nœud fils de $k \in G_r$ tel que $(m \rightarrow p_i)$ existe ;
- 7 ajouter m à N_{upd} ;
- 8 $k \leftarrow m$;
- 9 **jusqu'à** $k = j$;
- 10 **retourner** N_{upd} ;

Une fois tous les nœuds frontière détectés, on redirige tous les flux de l'arbre nominal débouchant sur ces nœuds vers les différents ponts secondaires ; ainsi, les ponts secondaires (8-9) et (14-13) seront utilisés pour transférer les flux entrant du nœud 8 vers G_b . Ensuite, les nœuds du chemin de reroutage seront également mis à jour selon l'ordre des flèches indiqué à la figure 35c.

Cette stratégie a l'avantage d'alléger le flux porté par le chemin de reroutage, mais son utilisation de la totalité de ponts secondaires peut nuire gravement au reroutage de futures pannes de liaisons dont le reroutage aurait été effectué et rendu possible à travers les ressources de ces ponts. De plus, cette façon de faire déstructure le réseau en détruisant la structure d'arbre nominal sur laquelle repose la stratégie de reroutage utilisée par le contrôleur, pour implémenter les stratégies de reroutage selon les différentes configurations de pannes, et intégrer celles-ci dans les nœuds à l'avance. De plus, le problème d'existence d'un potentiel nœud 18 mentionné dans le mécanisme N°2 reste sans solution et le nombre de possibilités à explorer pour obtenir l'ensemble des nœuds critiques est élevé, la majorité de ces possibilités n'étant pas utiles pour certaines configurations. Cette stratégie ne répond donc pas à nos objectifs de départ.

III.2.5. Mécanisme N°5 : Elimination du critère d'arrêt régissant la recherche des nœuds frontières dans le mécanisme N°4

Le critère régissant l'arrêt de la recherche des nœuds frontière dans le mécanisme N°4 ne permettra pas de mettre fin au processus de recherche dans cer-

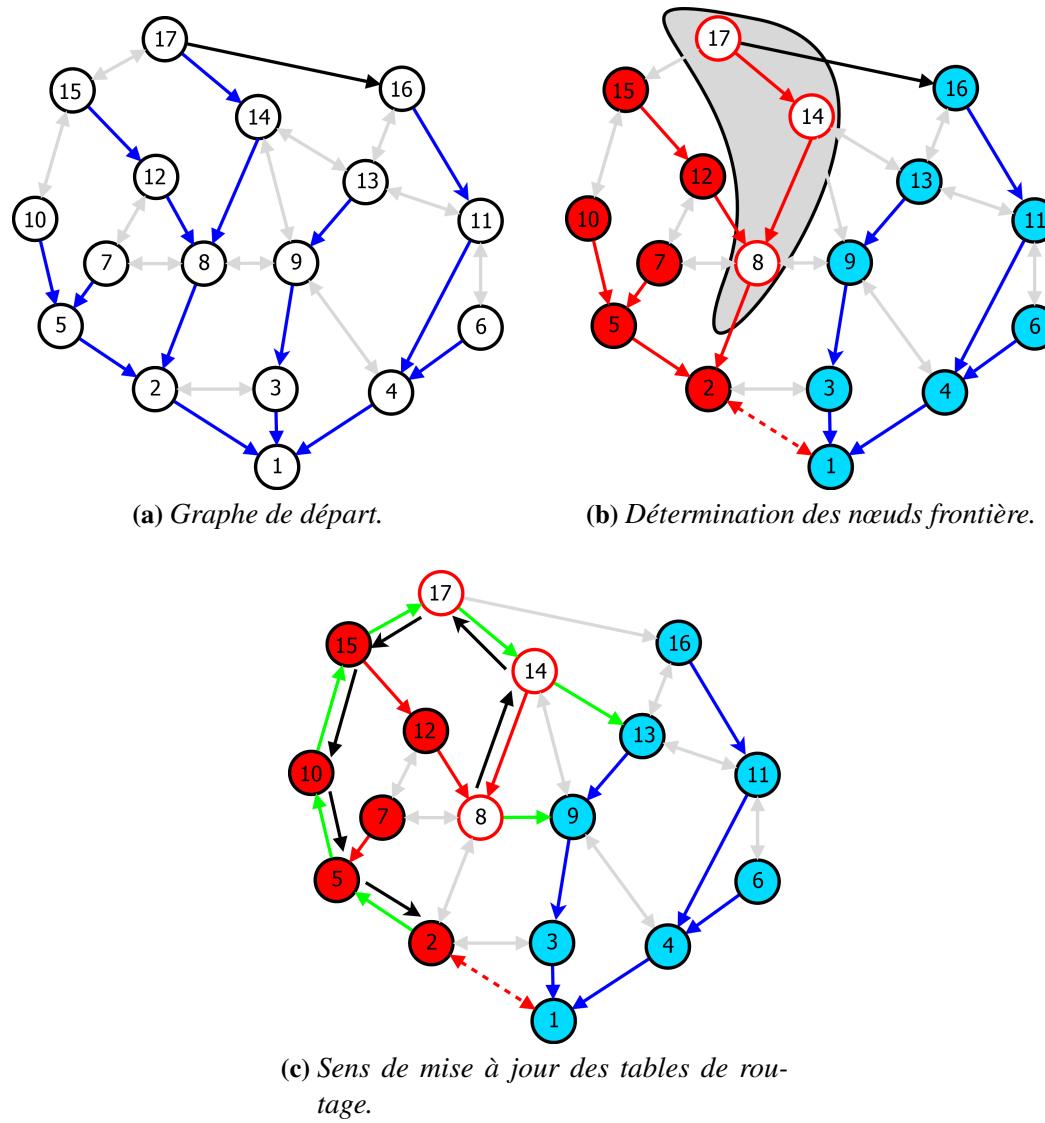


Figure 35 – Mécanisme de recherche N°4.

taines configurations. Par exemple, considérons le cas de la figure 36. L’application du processus de recherche du mécanisme N°4 suivant la portion de chemin $18 \rightarrow 19 \rightarrow 20 \rightarrow 21$ ne s’arrêtera jamais, puisqu’aucun de ces nœuds n’est connecté via le chemin nominal à l’extrémité du pont principal situé dans G_r . Le mécanisme N°5 remplace donc ce critère en procédant à la recherche jusqu’à un nœud directement connecté à G_b , c’est à dire jusqu’au dernier nœud frontière (voir algorithme 7).

L’inconvénient majeur de cette stratégie de détection est qu’elle explore tous les nœuds fils d’un nœud de frontière avant de se rendre compte que ce nœud est le dernier nœud frontière dans une branche d’arbre. Cela augmente le temps de recherche des nœuds frontière. La solution que nous proposons dans notre méca-

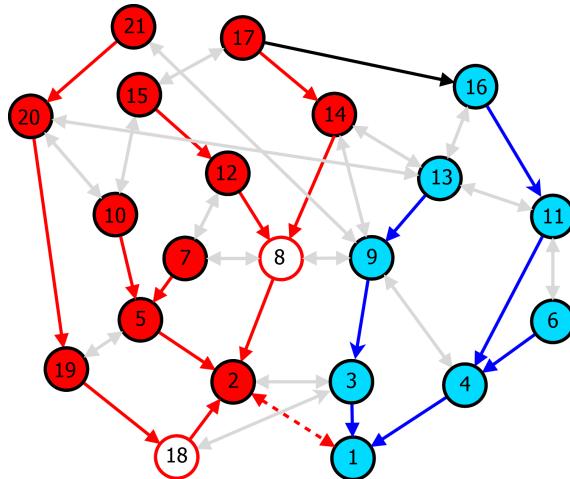


Figure 36 – Mécanisme N°5.

Algorithme 7 : Mécanisme N°5 de détection de nœuds à mettre à jour.

Entrées : Un graphe orienté G , une panne de liaison l , le nœud N_f qui a détecté la panne temporaire, le chemin de reroutage R_p , deux nœuds $k, m \in G_r$, un nœud $p_i \in G_b$ avec $i \in (1, 2, \dots)$, nœud $j \in G_r \cap R_p$ tel que l'arc $(j \rightarrow p_i) \in R_p$ existe

Sorties : L'ensemble N_{upd} de nœuds à mettre à jour

```

1 début
2    $k \leftarrow N_f$  ;
3   Pour chaque nœud  $n \in R_p$  faire
4     ajouter  $n$  à  $N_{upd}$  ;
5   répéter
6      $m \leftarrow$  nœud fils de  $k \in G_r$  tel que  $(m \rightarrow p_i)$  existe ;
7     ajouter  $m$  à  $N_{upd}$  ;
8      $k \leftarrow m$  ;
9   jusqu'à  $k \equiv \phi$ 
10  retourner  $N_{upd}$  ;

```

nisme de recherche est d'arrêter le processus de recherche dans une branche dès que le premier nœud frontière est trouvé.

III.3. Notre stratégie de mise à jour de tables de routage en environnement centralisé

Les différents mécanismes de détection des nœuds frontière précédemment présentés possèdent tous des inconvénients majeurs qui empêchent de satisfaire globalement les contraintes suivantes :

- la hauteur de l'arbre nominal doit être minimale ;

- l’arbre obtenu après la mise à jour des tables doit rester nominal ;
- le minimum de flux doit être acheminé sur le chemin de reroutage par rapport à chaque cas de panne de liaison persistante.

III.3.1. Généralisation de notre approche de détection des nœuds critiques

Considérons un réseau représenté par le graphe $G(N,L)$ où N désigne l’ensemble de nœuds et L l’ensemble des liaisons. Ce réseau est sujet à une panne persistante de liaison ($p - q$) (voir figure 37). Soit $|N| = \text{Cardinal}(N)$ le nombre de nœuds et $|L| = \text{Cardinal}(L)$ le nombre de liaisons.

Soit un nœud d , destination des flux de $G(N,L)$. La panne d’une liaison ($p - q$) engendre le sous-graphe G_r qui est un arbre enraciné au nœud p . Grâce à la structure d’arbre nominal, il existe pour chaque nœud de G_r un chemin permettant de le relier à p . Les liaisons ainsi que les nœuds en traits interrompus indiquent la présence possible d’un sous graphe à ce niveau. D’après la technique proposée par Pham et al. (2012), il existe au moins un sous arbre enraciné en r contenant des nœuds r_i et permettant de relier une extrémité i de G_r à un nœud s de G_b : Il s’agit du pont ($i - s$). De même, il peut exister suivant les configurations, des sous arbres N_c de G enracinés en des k_i et composés des nœuds $k_j \neq i$ permettant de relier directement G_r à G_b .

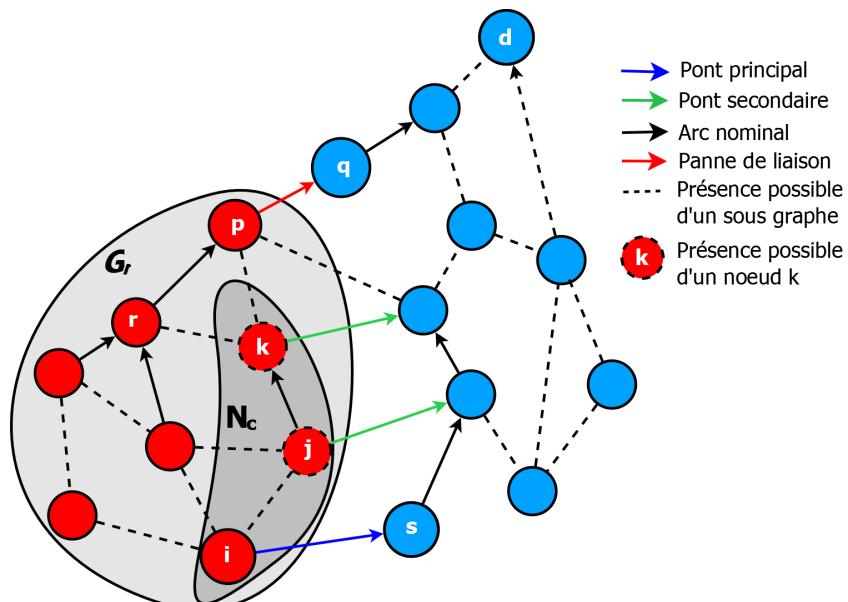


Figure 37 – Notre stratégie généralisée de détection des nœuds à mettre à jour.

Notre stratégie de détection de nœuds critiques s’appuie sur l’usage d’un seul pont par chemin, au lieu de plusieurs comme c’est le cas des mécanismes N°3,

N°4 et N°5 ; la recherche se fait à partir du nœud p détecteur de la panne et on utilise le pont secondaire dont l'extrémité i dans Gr est la plus proche de p , car ce dernier est le plus à même de conduire à un cycle via le chemin nominal. Nous recherchons principalement des nœuds critiques, contrairement aux stratégies précédentes. De ce fait, à partir de ce nœud p et pour chaque sous arbre enraciné en ce nœud, on détermine le nœud critique correspondant et seul ce nœud sera mis à jour, afin de réduire au maximum les risques de cycles. Le parcours d'une branche de l'arbre s'arrête dès qu'un nœud critique a été trouvé. Cette stratégie permet même de détecter le nœud 18 qui pose problème dans les mécanismes N°3 et N°4. Par exemple, pour le cas de la figure 35b, le nœud 8 est l'extrémité du pont secondaire (8-9) la plus proche du nœud 2 qui détecte la panne ; il est donc un nœud critique. Seul le pont secondaire (8-9) sera exploité au lieu de (8-9) et (14-13) comme c'est le cas dans le mécanisme N°4. En ce qui concerne les cas présentés par les figures 34 et 35, le nœud 18 sera sélectionné comme nœud critique et la recherche suivant le sous arbre enraciné en ce nœud sera arrêté directement. Cette façon de faire a l'avantage de réduire le poids des flux renvoyé sur le pont principal, ce qui suggère une disponibilité de davantage de ressources pour les futures pannes. Par ailleurs, le temps de recherche des nœuds critiques est considérablement réduit par rapport au mécanisme N°4, puisqu'une fois qu'on a un nœud critique sur une branche, la recherche s'arrête. Pourtant dans le mécanisme N°4, la recherche se poursuit jusqu'à ce que tous les nœuds frontières sur la branche soient détectés.

III.3.2. Principe général de mise à jour

La technique de détection que nous proposons permet de localiser les nœuds critiques dont les tables de routage feront l'objet des mises à jours, en plus des nœuds du chemin de reroutage qui d'office ont besoin d'être mis à jour à cause du reroutage. Les nœuds critiques que nous obtenons étant tous indépendants les uns des autres car situés dans des branches différentes, le contrôleur peut donc procéder à leur mise à jour simultanément à travers des *packet-out message*, sans redouter des cycles. Par contre, en ce qui concerne les nœuds du chemin de reroutage, la mise à jour va s'effectuer de proche en proche à partir de l'extrémité i du pont situé dans G_r , vers le nœud p détecteur de la panne de liaison.

Puisque les nœuds se situant sur le chemin de reroutage ont exploité leur table de routage afin de transporter le flux rerouté, la mise à jour pour ces nœuds va consister à appliquer ces configurations de reroutage dans leur table de routage en tant que nouvelles configurations de routage. Les nœuds critiques quant à eux verront leur table de routage modifiée afin de s'adapter à l'utilisation du pont se-

condaire ; leur table de reroutage doit également être reconstruite en tenant compte de cette nouvelle configuration des tables de routage ; il en est de même pour les nœuds des chemins de reroutage. Dès lors, les stratégies de reroutage précédemment mises au point par le contrôleur, seront recalculées par celui-ci et réintégrées dans les divers nœuds de G_r , faute de quoi l'arbre de routage ne serait plus nominal ni optimal ; la conséquence dans ce cas, serait le prolongement du temps de transit des paquets.

En considérant la figure 38a, la mise à jour des tables de routage des nœuds situés dans le chemin de reroutage sera faite suivant l'ordre $17 \rightarrow 15 \rightarrow 10 \rightarrow 5 \rightarrow 2$. Au niveau du nœud 17, il sera surtout question de réorienter les flux destinés à l'arc $17 \rightarrow 14$ vers le pont $17 \rightarrow 16$. Les arcs de type $15 \rightarrow 12$ seront remplacés par $15 \rightarrow 17$, pour aboutir au graphe de la figure 38b.

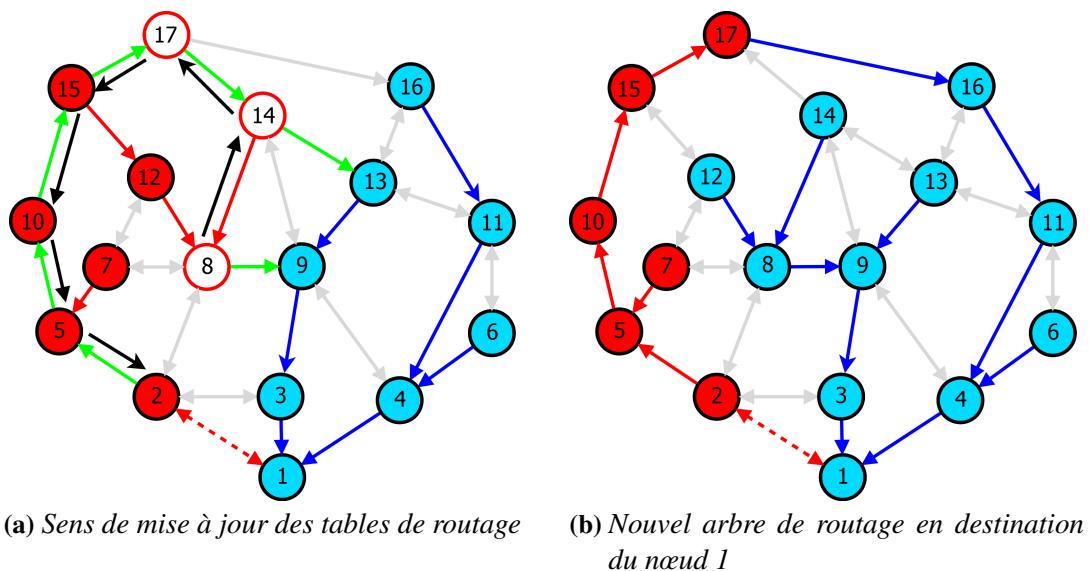


Figure 38 – Structure du réseau après mise à jour.

III.3.3. Mise à jour interne de proche en proche des nœuds du chemin de reroutage

Structure de notre message de mise à jour

Le message de mise à jour envoyé par le contrôleur contient une valeur k qui représente la position de l'ensemble des mises à jour correspondantes à un nœud donné, ainsi qu'un vecteur de mise à jour. Chaque nœud connaît donc la position de son vecteur de mise à jour dans le vecteur de liste de mise à jour grâce à k . Chaque entrée du vecteur de mise à jour est une liste de triplets (a, b, c) semblable à la structure des messages LSU du protocole OSPF (voir figure 39) où a représente

le port d'entrée du flux à modifier, b est le nouveau port de sortie de ce flux et c représente le port sur lequel il faut acheminer le reste du message. Si le nœud est un switch, alors a est une adresse MAC ; si le nœud est plutôt un routeur, alors a est une adresse IP. Pour les nœuds qui nécessitent plusieurs mises à jours, seul le dernier triplet aura sa valeur $c \neq 0$. Chaque nœud concerné lit la valeur k , récupère son vecteur de triplet, effectue sa ou ses mises à jour, puis incrémenté la variable k et renvoie le message au nœud suivant dans la liste ordonnée. Le dernier vecteur de triplets aura la valeur c de son dernier triplet nulle, pour signifier l'arrêt de transmission du message. Chaque nœud lorsqu'il reçoit son vecteur de triplets de mise à jour, vérifie si la valeur du port de sortie de son dernier triplet est non nulle ; si tel est le cas, alors après avoir procédé à sa mise à jour, il procède à la destruction du message de mise à jour, sinon il l'achemine vers le port de sortie.

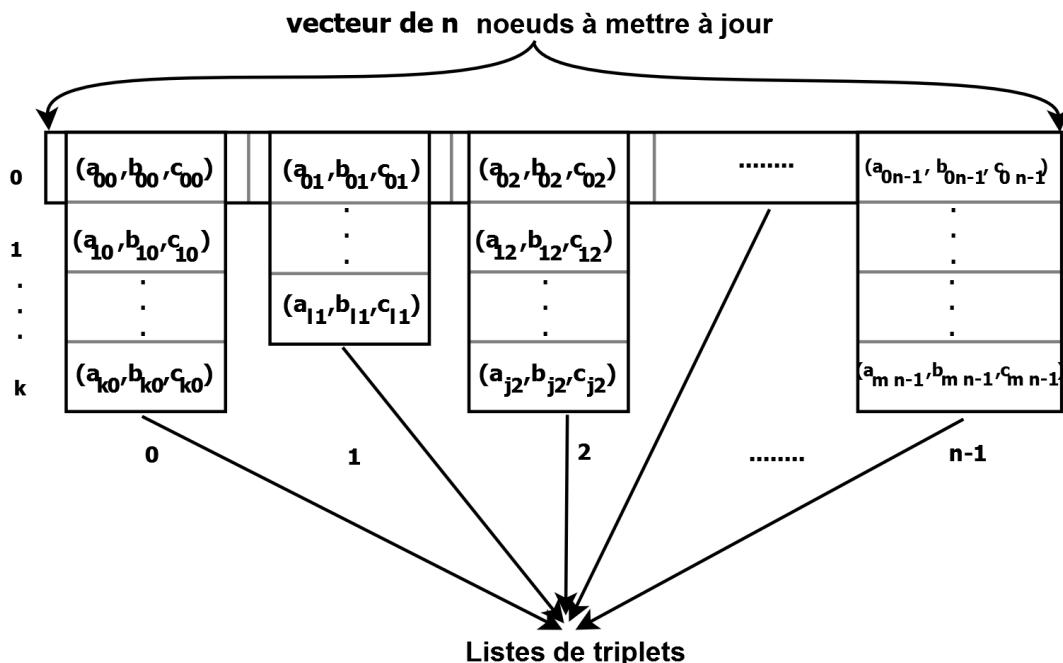


Figure 39 – Vecteur de liste de triplets de mise à jour.

L'évolution de la mise à jour interne à partir de ce vecteur de triplets est représentée par la figure 40. La valeur entière des éléments de ces triplets est juste à titre explicatifs pour une meilleure clarté ; en réalité, la valeur du premier élément de chaque triplet est soit une adresse MAC, soit une adresse IP en fonction de l'équipement. Cette figure présente la mise à jour successive des nœuds 17, 15, 10 et 5 de la figure 36.

La contrainte du MTU (Maximum transmission Unit)

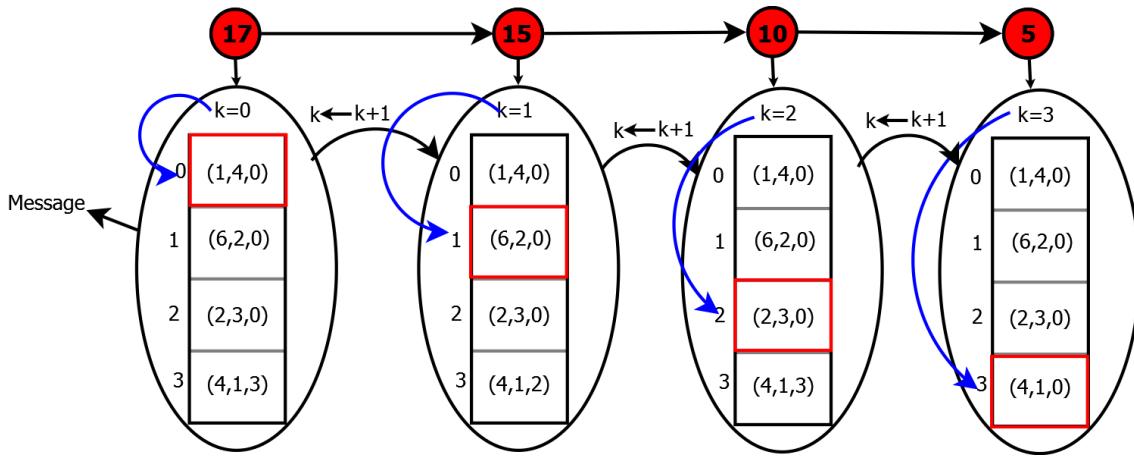


Figure 40 – Mise à jour d'un nœud à un autre avec le vecteur de triplet.

Le maximum transmission unit (MTU) est la taille maximale d'un paquet pouvant être transmis en une seule fois (sans fragmentation) sur une interface. Cette valeur permet de déterminer la taille de données à transmettre sur les interfaces réseaux afin d'assurer leur acheminement. La structure du message de mise à jour (vecteur de listes) mis en place par le contrôleur peut rapidement grandir avec un nombre important de nœuds à gérer dans ce processus, risquant ainsi d'atteindre rapidement la valeur limite du MTU. Par exemple, les valeurs de la MTU selon le type de réseau sont les suivantes :

- IPv4 : le MTU minimal est de 68 octets ([Postel, 1990](#));
- IPv6 : le MTU minimal par défaut vaut 1280 octets;
- sur Ethernet, MTU = 1500 octets par défaut ([Crawford, 1998](#));
- dans le cas de la carte réseau locale (ou localhost : l'hôte local en anglais), MTU = 16436 octets.

Compte tenu de cette valeur du MTU, le paquet de mise à jour des nœuds devra subir une fragmentation avant d'être transmis.

La politique de fragmentation du paquet de mise à jour originel

La fragmentation du paquet de mise à jour originel consistera à diviser le vecteur de base en plusieurs sous vecteurs qui seront affectés suivant leurs index aux nœuds concernés. La taille de ces sous vecteurs sera équivalente à la valeur du MTU des interfaces de transmission.

Pour la mise à jour, le groupe de nœuds concernés sera subdivisé en sous-groupes auxquels seront affectés les fragments de message de mise à jour les concernant. A la figure 41 par exemple, nous avons trois groupes. Le message original est subdivisé en trois parties et transmis simultanément au nœud à l'entrée de chaque groupe. le premier groupe démarre son processus de mise à jour à la

suite de la réception d'un *packet-out message* du contrôleur. Les autres groupes ne déclenchent leur processus de mise à jour qu'à la réception d'un message d'acquittement venant du groupe qui le précède. Ce message d'acquittement signale la fin du processus de mise à jour dans le groupe précédent. Celà permet d'éviter les pertes de trafic à l'intérieur et entre les goupes.

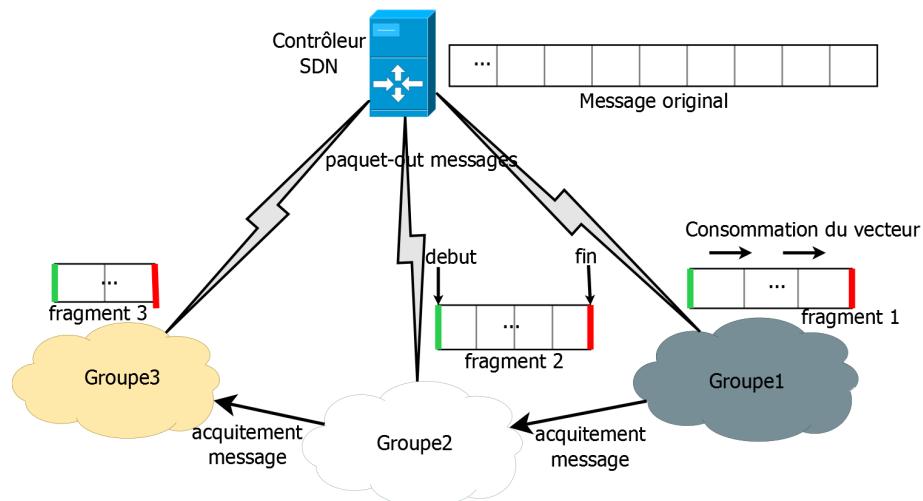


Figure 41 – Fragmentation du message de mise à jour.

III.3.4. Reconstruction des tables de reroutage en fonction des nouvelles configurations de routage

Les nœuds intervenant dans le processus de mise à jour doivent subir des modifications non seulement au niveau de leurs tables de routage, mais aussi de leur table de reroutage, les anciennes stratégies de reroutage précédemment intégrées dans les équipements étant devenues obsolètes ; le contrôleur doit ainsi recalculer les nouveaux chemins de reroutage en utilisant la technique proposée par Pham et al. (2012). Pour ce faire, il se construit en interne un nouvel arbre nominal sur la base duquel il détermine de nouveaux chemins de reroutage crédibles ; cela permet d'avoir une QoS comparable à celle du réseau en l'absence de pannes.

Pendant la période de recalculation de ces nouveaux chemins, le réseau fonctionne sur la base de l'ancienne stratégie de reroutage et utilise les ponts secondaires pour maintenir l'acheminement des flux de bout en bout ; par conséquent, le réseau utilise des chemins pas toujours optimaux durant cette période. A cause des contraintes de délai de transit et du taux de perte des paquets (Almes et al., 1999a) liés à la QoS, ces temps d'attente de recalculation et de nouvelles configurations par les différents nœuds doivent être minimaux ; raison pour laquelle les opérations de détermination de nœuds critiques et de recalculation des nouveaux chemins de routage

et reroutage doivent toutes se faire au préalable dans le contrôleur avant que ce dernier n'initie tout procédé de mise à jour des tables.

Concernant la mise à jour des tables de routage et des informations de reroutage, elles peuvent se faire en même temps au niveau de chacun des nœuds impliqués, puisque le contrôleur ayant déjà transmis les informations, aucun recalcul n'est plus nécessaire au niveau de ces nœuds.

III.3.5. Intérêts de notre méthode

Notre stratégie de mise à jour des tables de routage soucieuse de la QoS dans les réseaux virtuels, présente les avantages suivants :

- **l'absence des cycles dans le réseau :** la recherche de nœuds critiques permet d'éviter le routage des flux vers le nœud initiateur du reroutage. Ce nœud via le chemin nominal peut réacheminer les flux vers la liaison en panne ;
- **l'allégement du flux supporté par le chemin de reroutage :** le fait d'utiliser un pont secondaire permet cet allégement. Une partie du flux de Gr est transporté via un autre pont pour la partie bleue. De même, la redirection de tout flux destiné à tout arc nominal ($i \rightarrow j$) partant de l'extrémité du pont se trouvant dans Gr , en direction de tout autre nœud de Gr , vers le pont ($i - s$), participe à cet avantage ;
- **la conservation de la structure du réseau :** la mise en place d'un nouvel arbre nominal de routage dans Gr contribue à garantir cette conservation. Cela permet ainsi de maintenir le réseau apte à faire face aux futures pannes de liaisons en utilisant la stratégie IPFRR. Par ailleurs, l'utilisation d'un minimum de ponts contribue à assurer la disponibilité d'autres ponts pour les potentielles futures pannes de liaisons, car cela offre plusieurs choix de reroutage optimaux pour une panne donnée ;
- **la prévention de la congestion rapide dans le réseau :** il s'agit ici d'une conséquence immédiate de l'usage de pont secondaire. En usant de ce pont secondaire, les limites des ressources réseau ne sont pas rapidement atteintes, offrant de ce fait la possibilité de prendre en main du maximum de pannes possibles dans un contexte de pannes de liaisons multiples simultanées ou non-simultanées persistantes.

La stratégie de mise à jour des tables de routage que nous venons proposer dans cette section pour gérer les pannes persistantes de liaison, se fonde sur un reroutage unicemin des flux. Nous avons supposé initialement que la stratégie de reroutage utilisée pour rerouter les flux en cas de panne passagère de liaison, utilisait un seul

chemin pour le faire. Dans le cas où cette stratégie de reroutage utilise plusieurs chemins à cause de la séparation de flux comme c'est le cas au chapitre II, notre méthode de mise à jour des tables de routage fonctionne aussi. En effet, notre stratégie de mise à jour des tables de routage en cas de persistance d'une panne de liaison, s'appliquera à chacun des chemins de reroutages obtenus de la séparation de flux.

Plusieurs pannes de liaisons pouvant être déterminées persistantes quelques temps après leur survenue, nous étendons notre stratégie de mise à jour des tables de routage présentée ici, à la gestion de ces cas de pannes multiples persistantes dans la suite.

III.4. Extension à la gestion des pannes multiples persistantes

Dans les sections précédentes, nous avons abordé la question de mise à jour des tables de routage et de reroutage dans le cas d'une seule panne de liaison persistante. Cependant, il peut également arriver que l'on assiste à des pannes de plusieurs liaisons en cascade qui persistent dans le temps. Ces pannes peuvent être soit non simultanées, soit simultanées et adjacentes à un même nœud ; on parlera ainsi de panne de nœud lorsque toutes les liaisons adjacentes à ce nœud seront en panne. Quel que soit les cas, la question centrale est celle de savoir si la stratégie précédemment définie pour la panne d'une seule liaison sera aussi applicable dans ce contexte. Si tel n'est pas le cas, quels ajustements doivent-ils être effectués pour rendre cette stratégie utilisable ? L'étude de ces questions fait l'objet des sections suivantes.

III.4.1. Cas de pannes de liaisons multiples non simultanées et persistantes

Plusieurs pannes de liaisons sont qualifiées de non simultanées lorsqu'elles se produisent les unes à la suite des autres. Ces pannes peuvent chacune persister dans le temps (par exemple plus de 10 minutes (Son, 2014)), d'où le qualificatif de pannes persistantes. La détection de la persistance de chaque panne sera alors faite par le contrôleur de manière consécutive. Plusieurs approches de résolution peuvent être envisagées quant à la mise à jour des tables de routage.

III.4.1.1. Première solution : mise à jour consécutive des nœuds au fur et à mesure de la détection de la persistance des pannes

Cette solution stipule de résoudre les cas de panne de liaison persistante au fur et à mesure qu'elles sont détectées par le contrôleur. Chaque cas détecté est résolu en appliquant l'approche proposée à la section III.3.3 traitant de la panne persistante d'une seule liaison. Cette solution est inspirée de celle proposée par Son (2014) et liée au calcul des règles de rerouting pour les cas de pannes de liaisons multiples simultanées et non persistantes. La figure 42 permet d'illustrer cette solution pour les pannes persistantes et non simultanées de deux liaisons.

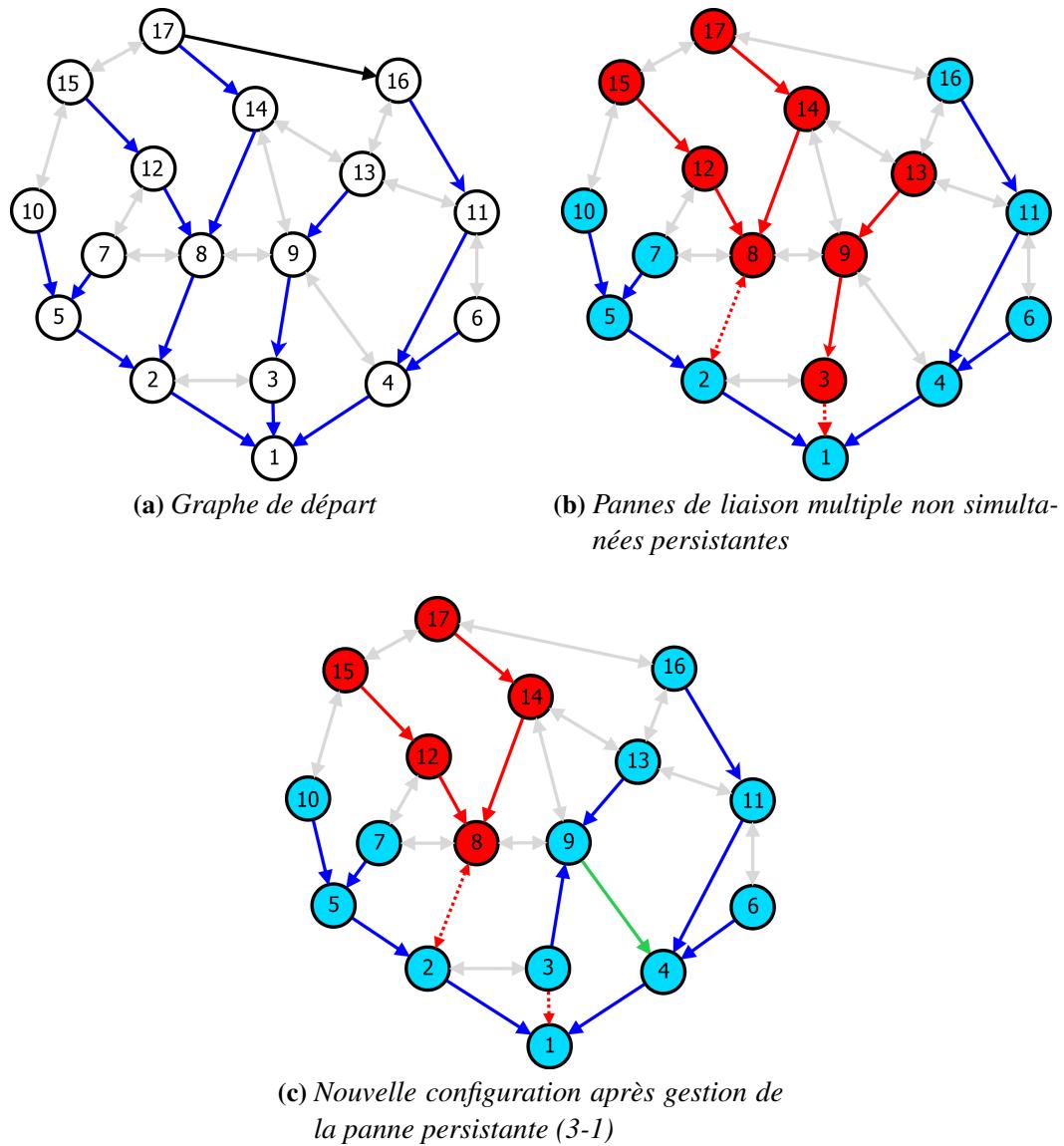


Figure 42 – Première solution : pannes de liaison multiple non simultanées persistantes.

Sur cette figure, considérons par exemple que la persistance de la panne (3-1) soit détectée avant celle de (8-2). Alors, le contrôleur recalcule aussitôt le chemin de reroutage afin d'utiliser le trajet $3 \rightarrow 9 \rightarrow 4$ pour rerouter le flux, et lance dans le réseau la mise à jour des nœuds 3 et 9; cette mise à jour induit la nouvelle configuration présentée par la figure 42c. Le contrôleur démarre directement le calcul de chemin de reroutage (le chemin $8 \rightarrow 14 \rightarrow 13$) pour la panne (8-2) en considérant cette nouvelle configuration, puis envoie le message de mise à jour une fois le calcul terminé.

Cependant, cette solution semble omettre certains cas de figure pertinents pouvant se produire. Par exemple, supposons que le message de mise à jour des nœuds pour la panne (3-1) ne soit pas encore arrivé au niveau du nœud 9 lorsque celui de la panne (8-2) est appliqué dans ce nœud. Alors, le résultat obtenu ne sera pas celui escompté, puisque le deuxième message de mise à jour est conçu à partir de la structure supposée du réseau après la mise à jour des nœuds pour la panne (3-1). De ce fait, si par la suite les informations de mise à jour pour la panne (3-1) arrivent au nœud 9 et portent sur les mêmes entrées que pour la panne de liaison (8-2), les informations précédemment mises à jour seront écrasées, entraînant la conservation de l'ancien chemin de reroutage pour la panne (8-2).

III.4.1.2. Deuxième solution : mise à jour synchronisée des tables de routage des nœuds au fur et à mesure de la détection des pannes de liaisons persistantes

Considérons un réseau soumis à un ensemble de pannes de liaisons non simultanées. Supposons que les liaisons sont bidirectionnelles et que chaque panne de liaison entraîne la subdivision du réseau en deux parties à savoir : G_r qui est la partie contenant le nœud qui détecte la panne, et G_b la partie qui contient la destination de l'arbre de routage nominal. L'inconvénient présenté dans la première solution pose le problème de synchronisation des opérations de mise à jour au sein d'un même nœud, pour des pannes de liaison différentes.

Théorème 2 *Pour toute paire de pannes de liaisons non simultanées, il existe au moins un nœud commun aux sous-graphes G_r et G_b engendrés chacun par ces deux pannes.*

Preuve. Soit un réseau virtuel VN représenté par un graphe $G(N, L)$ où $|N|$ est le nombre de nœuds et $|L|$ le nombre de liaisons. Soit d la destination des flux d'un arbre de routage nominal de G . Soit (l_1, l_2) un couple de pannes de liaisons. Soit n_1 le nœud ayant détecté l_1 , et n_2 celui ayant détecté l_2 . l_1 subdivise G en deux

sous-graphes G'_1 et G'_2 , avec $n_1 \in G'_1$ et $d \in G'_2$ telque $\cup_{i \in (1,2)} G'_i = G$. De même l_2 subdivise G en G''_1 et G''_2 telque $n_2 \in G''_1$ et $d \in G''_2$.

Supposons que $G'_1 \cap G''_1 = \emptyset$ et $G'_2 \cap G''_2 = \emptyset$. On a $d \in G'_2$ et $d \in G''_2 \Rightarrow d \in G'_2 \cap G''_2$; or $G'_2 \cap G''_2 = \emptyset$. Donc $d \in \emptyset$, ce qui est absurde. D'où $G'_2 \cap G''_2 \neq \emptyset$.

$G'_1 \cap G''_1 = \emptyset \Rightarrow n_1 \notin G''_1$ et $n_2 \notin G'_1$. Mais, la panne l_1 crée la subdivision $S_{1|G}$ avec $G'_1 \subset S_{1|G}$ et $G'_2 \subset S_{1|G}$. De même, l_2 crée la subdivision $S_{2|G}$ telque $G''_1 \subset S_{2|G}$ et $G''_2 \subset S_{2|G}$. Puisque chaque panne induit une subdivision du graphe G en deux, on a : $S_{1|G} \subset S_{2|G}$ ou $S_{2|G} \subset S_{1|G}$. Ainsi, $n_1 \in S_{2|G} \Rightarrow n_1 \in G''_1 \Rightarrow n_1 \in G'_1 \cap G''_1$ ce qui est absurde car $G'_1 \cap G''_1 = \emptyset$. \square

Le théorème 2 stipule de l'existence de noeuds communs aux différents sous arbres engendrés par n pannes de liaisons non simultanées et consécutivement persistantes. Ces noeuds en commun sont les plus à même de causer des conflits de mise à jour des tables de routage entre plusieurs messages. Ce théorème nous permet d'envisager deux approches de solutions à ce problème de synchronisation.

Approche 1 : attendre un packet-in message du réseau avant l'envoie des messages de mise à jour pour une autre panne de liaison

Lorsque les informations de mise à jour pour une panne donnée sont en cours d'application dans le réseau, le contrôleur attend de recevoir un packet-in message de la part du dernier groupe de noeuds à être mis à jour, avant d'envoyer les paquets de mise à jour pour une autre panne. En considérant la figure 41 de la section III.3.3, le dernier groupe de paquets est celui du groupe 3, ce qui signifie que le message informant le contrôleur de la fin du processus de mise à jour pour la panne traitée, proviendra de ce groupe. Dans ce groupe, c'est le dernier noeud à être mis à jour qui enverra ce message comme c'est par exemple le cas du noeud 5 de la figure 40 de la section III.3.3. Ce noeud saura qu'il est le dernier en regardant la valeur de c qui est nulle dans le dernier triplet de mise à jour. Cette approche a l'avantage d'éviter les cas de conflits et de perte d'informations de mises à jour pour des pannes différentes ; elle a par contre l'inconvénient majeur d'augmenter les temps de latence.

Approche 2 : attribuer une étiquette référençant le niveau de priorité des messages de mise à jour entre pannes

Il s'agit ici pour le contrôleur d'insérer dans les paquets de mise à jour pour une panne donnée, un ordre de priorité par rapport aux messages suivants qui sont liés à d'autres pannes de liaison détectées persistantes. Cet ordre de priorité est un champ qui définit une date ou un entier dont la valeur est incrémentée à chaque fois qu'une autre panne est détectée.

mentée au fur et à mesure que les pannes persistantes sont détectées. Il n'est donc pas besoin de connaître à l'avance toutes les pannes potentiellement persistantes dans le réseau pour exploiter ce champ. Par ailleurs, le contrôleur doit maintenir en interne la valeur courante de ce champ afin d'éviter l'affectation de la même étiquette à deux messages de mise à jour concernant des pannes différentes.

Lorsque plusieurs messages de mise à jour se rencontrent au sein d'un même nœud, la valeur du champ de priorité est utilisée pour respecter l'ordre de précédence ; cela se fait surtout quand il faut appliquer simultanément plusieurs informations de mise à jour conflictuelles sur une entrée commune de la table de routage du nœud concerné. Dès lors, le message dont la valeur de ce champ est la plus petite, est exploitée avant les autres messages dont les valeurs de ces champs sont les plus grandes. Quant aux informations de mises à jour ne portant pas sur les mêmes entrées, elles sont appliquées simultanément. Cette approche réduit les temps de latence en permettant la circulation simultanée des messages de mise à jour de différentes pannes, contrairement à la solution précédente.

L'approche 2 est celle que nous proposons aux problèmes de mise à jour des nœuds et de synchronisation pour les cas de pannes de liaisons multiples non simultanées. Cette approche 2 exploite celle utilisée pour le cas d'une seule panne de liaison proposée à la section III.3.3, mais avec quelques ajustements :

- un champ est ajouté à la structure du message de mise à jour pour gérer la synchronisation de la mise à jour au niveau des nœuds communs à plusieurs pannes ;
- on suppose la présence d'au moins deux pannes persistantes de liaisons dans le réseau.

L'approche 2 a aussi l'avantage de permettre la conservation de la cohérence des chemins dans le réseau ainsi que la prise en main d'un maximum de pannes de liaisons non simultanées à travers sa méthode de synchronisation des mises à jour. Cette approche permet également de prendre en main les cas de pannes simultanées de deux liaisons non adjacentes au même nœud, mais ne pourrait pas garantir de solution pour plus de trois pannes de liaisons ou même dans les cas de pannes multiples non solutionnable par la méthode de reroutage de Son (2014) lors de la tentative de résolution de la panne passagère de ces liaisons.

En ce qui concerne le cas de panne d'un nœud faisant intervenir simultanément plusieurs pannes de liaisons, notre méthode de synchronisation présentée dans l'approche 2 sera-t-elle aussi valable ?

III.4.2. Cas de la panne persistante d'un nœud

III.4.2.1. Panne persistante d'un nœud virtuel

Une panne de nœud (routeur, switch) physique ou virtuel désigne l'impossibilité de transit d'informations d'un port à un autre de ce nœud. Ce cas de panne induit directement la panne simultanée de plusieurs liaisons : nous pouvons donc également parler de pannes de liaisons multiples simultanées adjacentes au même nœud tel que présenté à la figure 43. Cette figure présente la panne du nœud 14 qui induit la panne des liaisons (14-17), (14-13), (14-9) et (14-8). Nous considérons pour l'instant une seule panne de nœud dans un plan virtuel.

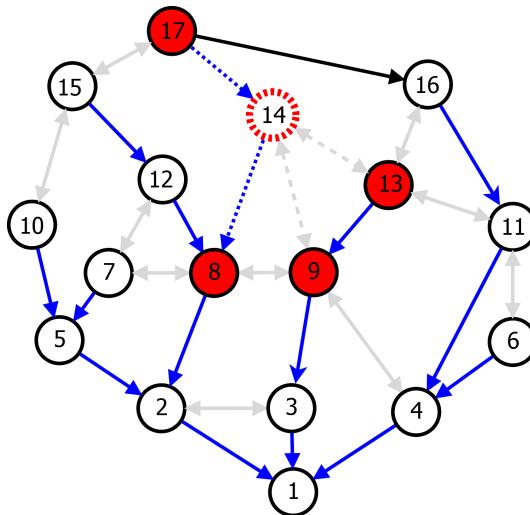


Figure 43 – Panne du nœud 14.

La panne d'un nœud peut alors se voir de deux façons différentes :

- la panne simultanée de toutes les liaisons adjacentes à ce nœud ;
- la détérioration du nœud en question.

Dans ces deux situations, il y a impossibilité de transit de flux d'un bout à l'autre de l'équipement et la détection de la persistance de la panne du nœud se matérialise par celle des pannes de liaisons qui lui sont adjacentes. Pour diverses raisons, la persistance de ces pannes peut être détecté et traité par le contrôleur de plusieurs façons.

Détection de la persistance des liaisons adjacentes les unes à la suite des autres

La détection de la persistance des liaisons est faite les unes à la suite des autres. Dans ce cas, le calcul des nouveaux chemins de reroutage est effectué en utilisant la stratégie de Son (2014) et le processus de mise à jour utilisé

est celui de l'approche 2 présenté précédemment dans le cas de pannes de liaisons multiples non simultanées.

Détection simultanée de la persistance des liaisons adjacentes au nœud en panne

Dans ce cas de figure, les pannes des liaisons ayant causées la panne de nœud sont toutes détectées persistantes au même moment. Suivant la stratégie IP-FRR (IP Fast ReRoute) ([Son, 2014](#)), les nœuds qui détecteront la panne du nœud 14 sont 17, 8, 9 et 13. Ces nœuds détecteurs seront les premiers à être mis à jour afin d'éviter les cycles dans le rerouting ; ensuite suivront les nœuds du chemin de rerouting trouvé en exploitant la stratégie de [Son \(2014\)](#). La mise à jour des nœuds détecteurs consistera principalement à dévier les flux destinés aux liaisons en panne, vers des liaisons disponibles. Pour le cas de la figure [43](#) où le nœud 1 est la destination des flux, les chemins $17 \rightarrow 16 \rightarrow 11 \rightarrow 4 \rightarrow 1$, $8 \rightarrow 2 \rightarrow 1$, $9 \rightarrow 3 \rightarrow 1$ et $13 \rightarrow 9 \rightarrow 3 \rightarrow 1$ peuvent respectivement être utilisés pour prendre en main les pannes de liaisons (14-17), (14-8), (14-9) et (14-13). Le principe de mise à jour pour ce cas est donc le suivant :

- le contrôleur exploite sa vue globale du réseau pour détecter le nœud en panne en fonction de la liste des pannes de liaisons ;
- pour chaque panne de liaison détectée, le contrôleur construit les messages de mise à jour à partir de notre approche proposée à la section [III.3.3](#) et en ajoutant une étiquette comme dans le cas de pannes de liaison non simultanée pour gérer la synchronisation ;
- le contrôleur envoie les informations de mise à jour vers les nœuds détecteurs.

III.4.2.2. Panne persistante d'un nœud physique

Le paradigme de virtualisation réseau implique intrinsèquement le partage d'une infrastructure physique en plusieurs autres qui sont virtuels et abrités par cette infrastructure ([N.M. Mosharaf Kabir Chowdhury, 2010](#)). La détérioration d'un nœud physique implique donc celle de plusieurs nœuds virtuels appartenant tous au même plan ou à des plans différents selon le modèle de mappage qui existe entre le réseau physique et les réseaux virtuels hébergés ([Shahriar et al., 2016](#)). La figure [44](#) présente un environnement de virtualisation réseau composé d'un réseau physique de base et deux réseaux virtuels VN1 et VN2. Les nœuds 3 et 4 de VN1, le nœud 5 de VN2, sont tous hébergés au sein du nœud 2 du réseau physique. Lorsque le nœud physique 2 tombe en panne, il induit directement la panne des nœuds 3 et

4 de VN1, ainsi que celle du nœud 5 de VN2 ; les liaisons physiques et virtuelles également tombent en panne de la même façon.

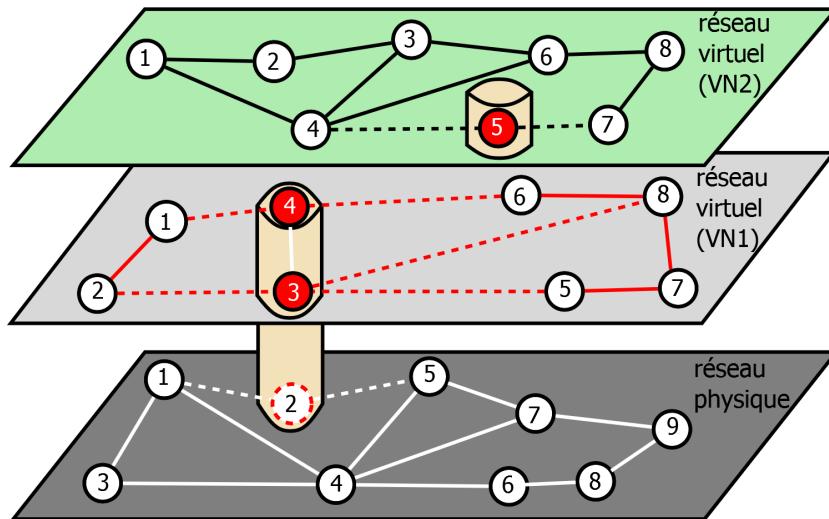


Figure 44 – Panne d'un nœud physique.

Dans ce type de configuration, les solutions proposées dans ([Shahriar et al., 2016](#)) ou encore dans ([Son, 2014](#)) peuvent être exploité pour prendre en main le rerouting. Par ailleurs, les différentes approches de prise en main du rerouting sur plusieurs couches proposées dans le cadre des réseaux optiques (télécommunication) multicouches ([Doumith, 2007](#)) bâti sur une seule infrastructure de réseau physique, nous permet d'envisager plusieurs possibilités en ce qui concerne la mise à jour des tables de routage.

La mise à jour non coordonnée : elle consiste à appliquer notre stratégie proposée précédemment pour la panne de nœud virtuel, dans chaque réseau virtuel. De ce fait, le contrôleur déclenche la mise à jour pour chaque plan virtuel concerné par la panne du nœud physique. L'inconvénient d'une telle approche réside dans le fait qu'elle induit une baisse des performances du réseau suite à une réservation désorganisée de bande passante de protection, surtout en cas d'insuffisance de bande passante résiduelle dans le réseau.

La mise à jour coordonnée : il s'agit de définir la séquentialité du processus de mise à jour des différents réseaux concernés. Mais avant tout, il faut mettre à jour les nœuds nécessaires au niveau du réseau physique (niveau de la panne) avant de s'intéresser aux plans virtuels hébergés : cette stratégie a l'avantage de garantir l'intégrité du réseau pilier, garant de la QoS dans le réseau à travers ses ressources partagées (bande passante, débit, ...). L'approche de mise à jour à utiliser ici est celle que nous avons proposée pour une panne de nœud virtuel.

A cause du problème de dimensionnement du réseau¹ et du mappage des réseaux virtuels par rapport au réseau physique, la panne d'un nœud physique peut conduire à la panne simultanée de plusieurs nœuds virtuels ; c'est le cas par exemple des nœuds virtuels 3 et 4 du réseau virtuel VN1 de la figure 44. Cette hétérogénéité des types de pannes existantes sur l'ensemble de ces réseaux virtuels rend très difficile la coordination du processus de mise à jour. Il faut alors définir des critères de priorité entre plans virtuels afin de gérer les cas de compétition en présence de pannes multiples simultanées persistantes. Ces critères sont définis par le fournisseur d'infrastructure réseau en fonction des objectifs visés. Nous définissons ici deux critères de priorité : **le nombre de nœuds virtuels en panne et la charge du trafic**. Ainsi, selon le but visé, on peut :

- mettre prioritairement à jour les nœuds virtuels du réseau présentant le moins de nœuds virtuels en panne : ceci a l'avantage de réduire au maximum les incohérences dans le routage, d'autant plus que notre approche de mise à jour ne prend pas en main les pannes de nœuds multiples. Pour le cas de la figure 44, les nœuds du réseau virtuel VN2 seront mis à jour prioritairement par rapport à ceux de VN1 ;
- privilégier la mise à jour des nœuds au niveau du réseau virtuel dont le trafic est le plus élevé : l'avantage de cette approche est la fidélisation des clients à travers une QoS orientée trafic.

III.5. Résultats de simulations

Les différents processus de mise à jour décris précédemment ont été mis en œuvre dans l'environnement de simulation réseau appelé OMNeT++ version 5.0 et nous y avons extrait quelques résultats qui nous ont permis d'asseoir notre stratégie. L'objectif de nos simulations était de faire une comparaison de la qualité de service du réseau en absence de pannes persistantes et en présence de celles-ci, afin de montrer l'avantage de notre stratégie de mise à jour des tables de routage dans l'amélioration de la QoS en présence de panne. Ces simulations ont été mené sur les mêmes réseaux du chapitre II pendant une durée de 30 secondes. Nous détaillons et interprétons dans la suite, les résultats numériques obtenus suivants les

1. Le dimensionnement d'un réseau consiste à déterminer les exigences de capacité minimale qui permettront toujours de répondre aux exigences de qualité de service du trafic. Pour ce faire, le dimensionnement implique de planifier le trafic aux heures de pointe, c'est-à-dire à cette heure de la journée pendant laquelle l'intensité du trafic est à son maximum (Doumith, 2007).

différents scénarios explorés dans le cas de panne persistante d'une liaison et celui de plusieurs liaisons.

III.5.1. Cas de panne d'une seule liaison

III.5.1.1. Délai de transit des paquets

Réseau1 (5 nœuds et 7 liaisons)

Pour le réseau1 (5 nœuds et 7 liaisons) de la figure 45, l'analyse des délais d'acheminement des paquets dans divers nœuds et dans différentes situations d'instabilité du réseau (figures 45a et 45b), nous a permis d'apprécier la variabilité de la QoS dans ce dernier.

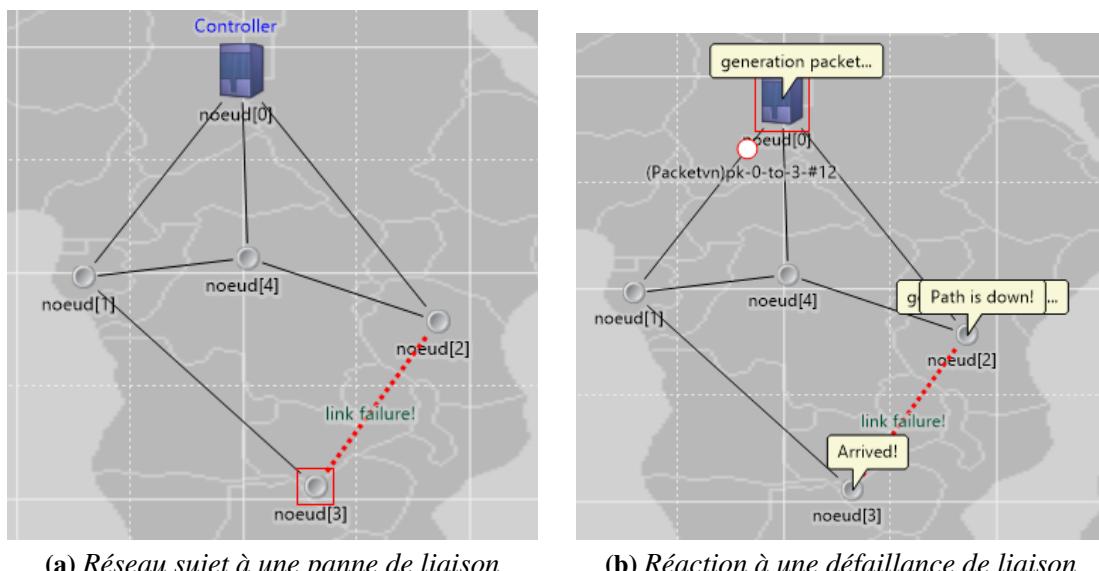


Figure 45 – Structure du réseau 1.

Les données ont été obtenu en exécutant successivement dans chacun des réseaux de test, les simulations dans les situations suivantes :

- en absence de toute panne ;
- en présence d'une panne passagère de liaison ;
- en présence de la persistance de la panne de liaison précédente.

En fixant la destination des paquets pour le nœud[3] (voir figure 45b), on obtient l'arbre nominal de la figure 46, à partir duquel le réseau exploite le nœud[4] comme nœud critique ; cela permet d'obtenir les délais d'acheminement (délais de transit) des paquets présenté à la figure 47. On peut d'ailleurs y constater que les délais d'acheminement moyen des paquets en présence de panne persistante sont plus bas en comparaison de ceux observés en présence de panne passagère, ce qui se

justifie par l'exploitation de notre nœud critique (le nœud[4]) par certains paquets. Les nœuds dont les tables de routage sont mis à jour sont : le nœud[2], le nœud[4] et le contrôleur.

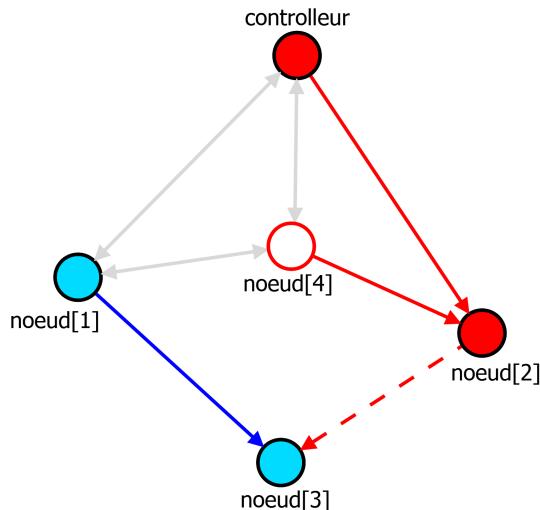


Figure 46 – Arbre nominal avec pour destination le nœud[3] dans le réseau1.

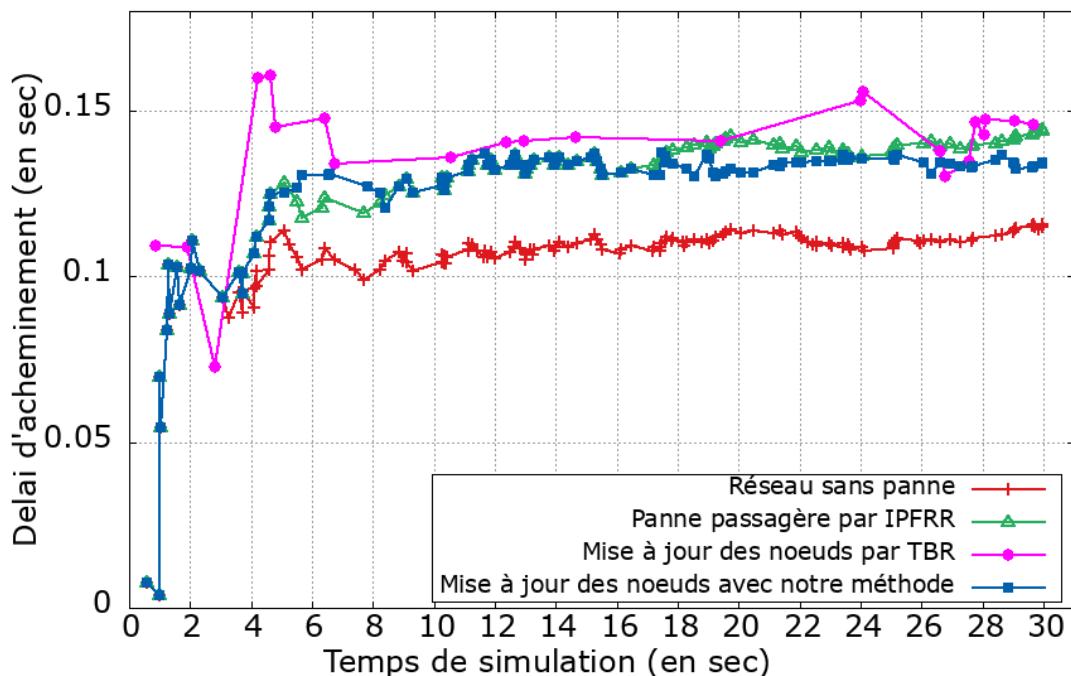


Figure 47 – Variations des délais de transit des paquets du nœud[3] dans le réseau1.

Par ailleurs, on peut constater d'après la figure 47 que notre approche de mise à jour de nœuds permet d'améliorer les temps de transit des paquets par rapport à celle proposée par le protocole TBR(Tree-Based Routing) ; cela se justifie certainement par le fait que la prise de décision locale basée sur les informations collectées

à partir des voisins dans le protocole TBR, ne permet pas toujours d'avoir des chemins suffisamment optimaux pour les nœuds à mettre à jour.

Autour du temps $t = 1.16985$, on observe une hausse brusque des délais d'acheminement des paquets, ce qui s'explique par le fait qu'une panne soit survenue à cet instant là ; cela nous permet de mesurer l'impact assez négatif des pannes sur la QoS dans le réseau, bien que cet impact tend à s'amenuiser dans la plupart des nœuds ordinaires sur l'ensemble des 30 secondes pendant lesquelles ont duré les simulations dans OMNeT++.

En étudiant la variation des délais de transit pour le nœud critique nœud[4] (voir graphe 48) et le nœud[1] (voir graphe 49) sans fixer de destination commune pour tous les paquets, ce qui se rapproche d'ailleurs le plus d'un réseau normal, on peut constater que les variations des délai d'acheminement sont proches les unes des autres pour le nœud critique ; cela s'explique certainement par le fait qu'il soit l'un des points de sortie du rerouting local mis en œuvre dans la technique IPFRR et permet donc l'interconnexion des deux parties G_r et G_b générées par une panne de liaison ; par contre, au niveau du nœud[1] qui est un nœud de G_b , cette variation est très importante entre l'état du réseau en l'absence de panne et son état en présence de panne. Cela est dû à la panne de liaison qui perturbe le trafic.

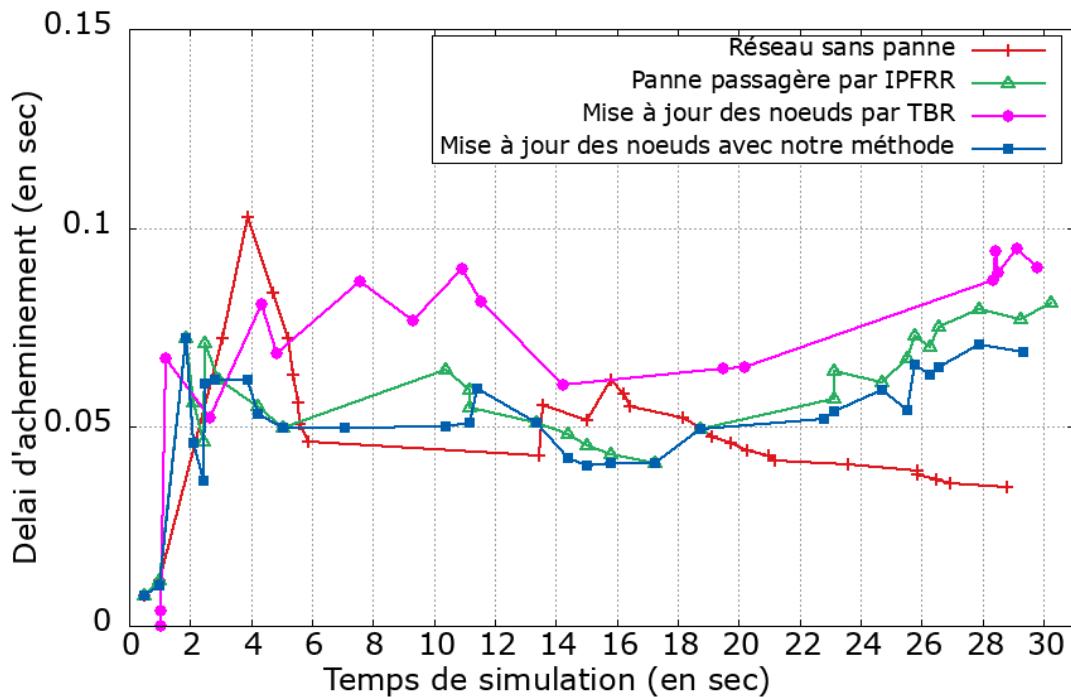


Figure 48 – Variations des délais d'acheminement des paquets du nœud[4] dans le réseau1.

Réseau2 (10 nœuds et 18 liaisons)

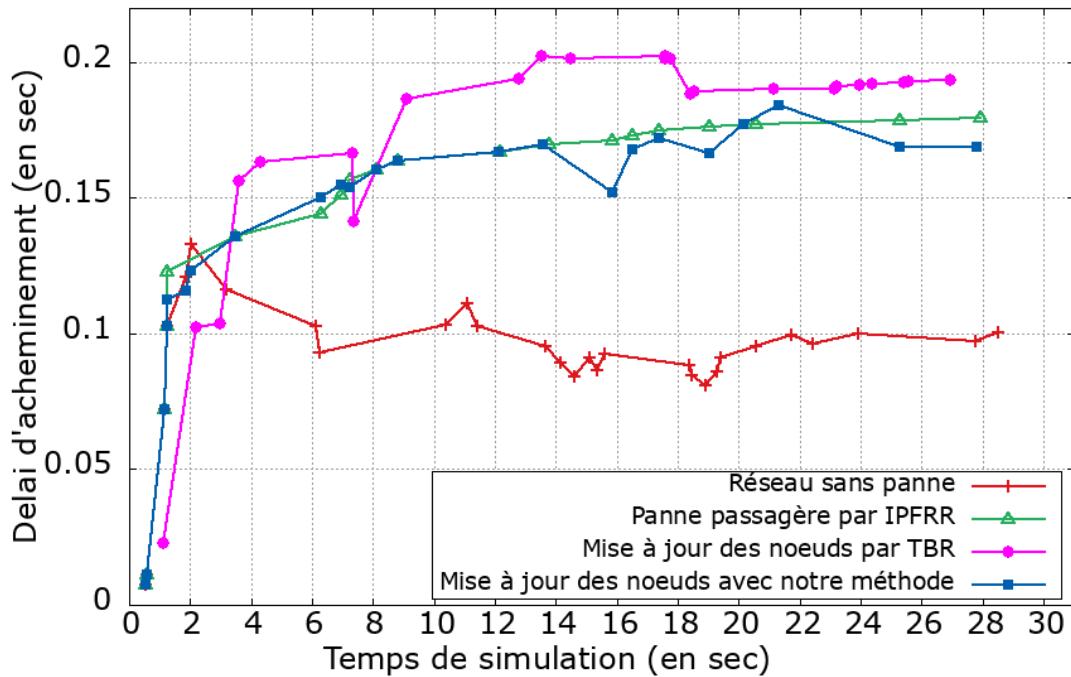


Figure 49 – Variations des délais d'acheminement des paquets du nœud[1] dans le réseau1.

En analysant les délais de transit des paquets pour le réseau2 sujet également à une panne de liaison, (voir figure 50), on obtient des résultats similaires à ceux du réseau1 (voir figure 51), notamment au niveau de la destination commune des paquets qui est ici le nœud[4]. L'observation précédente faite sur le nœud[3] dans le cas du réseau1 se vérifie également ici comme le montre la courbe de la figure 51 pour le nœud[4] du réseau2 ; ce qui laisse suggérer que notre stratégie de mise à jour améliore les délais de transit des paquets en présence de panne persistante, indépendamment de la taille du réseau.

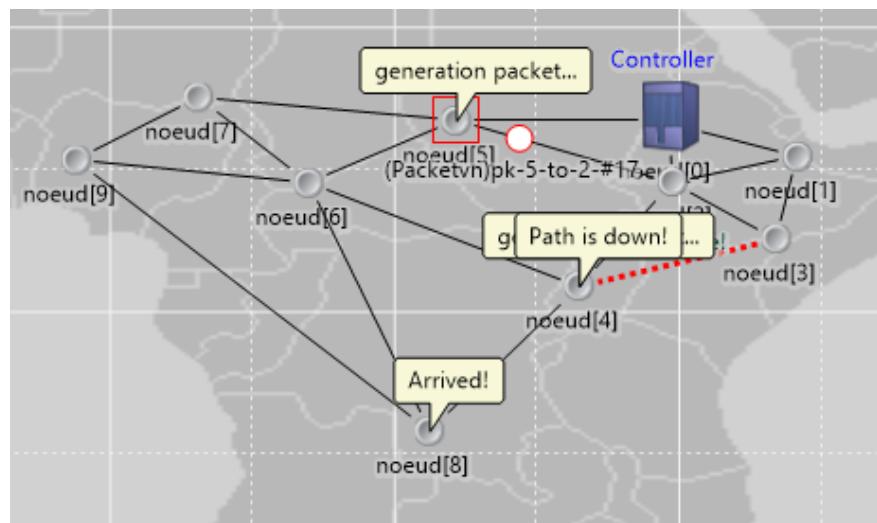


Figure 50 – Comportement du réseau2 en présence de panne persistante de liaison.

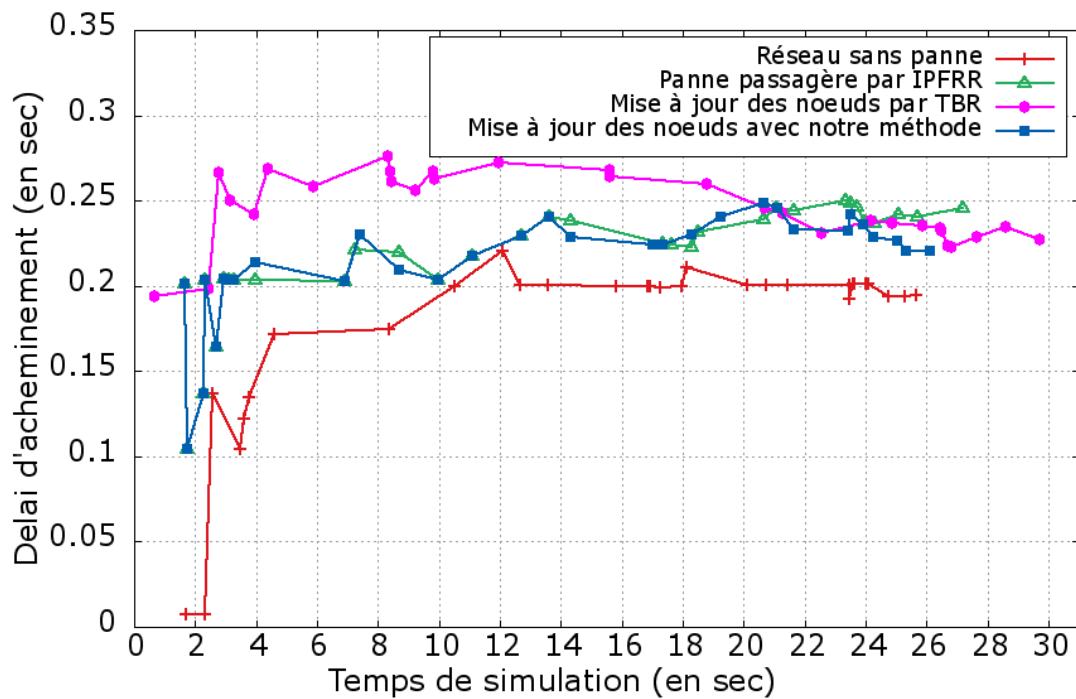


Figure 51 – Variations des délais de transit des paquets pour le nœud[4] du réseau2.

Le réseau3 présentant des caractéristiques proches du réseau2, on observera donc presque le même comportement que dans le réseau2. Pour raison de scalabilité, nous passons au réseau4 qui est bien plus grand.

Réseau4 (60 nœuds et 81 liaisons)

Des simulations sur des réseaux plus grands (60 nœuds et 81 liaisons) permettent de vérifier davantage les résultats précédents, tel que présenté à la figure 52. Les données de la figure 52 ont été obtenues en fixant le nœud[4] comme destination des paquets.

III.5.1.2. Analyse du taux de perte de paquets

L’analyse des taux de perte de paquets en ce qui concerne des nœuds stratégiques comme N_f (nœud ayant détecté la panne de liaison) ou le nœud destination de l’arbre de routage, permet d’obtenir le résultat de la figure 53. On peut y constater que notre approche réduit considérablement le taux de perte des paquets par rapport à celle de Son (2014). Le nœud 2 a été choisi parce qu’il est la racine de G_r qui mène directement à la liaison en panne et les paquets sont reroutés à travers ce dernier. Ce nœud 2 est donc celui le plus à même de perdre les paquets dans le réseau. Le nœud 3 quant à lui a été choisi parce qu’il est le nœud racine de l’arbre de routage nominal global pour l’exemple considéré. La simulation a été réalisée

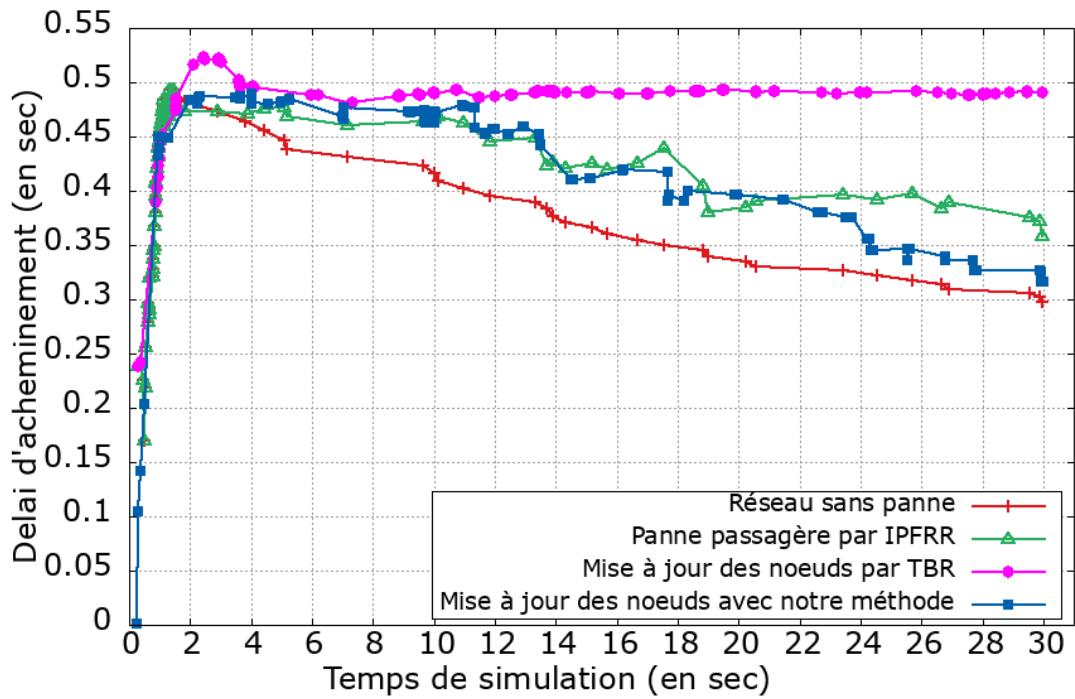


Figure 52 – Variations des délais de transit des paquets pour le nœud[4] du réseau4.

en présence de panne persistante de la liaison (nœud[3]-nœud[2]) présentée à la figure 46.

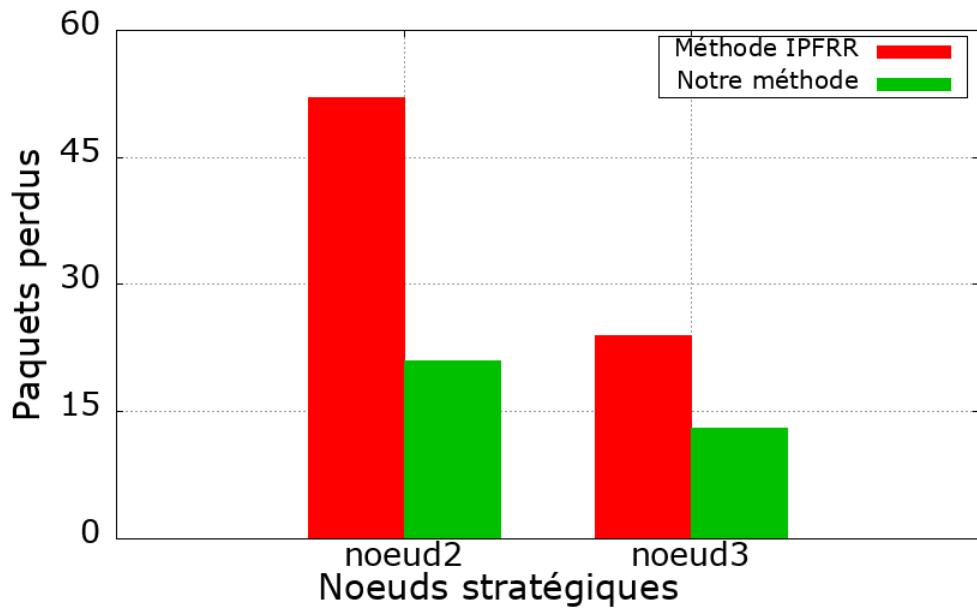


Figure 53 – Comparaison du taux de perte de paquets de notre stratégie de mise à jour des tables de routage avec celle de Son (2014) dans le réseau1.

III.5.2. Cas de panne persistante de plusieurs liaisons

Les simulations présentées ici ont pour but de montrer l'efficacité de notre approche à prendre en main aussi bien les pannes de liaisons multiples que les pannes d'une seule liaison présentées à la section III.3.3. Nos résultats seront comparés à ceux de l'approche TBR (Tree-Based Routing) et IPFRR (IP Fast ReRoute) de la littérature. Afin de conserver la conformité de nos tests avec ceux réalisés à la section III.5.1, nous conservons les mêmes réseaux de test. Les différentes topologies sont toujours générées aléatoirement, ceci dans le but de garantir l'aptitude de notre approche de mise à jour à fonctionner dans n'importe quelle topologie respectant nos hypothèses de départ.

III.5.2.1. Délai de transit des paquets

Réseau2 (10 nœuds et 18 liaisons)

Concernant le réseau2, la panne du nœud[3] correspond à la panne des liaisons (nœud[2]-nœud[3]), (nœud[4]-nœud[3]) et (nœud[1]-nœud[3]) comme présenté à la figure 54. Les nœud[2], nœud[4] et nœud[1] sont les nœuds stratégiques qui permettent de prendre en main la panne du nœud[3] à travers la détection des pannes de liaisons rattachées à ce nœud[3].

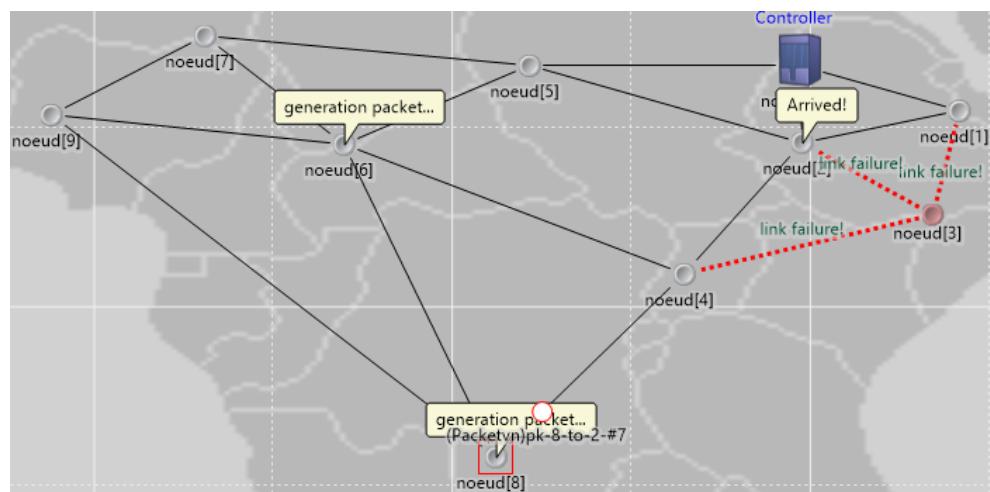


Figure 54 – Aspect du reseau2 en présence de la panne du nœud[3].

L'analyse des délais d'acheminement des paquets a permis d'obtenir le résultat présenté à la figure 55.

On peut constater sur cette figure que notre approche est meilleure que celle proposée par le protocole TBR (Tree-Based Routing) en termes de délai de transit des paquets. En effet, les délais de transit sont plus importants pour le TBR qu'avec notre approche. Cette grande différence pourrait s'expliquer par le fait que

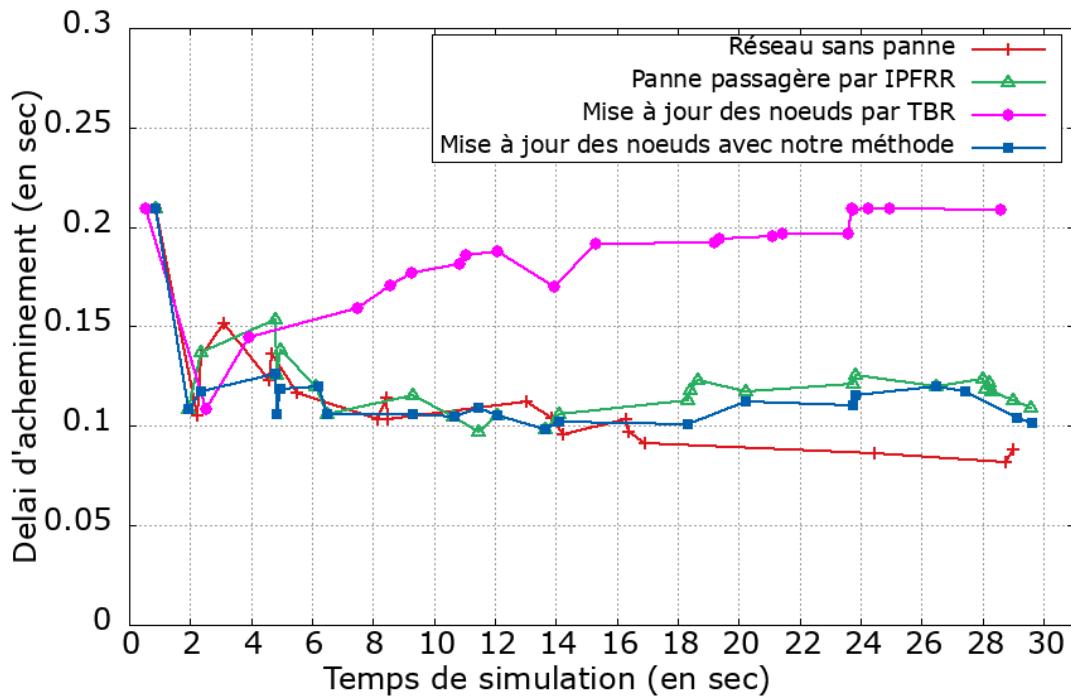


Figure 55 – Variations des délais d'acheminement des paquets du nœud[4] du reseau2 dans le cas de panne du nœud[3].

l'approche TBR se base sur les informations de voisins pour construire ses informations de mise à jour ; puisqu'il y a certains voisins dont les informations pourraient ne pas parvenir au nœud destinataire, les informations de mise à jour peuvent donc être erronées. Notre approche exploite effectivement l'ensemble des informations des nœuds voisins pour construire ses informations de mise à jour, ce qui permet d'avoir des chemins optimaux réduisant le délai de transit des paquets.

Par ailleurs, notre méthode de mise à jour permet d'offrir des délais de transit meilleurs que ceux de l'approche IPFRR de Son (2014). Celà est dû aux nœuds critiques que nous utilisons lors de la mise à jour, en plus de ceux présents dans le chemin de reroutage exploité pour traiter la panne temporaire de liaison. Ces observations montrent que les nœuds critiques que nous impliquons dans le processus de mise à jour sont très importants pour offrir une meilleure QoS face à une panne persistante de liaison. Les nœuds du chemin de reroutage ne doivent donc pas être les seuls à être mis à jour.

Les observations faites dans le cadre de ce réseau2, sont presque les mêmes pour le réseau3, car présentant quasiment les mêmes caractéristiques en terme d'étendue ; raison pour laquelle il serait plus intéressant de passer à une plus grande échelle représentée par le réseau4.

Réseau4 (60 nœuds et 81 liaisons)

Pour les réseaux de grande taille (comme le reseau4 avec 60 nœuds), la variation des délais de transit des paquets présentée à la figure 56 nous permet de constater que l'approche que nous proposons est encore meilleure que celle proposée par le protocole TBR. De plus, la variation des délais d'acheminement des paquets entre ces deux méthodes est assez importante. Cette grande différence peut s'expliquer par le nombre assez important de nœuds dans le réseau, ce qui contribue à renforcer la difficulté de prise de décision à partir des informations des voisins qui sont en nombre important.

On peut aussi constater que notre approche offre ici aussi de meilleurs délais d'acheminement. De plus, la différence de delais entre notre méthode et celle de Son (2014) est assez importante. Celà est encore dû à l'utilisation des nœuds critiques. Les résultats présentés ici montrent que ces nœuds critiques ont une grande influence sur les délais d'acheminement des paquets lorsque le réseau est de grande taille ; ceci justifie l'intérêt de notre recherche de ces nœuds critiques dans notre stratégie de mise à jour. Mais, la situation idéale reste celle dans laquelle le réseau ne subit aucune panne. En effet, en l'abscence de panne, les délais d'acheminements de paquets sont meilleurs.

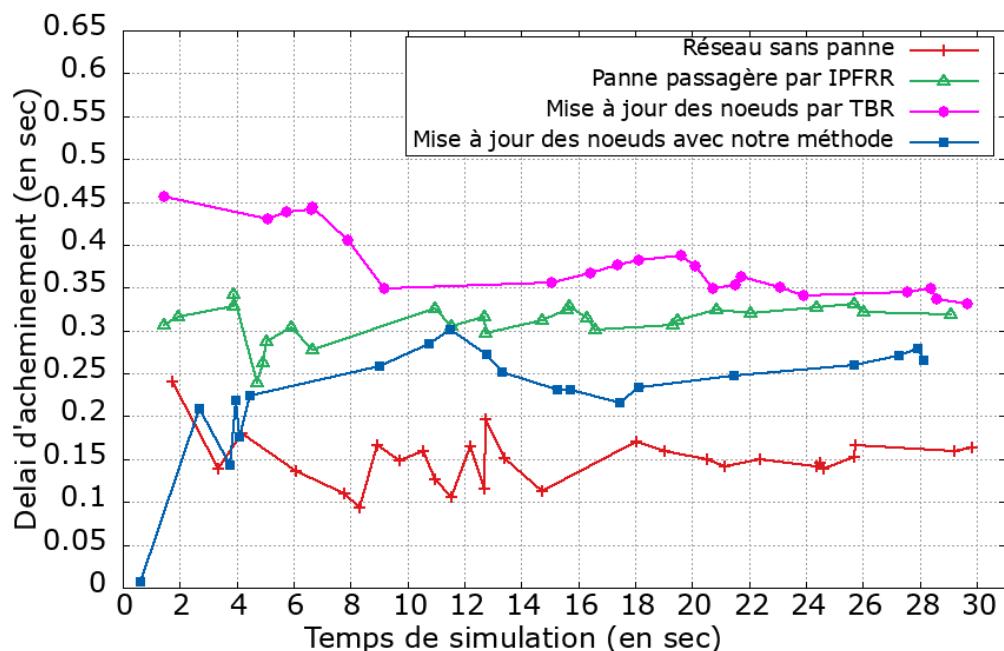


Figure 56 – Variations des délais d'acheminement des paquets du nœud[4] du reseau4 dans le cas de panne du nœud[3].

III.5.2.2. Taux de perte de paquets dans un réseau de grande taille

L'étude du taux de perte des paquets dans un réseau de grande taille tel que le reseau4 nous permet d'obtenir le résultat de la figure 57. Les paquets ont été gé-

néré aléatoirement et une moyenne de 5120 paquets émis dans le réseau⁴ ont été considéré sur l'ensemble des nœuds du réseau. On constate globalement une forte augmentation de l'ordre de 60%, du taux de perte de paquets par rapport à celui observé en l'absence de panne. Cette situation pourrait s'expliquer par l'insuffisance des ressources nécessaires pour rerouter les flux à un moment donné dans le réseau.

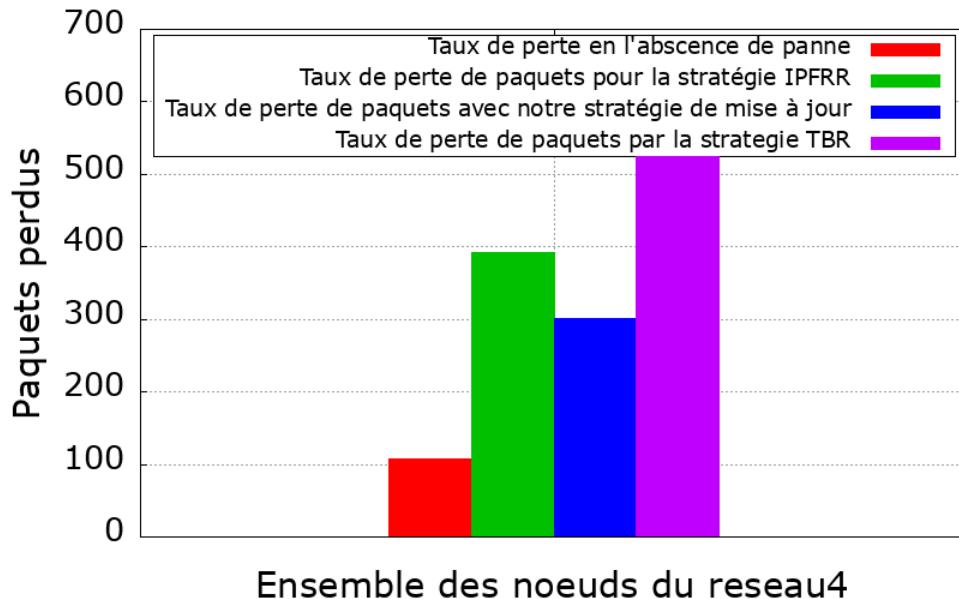


Figure 57 – *Taux de perte de paquets de notre stratégie de mise à jour des tables de routage vs la méthode de Son (2014) vs le protocole TBR pour le cas de panne de nœud dans le réseau4.*

Nous pouvons par ailleurs observer que notre approche de mise à jour ne permet certes pas de faire disparaître les pertes de paquets, mais contribue à les réduire de l'ordre de 24% (par rapport à la méthode IPFRR) à 43% (par rapport à la méthode TBR).

III.6. Conclusion

Il était question pour nous dans ce chapitre de proposer une solution au problème de mise à jour des tables de routage dans un environnement réseau à architecture centralisée, soumise aux pannes persistantes de liaisons et de nœuds sans nuire au trafic réseau. C'est ainsi que nous avons proposé une stratégie de mise à jour permettant une prise en main avisée d'un tel problème en ciblant judicieusement les nœuds à impliquer dans le processus, et en mettant en place une méthode de mise à jour de proche en proche des nœuds sur la base d'une structure de message de mise à jour (inspirée de RIP et OSPF) comportant les informations de

modification des tables de routage. Les conséquences immédiates de cette stratégie comme l'ont montré les simulations réalisées, sont la réduction des délais d'acheminement des paquets de bout en bout et les taux de perte des paquets dus aux conflits, tout en conservant autant que possible la structure du réseau, comparativement au reroutage automatique utilisé par IPFRR pour le cas de pannes passagères d'une seule liaison.

Cependant, la solution de mise à jour pour une panne persistante d'une seule liaison, ne peut pas être appliquée directement à une panne persistante d'un nœud sans ajouter une étiquette à la structure du message de mise à jour ; cette étiquette permet de faire face aux problèmes de synchronisation entre informations de mise à jour se rencontrant au sein d'un même nœud, et appartenant aux pannes de liaisons différentes. Les résultats de simulations révèlent également une amélioration du temps de transit des paquets ainsi qu'une réduction conséquente du taux de perte de paquets (de l'ordre de 25%) par rapport à la méthode IPFRR ou l'approche TBR (Tree-Based Routing) de la littérature. De ce fait, l'influence de l'approche de mise à jour des noeuds que nous avons proposé, minimise les pertes de paquets. Nous avons pu néanmoins constater une assez grande augmentation du taux de perte des paquets en présence d'une panne de nœud dans un réseau de grande taille ; ce qui se justifie par le degré élevé des nœuds dans ces types de réseaux, mais aussi la grande quantité de trafic qu'ils contrôlent et dont la surcharge peut causer la panne des nœuds concernés. Le chapitre suivant se propose d'apporter une solution à la gestion de ces excédents de trafic au sein des nœuds, qui nuisent à la QoS.

IV

CHAPITRE

GESTION DE CONGESTIONS PAR REPOSITIONNEMENT DYNAMIQUE DES SERVEURS DE SERVICES VIRTUELS

SOMMAIRE

IV.1 - Introduction	111
IV.2 - Méthode de Horiuchi et Tachibana complétée	112
IV.3 - Technique de séparation du trafic de migration	117
IV.4 - Simulations et discussion	127
IV.5 - Conclusion	135

IV.1. Introduction

Aux chapitres II et III, nous avons proposé des approches de rerouting qui redirigent les flux issus de la panne de liaison ou de nœud à travers des chemins alternatifs, vers une destination connue. Ce rerouting implique la prise en charge d'un trafic inattendu au sein des nœuds se trouvant sur les chemins de reroutages utilisés, avec un risque de conduire à la congestion au sein de ces nœuds. Dans ce cas, si ces nœuds hébergent des services, alors leur qualité sera détériorée. Le rerouting n'est qu'un exemple des causes de variation de trafic parmi lesquels on retrouve aussi la mobilité des utilisateurs et le changement de topologie. Nous focalisant sur la variation du trafic au delà de la spécificité des causes, nous proposons dans ce chapitre une solution à ce problème de congestion de trafic au sein d'un nœud offrant un service virtuel.

Notre solution est le repositionnement de service virtuel, qui consiste à faire migrer un service virtuel de son lieu de fourniture habituel vers un autre ayant les

ressources nécessaires pour satisfaire les besoins de QoS face au trafic courant. Pour ce faire, les questions de choix du nouveau nœud de réception du service virtuel et la technique de migration à utiliser, se posent. Nous proposons ici deux méthodes de repositionnement :

1. la première méthode vise à compléter l'approche de repositionnement de serveur de Horiuchi et Tachibana (2018), dont une description a été faite à la section I.7.2 du chapitre I, en ajoutant des éléments à l'algorithme utilisé. Cette méthode conserve la même structure arborescente ;
2. la deuxième méthode intègre des approches nouvelles de repositionnement, fonctionnant dans une topologie de graphe. Cette approche offre plus de chemins de repositionnement du service considéré.

Nous considérons tout au long de ce chapitre un seul serveur à repositionner dans un plan virtuel.

IV.2. Méthode de Horiuchi et Tachibana complétée

Dans cette section, nous proposons quelques solutions répondant aux problèmes de gestion de poids de trafic égal dans les nœuds feuilles de la topologie arborescente, du repositionnement automatique et du mécanisme de migration des données du service virtuel. Ces problèmes liés à la méthode de Horiuchi et Tachibana (2018) ont été présentés au chapitre I, section I.7.2.

IV.2.1. Traitement du cas de l'égalité des poids de flux

Au chapitre I section I.7.2, nous avions présenté l'approche de Horiuchi et Tachibana (2018), dans laquelle le repositionnement s'effectue sur la base d'une topologie sous forme d'arbre. L'idée consistait alors à faire migrer le serveur virtuel de son point originel vers une destination connue dans l'arbre, en remontant progressivement cet arbre (voir figure 20). A chaque niveau de l'arbre, le nœud présentant le plus faible poids de trafic courant est sélectionné. Nous avions alors ressorti quelques limites parmi lesquelles :

- le manque de solution lorsque les nœuds feuilles de l'arbre ont le même poids de trafic ;
- un nombre important de mobilité d'un service virtuel ;
- la non conservation d'un état cohérent du serveur une fois repositionné.

Pour intégrer l'égalité de poids de flux, nous sélectionnons au début de l'algorithme, le nœud feuille dont le nœud parent possède le plus grand poids de trafic. A ce niveau, deux scénarios sont envisageables :

1. les nœuds parents ont des poids égaux

on remonte dans l'arborescence jusqu'à trouver un niveau où les poids de trafic sont différents. Une fois ce niveau atteint, on sélectionne parmi les nœuds feuille du nœud ayant le plus gros poids de trafic, n'importe quel nœud feuille de ce nœud parent ; puis on ajoute le poids de trafic de ce nœud feuille à celui de son nœud parent. Ensuite, on élimine (dans le processus) le nœud feuille précédemment sélectionné et on recommence le processus jusqu'à ce qu'il n'y ait plus que deux nœuds dans l'arbre. Finalement, le nœud sélectionné pour abriter le serveur est celui qui a la plus haute charge de trafic. La figure 58 illustre cette piste de solution.

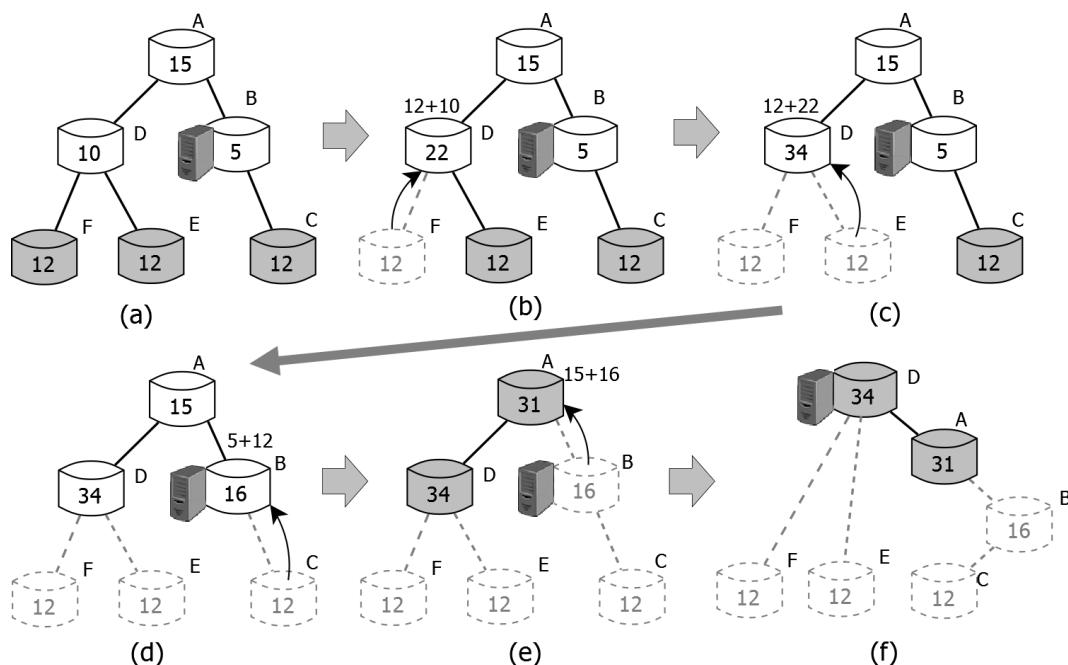


Figure 58 – Solution pour le problème de poids de trafic égaux dans les nœuds terminaux avec des parents de poids de trafic différents.

2. nous avons un arbre avec un nœud racine suivi directement par les nœuds feuilles (avec des poids de trafic égaux)

nous sélectionnons au hasard n'importe quel nœud feuille (voir figure 59). Plus généralement, si la hauteur de la topologie d'arbre est $h = 1$, alors n'importe quel nœud feuille peut être sélectionné.

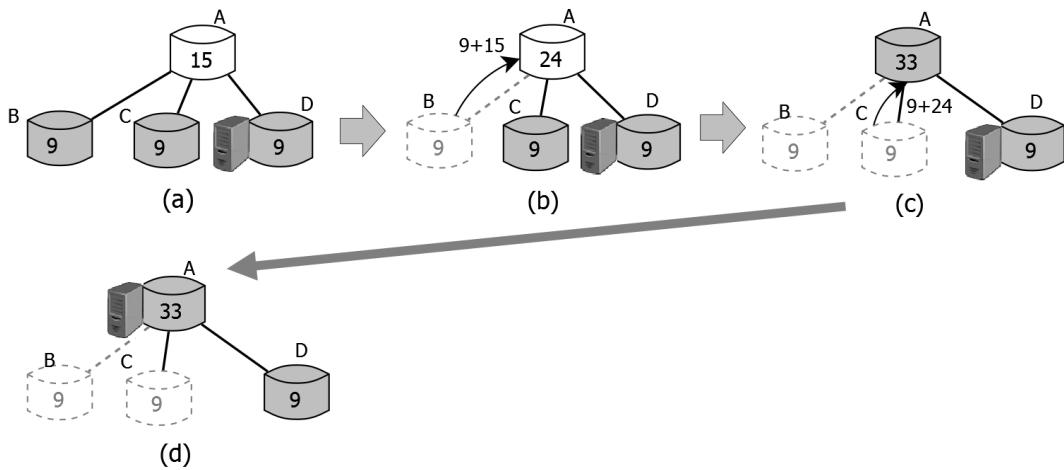


Figure 59 – Solution pour le problème du poids de trafic égaux dans les nœuds feuilles avec une hauteur d’arbre $h = 1$.

IV.2.2. Réduction des repositionnements

D’après Horiuchi et Tachibana (2018), lorsqu’une surcharge de trafic a lieu au niveau d’un nœud en terme de trafic, le processus de repositionnement du serveur est aussitôt enclenché ; ce processus de repositionnement qui a pour objectif d’améliorer la qualité de service, peut la détériorer davantage. Cette détérioration est due au temps nécessaire pour effectivement repositionner le serveur, et aux perturbations que cette opération engendre dans le réseau (Keshavamurthy & Guruprasad, 2015). Si plusieurs nœuds déclenchent ainsi un repositionnement, la QoS sera davantage menacée ; d’où la nécessité de mettre en place une astuce visant à ne recourir au repositionnement du serveur qu’en cas d’extrême nécessité. Ainsi, un délai peut être défini après le constat de surcharge, au-delà duquel on initie effectivement le processus de repositionnement.

Le cas d’extrême nécessité de repositionnement de serveur pourrait se définir à travers des critères dont nous mentionnons ici quelques-uns :

1. le type de service offert

En fonction des services offerts par les réseaux virtuels, le traitement qui leur ai réservé en termes de ressources, de politiques de gestion des flux, doit tenir compte de leur nature ainsi que de leur priorité les uns par rapport aux autres (Seddiki, 2015). Ainsi, en cas de surcharge du réseau, le temps de réaction immédiate est relatif à la nature du service en jeu. De plus, certains services tels que la VOIP, la VOD sont moins tolérants aux temps de latences que d’autres et nécessitent donc un temps de réaction plus ou moins rapide en situation de pannes ;

2. la disponibilité des ressources dans l’environnement virtuel considéré

Au moment où on a le plus besoin de repositionner un serveur, il peut arriver qu'on ne trouve pas un nœud qui soit capable de recevoir ce serveur ; dès lors, il faut attendre pour prendre effectivement en main le repositionnement dans une telle situation.

Le délai de réaction mis en place permettrait ainsi d'éviter des repositionnements inutiles dus à des surcharges de trafic temporaires. De façon pratique, le temps de réaction est défini par le fournisseur du service sur la base du type de service proposé. Mais, en considérant le repositionnement, un temps de migration aléatoire pourrait prolonger le temps d'indisponibilité de service.

IV.2.3. Mécanisme de migration des données

Il existe plusieurs mécanismes permettant d'assurer la migration d'entités virtuelles d'un nœud à un autre.

La migration à froid (ou arrêt et copie) ([Keshavamurthy & Guruprasad, 2015](#))

Dans cette approche, le service est d'abord arrêté, puis ses données sont copiées vers le nouveau nœud ; une fois la copie terminée dans le nœud de destination, le service y est remis en marche. L'avantage principal de cette méthode est qu'elle assure la migration sans faute de la mémoire du serveur. De plus, l'état de la mémoire du serveur n'est pas modifié sur l'hôte source. Par contre, les temps d'arrêts de service et de démarrage sur le nouveau nœud sont plus longs.

La migration à chaud (ou temps réel) ([Clark et al., 2005](#))

On distingue trois approches de migration à chaud : la pré-copie, la post-copie et la post-copie hybride.

- **La pré-copie** (voir figure 60) consiste principalement à transmettre l'ensemble des pages mémoires du serveur vers l'hôte de destination pendant que le service est en fonctionnement. Après un seuil de copie donné, le service est arrêté sur l'hôte source et le reste des pages mémoires modifiée est copié vers l'hôte de destination. Le problème majeur intervient lorsque la durée d'interruption maximale est trop faible pour pouvoir envoyer les dernières pages modifiées au serveur de destination ([Kherbache, 2016](#)). Dans ce cas la migration peut durer indéfiniment.
- **La post-copie** (voir figure 61) contrairement à la pré-copie, débute par l'arrêt immédiat de la machine virtuelle sur le serveur source. Ensuite, l'algorithme de post-copie transfère uniquement les données nécessaires au démarrage du service, avant d'activer le service sur l'hôte de destination.

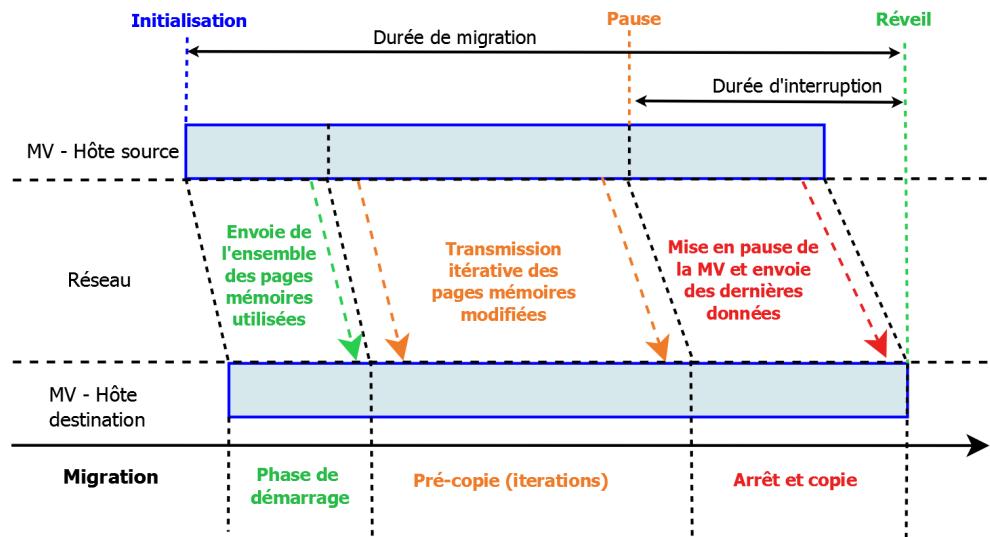


Figure 60 – Algorithme de pré-copie (Kherbache, 2016).

L'inconvénient majeur par rapport à la pré-copie repose sur la robustesse de l'approche (Kherbache, 2016). En effet, étant donné que la machine virtuelle est directement réveillée sur l'hôte de destination dans un état inconsistant (sans mémoire), la défaillance de l'un des deux hôtes, source ou destination en cours de migration, entraîne la perte inévitable de l'intégrité de l'état mémoire du serveur.

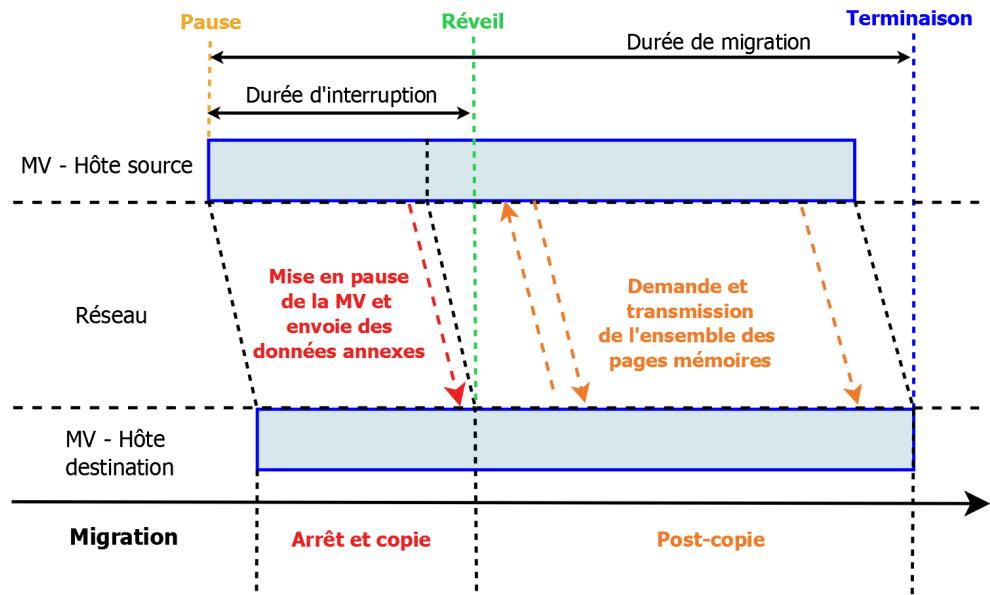


Figure 61 – Algorithme de post-copie (Kherbache, 2016).

- **La post-copie hybride** (Luo et al., 2008) a été proposé afin de réduire les problèmes de performances de la post-copie. Elle reprend les principes de la post-copie, en ajoutant une étape de pré-copie. La figure 62 présente l'algo-

rithme de post-copie hybride. L'algorithme exécute la première étape de pré-copie en transférant les pages mémoires utilisées par la machine virtuelle, avant de remettre en marche cette machine virtuelle sur l'hôte de destination. Ainsi, seules les pages mémoires modifiées depuis la mise en pause de la MV doivent être transférés en suivant les principes de la post-copie ; cela restreint la perte de performance occasionnée à la MV. La durée d'interruption reste donc minimale et identique à la post-copie classique, mais avec la conservation de la cohérence de la machine virtuelle ; ce qui en fait une bonne alternative pour les MVs présentant une forte activité mémoire. Le principal inconvénient de cette approche est que le temps d'interruption de service est plus prolongé.

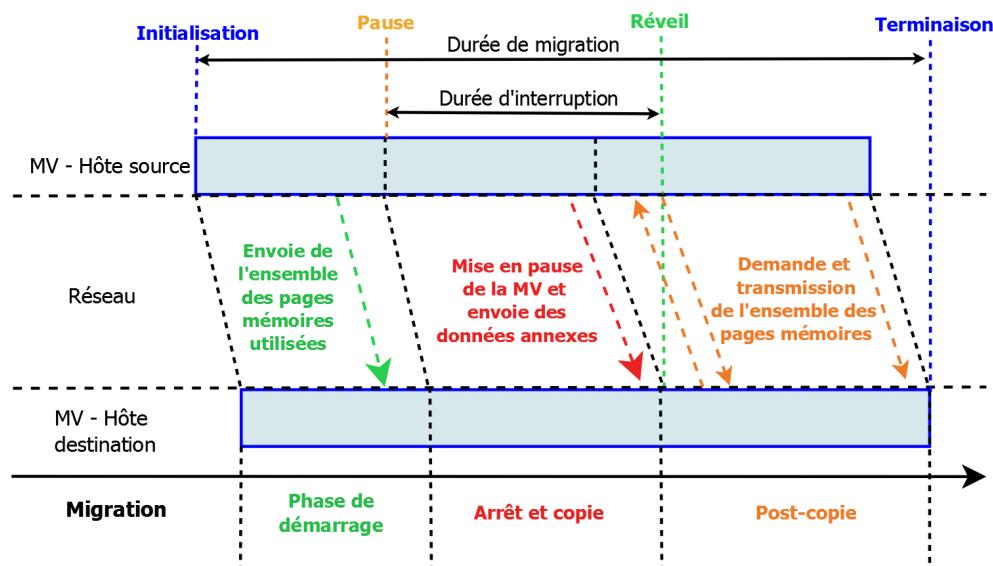


Figure 62 – Algorithme de post-copie hybride (Kherbache, 2016).

Pour assurer le repositionnement d'un service virtuel en conservant la cohérence du service une fois restauré dans le noeud de destination, nous adoptons la post-copie hybride. Nous résolvons la limite du temps d'interruption important souligné dans cette méthode en accélérant la migration des données du service virtuel par la séparation de flux. Les résultats obtenus au chapitre II justifient son utilisation dans notre stratégie de repositionnement de service virtuel.

IV.3. Technique de séparation du trafic de migration

Dans cette section, nous présentons le repositionnement de serveur virtuel par séparation de flux ; cette méthode de repositionnement transcende celle proposée par [Horiuchi et Tachibana \(2018\)](#) en offrant plus de chemins de repositionnement

tout en intégrant une technique de migration de données. Les objectifs visés sont la réduction des temps de migration et d'indisponibilité du service virtuel, qui sont des critères importants de QoS.

La grosse difficulté de l'utilisation de la méthode de séparation de flux dans un tel contexte réside au niveau du réassemblage des paquets au nœud destination. En effet, à cause de l'hétérogénéité des liaisons, les paquets n'arrivent pas toujours dans l'ordre d'émission et posent donc des difficultés dans la reconstruction du flux original. Une solution à ce problème dans la littérature est de limiter le nombre de séparations inutiles et d'étiqueter les paquets d'un même flux de façon à pouvoir les réassembler facilement sans perte ([Veerassamy et al., 1994](#)).

Une autre difficulté majeure c'est le choix des liaisons devant transporter les flux. Le nombre de ces liaisons détermine le nombre de petits flux qui seront générés. Il est question ici de fixer les règles qui vont régir le choix judicieux d'une liaison pour le transport des flux. Plusieurs critères de choix peuvent être envisagés :

- **la longueur de la liaison** : on peut dans ce cas choisir les chemins qui offrent la distance minimum entre la source et la destination. Mais, on peut avoir un chemin de distance minimale qui offre une bande passante très faible. La conséquence est le délai d'acheminement élevé, ce qui n'est pas acceptable pour assurer la QoS ;
- **la bande passante** : il est question de considérer que la bande passante suffit pour garantir des chemins compatibles avec la QoS. Toutefois le nombre de nœuds impliqués dans un chemin offrant une bonne bande passante peut aussi poser problème. En effet, dans le cadre du repositionnement de service ou de migration de machines virtuelles, à chaque nœud intermédiaire entre le nœud source et le nœud destination, il y'a un temps de latence qui est observé. Ce temps de latence est multiplié par le nombre de ces nœuds intermédiaires ([Liu et al., 2014](#)) ;
- **le débit** : après avoir trouvé un chemin offrant une bonne bande passante, le débit sur ce chemin peut-être faible et ne pas permettre de repositionner rapidement le service.

Compte tenu de ces difficultés, nous proposons la méthode de repositionnement baptisée FSB-DReViSeR qui se décline en deux versions : FSB-DReViSeR-bandwidth qui est une approche centrée sur la bande passante ; et FSB-DReViSeR-throughput centrée sur le débit.

IV.3.1. Repositionnement de serveurs par séparation de flux centrée sur la bande passante : FSB-DReViSeR-bandwidth

IV.3.1.1. Hypothèses et notations

Avant de décrire notre solution, considérons les hypothèses suivantes :

- le réseau a une topologie de graphe ;
- les liaisons sont bidirectionnelles et entre toute paire de nœuds, il existe au moins deux chemins disjoints permettant de les relier ;
- à chaque liaison, est associée un poids représentant la bande passante, la longueur de la liaison ou le débit réel sur la liaison ;
- le temps d'interruption lors de la copie des fichiers pendant la migration du service est la même pour tous les nœuds.

Dans la suite, nous considérons les notations du tableau VII.

Symbol	Description
n_i	noeud numéro i
P_i	chemin de numéro i
l_{ik}	liaison entre le nœud n_i et n_k
d_{ik}	temps de migration du nœud n_i à n_k
bp_i	bande passante de la liaison numéro i
Bp_i	bande passante du chemin numéro i
w_{ik}	longueur de la liaison l_{ik}
TM_i	temps de migration total sur la liaison l_{ik}

Table VII – Différentes notations utilisées dans FSB-DReViSeR.

IV.3.1.2. Description de la méthode

L'idée est de subdiviser le flux original en plusieurs sous-flux, acheminés simultanément à travers plusieurs chemins jusqu'au nouveau nœud qui hébergera le service. Tous les sous-flux ont la même destination. Seul le nœud hébergeant initialement le service à migrer, peut appliquer une séparation, ceci afin de réduire le nombre de subdivisions. Celà signifie que le flux de service à migrer n'est subdivisé qu'une seule fois tout au long du processus de migration jusqu'au nouveau hôte.

Notre objectif étant d'assurer un repositionnement rapide du service virtuel, le trafic de migration sera donc acheminé à travers les plus courts chemins du nœud source vers la destination. Ici, les critères qui régissent le choix des meilleurs chemins de migration sont la bande passante, la longueur des liaisons et le nombre de nœuds intermédiaires entre la source et la destination. Cela permet de s'affranchir

des cas de sélection erronées de certains chemins de migration sur la seule base de la bande passante, comme c'est par exemple le cas dans (Liu et al., 2014).

En effet, dans le contexte de migration de machines virtuelles, la méthode de migration unicemin de Liu et al. (2014) qui exploite le seul critère de bande passante pour la sélection du chemin de migration, souffre des limites suivantes :

- 1. Mauvais choix de chemin de migration, lorsqu'il en existe plusieurs de bande passante maximale**

Considérons le cas de la figure 63 dans laquelle nous voulons déplacer une machine virtuelle de n_1 à n_8 . Trois chemins peuvent être utilisés : chemin 1 avec 9 Gbit/s, chemin 2 avec 8 Gbit/s et chemin 3 avec 8 Gbit/s de bande passante. La méthode de Liu et al. (2014) sélectionnera n'importe lequel des chemin 1 ou 3 comme chemin de migration le plus approprié, car fournissant la plus grande bande passante disponible. Néanmoins, en regardant le nombre de sauts, le chemin 3 est meilleur que le chemin 1.

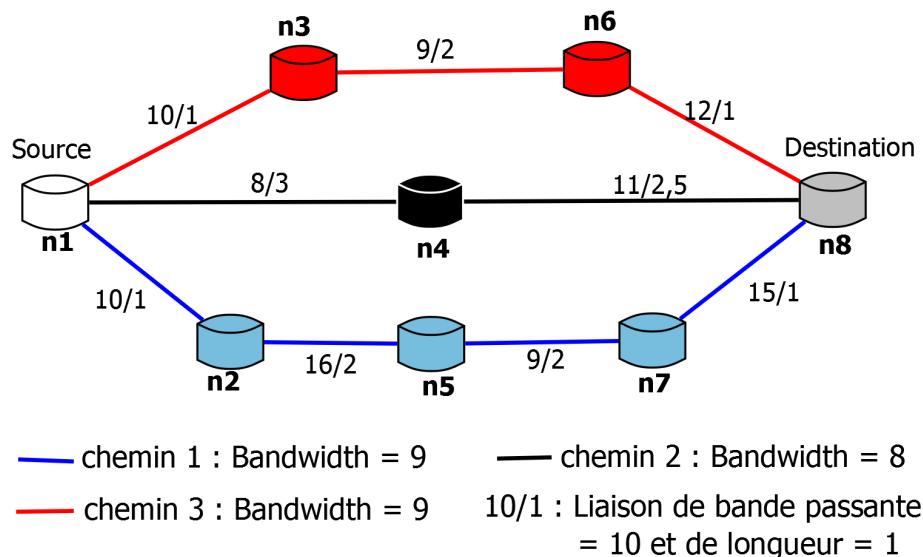


Figure 63 – Mauvais choix de chemin avec (Liu et al., 2014) dans le cas de liaisons avec bandes passantes égales.

- 2. décision de chemin de migration non valide dans un graphe avec chemins de bandes passantes différentes**

En supposant le cas de la figure 64, la méthode de Liu et al. (2014) sélectionnera le chemin 1 comme le plus approprié pour la migration de la machine virtuelle ; cela est dû au fait qu'en déterminant de façon comparée la bande passante du chemin 1 avec les trois autres ($5 > 4 > 3 > 2$), nous trouvons que celle du chemin 1 est la plus grande. Mais, en se concentrant sur le nombre

de sauts et la distance des liens dans les différents chemins, on déduit les résultats suivants :

- chemin 1 (bande passante = 5Gbits/s) : nombre de sauts = 5 et distance globale = $2 + 1 + 1 + 1 + 2.5 = 7.5$;
- chemin 2 (bande passante = 2Gbits/s) : nombre de sauts = 3 et distance globale = $2.5 + 3 + 2.5 = 8$;
- chemin 3 (bande passante = 3Gbits/s) : nombre de sauts = 5 et distance globale = $1 + 2 + 1 + 1 + 1 = 6$;
- chemin 4 (bande passante = 4Gbits/s) : nombre de sauts = 4 et distance globale = $2 + 1 + 1 + 2 = 6$.

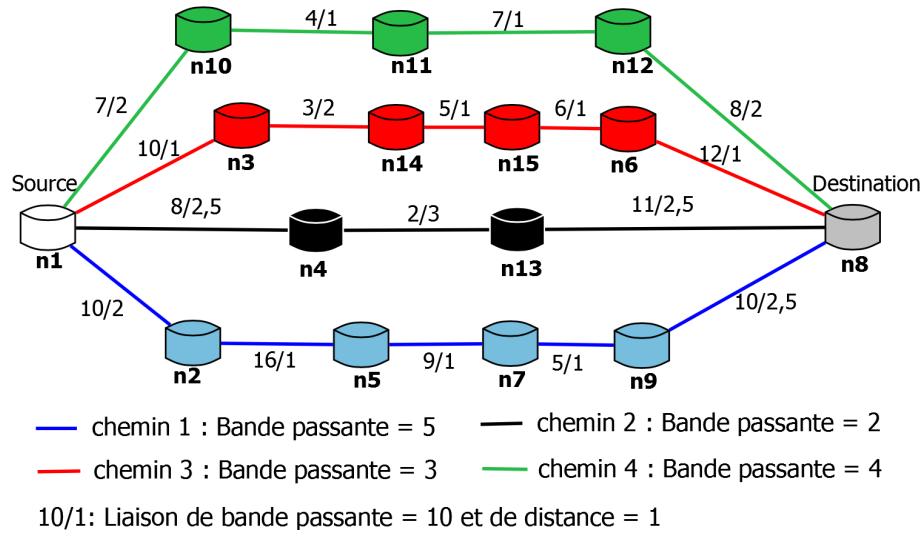


Figure 64 – Mauvais choix de chemin avec (Liu et al., 2014) dans le cas de liaisons avec bandes passantes différentes.

Malgré le fait que le chemin 1 offre la plus grande bande passante, les chemins 2, chemin 3 et chemin 4 pourraient être les meilleurs chemins ; ainsi, pris comme critère unique, la bande passante n'est pas le critère le plus important pour un bon calcul du chemin de migration.

En considérant ainsi plusieurs critères, nous pouvons faire des choix de chemins encore plus judicieux que (Liu et al., 2014).

Avant de choisir les chemins des sous-flux, nous estimons le temps de migration total de chacun d'eux. Pour cela, nous considérons que le temps de migration sur une unité de distance est donnée par la formule IV.1 issue de (Liu et al., 2014), où d_{mig} est le temps de migration et évolue linéairement par rapport à la quantité de mémoire transmise m_{mig} et à la bande passante b_p disponible pour la migration. Le temps d'interruption i_{mig} de la copie des fichiers de la machine virtuelle, généralement très faible, a très peu d'impact sur le temps de migration.

$$\frac{m_{mig}}{b_p} + i_{mig} \quad (\text{IV.1})$$

Ainsi, pour le lien l_{ik} , le temps de migration est donné par l'équation IV.2.

$$d_{ik} = w_{ik} \cdot d_{mig} \quad (\text{IV.2})$$

Pour tout chemin P_i du graphe, la durée de migration totale TM_i est donnée par IV.3.

$$TM_i = \sum_{k=1}^{n-1} d_{k(k+1)} \quad (\text{IV.3})$$

Le meilleur chemin de migration est celui dont le temps de migration est minimal : $\text{Min}(TM_i)$, ($1 \leq i \leq n$), où n est le nombre de chemins entre la source et la destination. La figure 65 illustre cette situation pour le cas d'une mauvaise sélection de chemin de migration de machine virtuelle rencontrée dans (Liu et al., 2014) lorsque les bandes passantes sont égales.

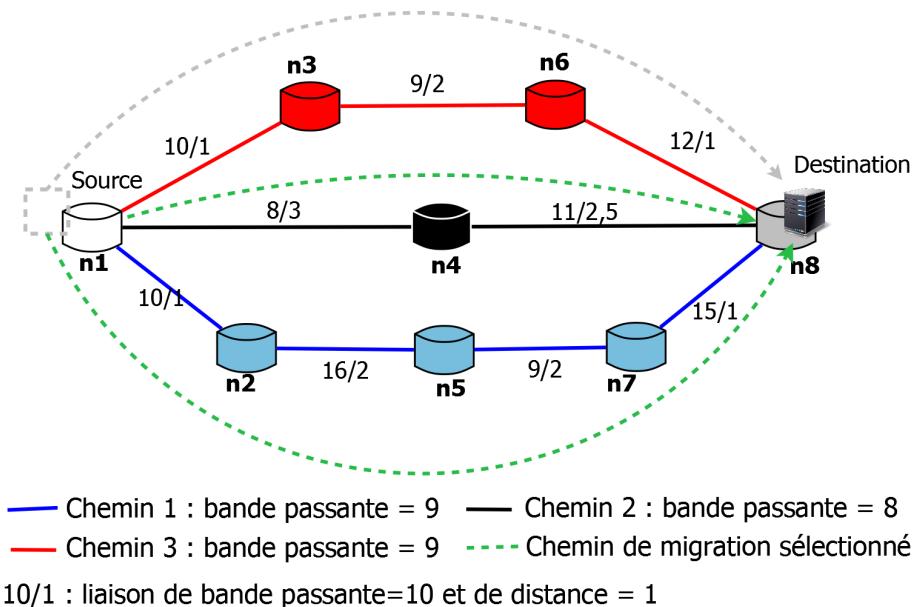


Figure 65 – Sélection des chemins de migration par séparation de flux dans le cas de chemins avec bandes passantes égales.

En considérant que le temps de transfert de la mémoire est $m_{mig} = 15\text{min}$ et le temps d'interruption est $i_{mig} = 1\text{min}$, en calculant le temps global de repositionnement, nous obtenons les valeurs suivantes :

- chemin 1 : bp=9Gbps et $d_{mig}=15/9 + 1 = 1.7 + 1 = 2.7 \text{ min}$. $d_{n1n2} = 1 * 2.7 = 2.7 \text{ min}$; $d_{n2n5} = 2 * 2.7 = 5.4 \text{ min}$; $d_{n5n7} = 2 * 2.7 = 5.4 \text{ min}$; $d_{n7n8} = 1 * 15/9 + 1 = 1.7 + 1 = 2.7 \text{ min}$.

- $2.7 = 2.7$ min ; la durée totale de migration pour le chemin 1 est : $TM_1 = 2 * 2.7 + 2 * 5.4 = 5.4 + 10.8 = 16.2$ min ;
- chemin 2 : $bp=8\text{Gbps}$ et $d_{mig}=15/8 + 1 = 1.9 + 1 = 2.9$ min. $TM_2 = 3 * 2.9 + 2.5 * 2.9 = 8.7 + 7.25 = 15.95$ min ;
 - chemin 3 : $bp=9\text{Gbps}$ and $d_{mig}=15/9 + 1 = 1.7 + 1 = 2.7$ min. $TM_3 = 1 * 2.7 + 2 * 2.7 + 1 * 2.7 = 10.8$ min.

D'après ces calculs, nous constatons que $TM_3 < TM_2 < TM_1$; ce qui signifie que dans le cas d'une séparation de flux en deux parties, les meilleurs chemins de repositionnement sont le chemin 3 et le chemin 2, car fournissant le temps de repositionnement le plus court. Dans le cas d'une approche de migration à chemin unique, le chemin 3 serait sélectionné comme le meilleur chemin de migration, qui est inférieur à celui de Liu et al. (2014) (ils considéraient également le chemin 1 comme le meilleur chemin en raison de la taille de bande passante égale).

En ce qui concerne la deuxième limite de Liu et al. (2014) liée à la sélection de chemin de migration non valide lorsque les largeurs de bandes passantes de chemins sont différentes, la figure 66 illustre comment notre méthode de sélection de chemin de migration y apporte une solution.

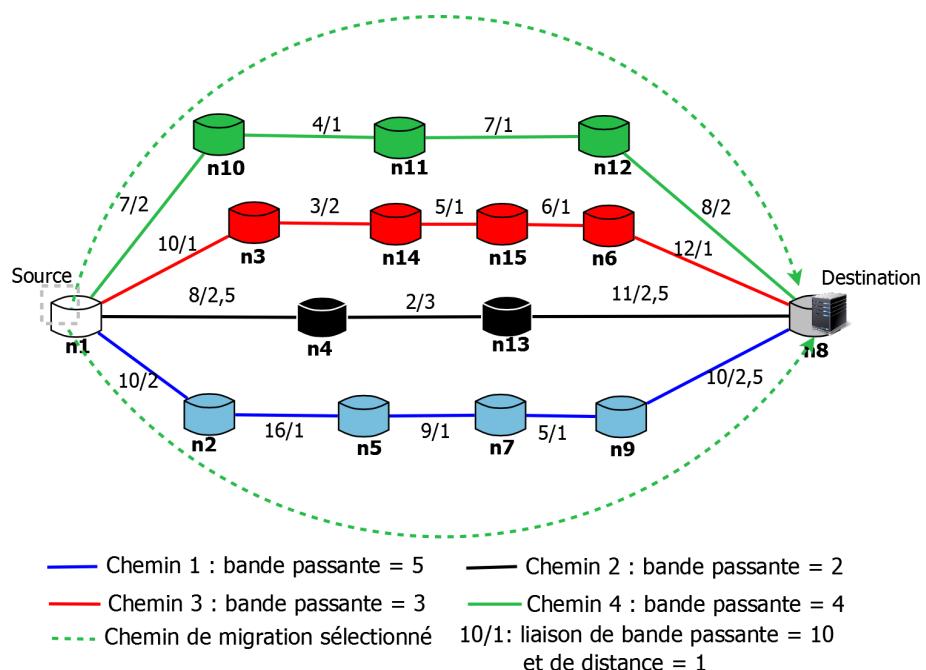


Figure 66 – Sélection des chemins de migration par séparation de flux dans le cas de chemins avec bandes passantes différentes.

Lorsque nous estimons le temps de repositionnement potentiel via chaque chemin, les valeurs suivantes sont obtenues :

- chemin 1 : $bp = 5 \text{ Gbps}$ et $d_{mig} = 15/5 + 1 = 3 + 1 = 4 \text{ min. le temps de migration total pour le chemin 1 est : } TM_1 = 2 * 4 + 1 * 4 + 1 * 4 + 2.5 * 4 = 30 \text{ min ;}$
- chemin 2 : $bp = 2\text{Gbps}$ et $d_{mig} = 15/2 + 1 = 7.5 + 1 = 8.5 \text{ min. } TM_2 = 8.5 * 2.5 + 8.5 * 3 + 8.5 * 2.5 = 68 \text{ min ;}$
- chemin 3 : $bp = 3\text{Gbps}$ et $d_{mig} = 15/3 + 1 = 5 + 1 = 6 \text{ min. } TM_3 = 1 * 6 + 2 * 6 + 1 * 6 + 1 * 6 = 36 \text{ min.}$
- chemin 4 : $bp = 4\text{Gbps}$ et $d_{mig} = 15/4 + 1 = 3.75 + 1 = 4.75 \text{ min. } TM_4 = 2 * 4.75 + 1 * 4.75 + 2 * 4.75 = 28.5 \text{ min.}$

Ces valeurs suggèrent que les meilleurs chemins de repositionnement potentiels sont le chemin 1 et le chemin 4, qui sont meilleurs que le chemin unique 1 sélectionné par [Liu et al. \(2014\)](#).

Par ailleurs, une fois les meilleurs chemins de repositionnement choisis, nous utilisons la méthode de post-copie hybride utilisée dans la migration des machines virtuelles ([Luo et al., 2008](#)) pour assurer la migration des fichiers du service. En effet, cette stratégie de migration permettra de conserver un état de service cohérent lors de sa restauration dans l'hôte de destination. Durant la phase de pré-copie, nous transférons graduellement les pages mémoires modifiées (contenant les données du service, le trafic courant, l'état d'exécution des requêtes,...) sur les plus courts chemins précédemment sélectionnés en direction du nouvel hôte.

IV.3.2. Repositionnement de serveurs par séparation de flux centré sur le débit des liaisons : FSB-DReViSeR-throughput

Dans la section [IV.3.1](#), nous avons proposé une méthode de repositionnement de serveur virtuel basée sur la séparation des flux en utilisant la bande passante et la longueur des liens comme critères de sélection de chemin de migration. Néanmoins, en pratique, les flux de données sur les liaisons ne sont pas constants comme la bande passante ; ainsi, certains chemins peuvent être sélectionnés comme les plus courts en fonction de la bande passante, tandis que le débit réel de données sur les liaisons correspondantes ne convient pas à la migration des données du service. Cette section présente notre solution basée sur le débit et la longueur des liens comme critères de sélection des chemins les plus courts pour le repositionnement du service.

IV.3.2.1. Difficulté du choix de chemins de migration sur la base du débit

La principale difficulté liée à la considération du débit comme critère de choix des chemins migration de service, est sa dynamicité. Cette dynamicité est spécifique à chaque lien et évolue avec le trafic réseau courant. Cette situation implique plusieurs défis à relever :

- comment connaître à chaque fois, le débit réel des liaisons en fonction du trafic ? Cette question induit d'avoir une carte du trafic réseau. Les serveurs virtuels n'ont pas cette capacité ; c'est pourquoi le SDN grâce à sa vue globale du réseau, est la solution appropriée pour relever ce défi ;
- comment faire un choix fiable des chemins de migration sur la base du débit, qui garantira que les flux soient livrés dans un délai acceptable ? En effet, un meilleur chemin à un instant t peut ne plus être un meilleur chemin à un instant $t + 1$ en raison de la variabilité du débit. Pour résoudre ce problème, nous utilisons la moyenne de débit sur chaque lien. Cela permet de prendre en compte les principales variations de débit.

IV.3.2.2. Description de la méthode d'exploitation du débit

Considérant la question de carte du trafic réseau, le contrôleur SDN grâce à sa vue globale du réseau (topologie, pannes, trafic ...), est la solution. En effet, grâce à l'activité de comptabilité du réseau dans le contrôleur, il est possible d'obtenir cette carte du trafic. La comptabilité de réseau est une activité qui permet à un fournisseur d'infrastructure, de surveiller l'activité au sein du réseau afin de détecter des comportements suspects ou extraire des informations utiles à la rentabilité de son réseau (Keller, Lee, & Rexford, 2009). Cette comptabilité permet d'incorporer de la sécurité et de la crédibilité (Mekouar, Iraqi, & Boutaba, 2010) dans le réseau.

Dans notre solution, lorsqu'un repositionnement de serveur est requis, le nœud source transmet un message au contrôleur pour qu'il établisse la carte du trafic réseau. Cette carte sera par la suite utilisée pour déterminer les meilleurs chemins de repositionnement. Grâce à la carte du trafic réseau, le contrôleur estime les variations de débit v_{min} et v_{max} de chaque liaison sur la base des observations réalisées pendant une période δt . Ensuite, le contrôleur calcule les plus courts chemins en considérant v_{min} d'une part, puis v_{max} d'autre part ; ceci signifie que pour une liaison l_{ik} donnée, les temps minimum et maximum de repositionnement sur ces liaisons sont respectivement données par les équations IV.4 et IV.5 où $i_{(mig)ik}$ est le temps de latence dans les nœuds intermédiaires entre les nœuds n_i et n_k .

$$d_{ik(min)} = \frac{m_{mig}}{v_{min}} + i_{(mig)ik} \quad (\text{IV.4})$$

$$d_{ik(max)} = \frac{m_{mig}}{v_{max}} + i_{(mig)ik} \quad (\text{IV.5})$$

Les équations IV.3, IV.4 et IV.5 permettent de déduire que pour un chemin P_i , les temps minimum et maximum de repositionnement sont respectivement données par les relations IV.6 et IV.7, avec $d_{k(k+1)min} = \frac{m_{mig}}{\text{Min}_{j=1}^n v_{(min)j}} + i_{(mig)ik}$.

$$TM_{i(min)} = \sum_{k=1}^{n-1} d_{k(k+1)min} \quad (\text{IV.6})$$

$$TM_{i(max)} = \sum_{k=1}^{n-1} d_{k(k+1)max} \quad (\text{IV.7})$$

L'expression $\text{Min}_{j=1}^n v_{(min)j}$ représente le fait que le débit sur un chemin P_i est le minimum des flux de données de l'ensemble des liaisons composant le chemin. Cette valeur est donnée par le débit sur la liaison de bande passante minimale.

Une méthode de sélection de chemin de repositionnement basée sur $TM_{i(min)}$ ne prendrait pas en compte les débit plus élevés. Par ailleurs, une méthode de sélection basée sur $TM_{i(max)}$ pousserait à envoyer sur certaines liaisons des flux de données supérieures à leur capacité autorisée. Dès lors, nous utilisons le débit moyen sur l'ensemble des liaisons de chaque chemin, comme un des critères de sélection des chemins de repositionnement. Le débit moyen d'une liaison entre deux nœuds n_i et n_k est donné par l'équation IV.8 :

$$d_{ik} = \frac{d_{ik(min)} + d_{ik(max)}}{2} \quad (\text{IV.8})$$

Cette moyenne devrait réduire les cas de débit de données non pris en compte. La figure 67 montre cette utilisation du débit comme l'un des critères de sélection des chemins de migration au lieu de la bande passante. Dans cette figure, le débit moyen de chaque chemin est pris en compte. Les différents temps de repositionnement sont calculés comme suit :

- chemin 1 : bp=9Gbps et $d_{mig}=15/7 + 1 = 2.14 + 1 = 3.14$ min. $d_{n1n2} = 1 * 3.14 = 3.14$ min; $d_{n2n5} = 2 * 3.14 = 6.28$ min; $d_{n5n7} = 2 * 3.14 = 6.28$ min; $d_{n7n8} = 1 * 3.14 = 3.14$ min; le temps de migration total du chemin 1 est : $TM_1 = 2 * 6.28 + 2 * 3.14 = 18.84$ min;
- chemin 2 : bp=8Gbps et $d_{mig}=15/6 + 1 = 2.5 + 1 = 3.5$ min. $TM_2 = 3 * 3.5 + 2.5 * 3.5 = 10.5 + 8.75 = 19.25$ min;

- chemin 3 : $bp=9\text{Gbps}$ et $d_{mig}=15/4 + 1 = 3.75 + 1 = 4.75 \text{ min}$. $TM_3 = 1 * 4.75 + 2 * 4.75 + 1 * 4.75 = 12.56 \text{ min}$.

Ainsi dans le cas de la séparation de flux en deux parties présentée à la figure 67, les meilleurs chemins de repositionnement sont le chemin 3 et le chemin 1, au lieu du chemin 3 et du chemin 2 sélectionnés lorsque la bande passante est utilisée plutôt que le débit.

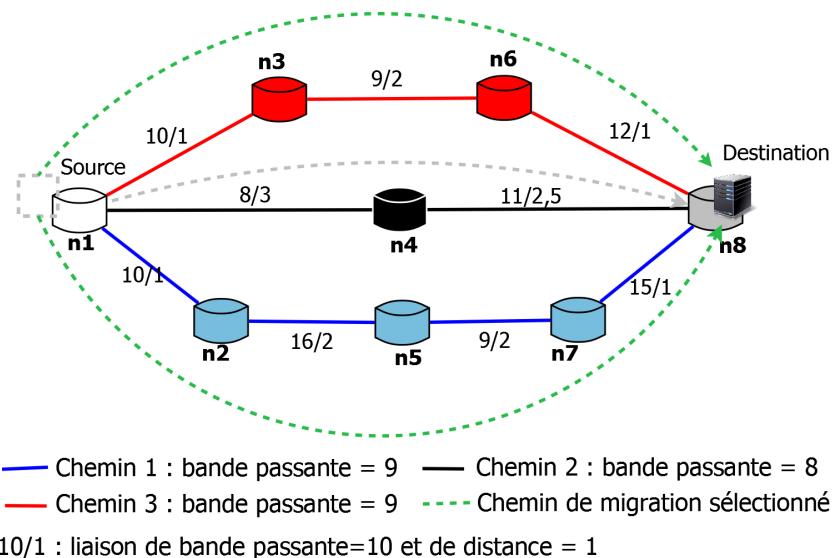


Figure 67 – Sélection des chemins de migration par séparation de flux sur la base du débit de données des liaisons

IV.4. Simulations et discussion

Afin d'évaluer les performances de nos méthodes de repositionnement de serveur, nous avons procédé à des simulations dans l'environnement de simulation OMNeT++ version 5.0. Les réseaux de tests sont les mêmes que ceux présentés au chapitre II. Dans un premier temps, nous évaluons les performances de la méthode de Horiuchi et Tachibana (2018) complétée. Ces performances s'évaluent à travers l'impact du nombre de repositionnements sur la QoS globale ainsi que la durée de migration des services entre un nœud source et un nœud destination. Dans un second temps, nous étudions l'influence de la séparation de flux sur le repositionnement de serveur virtuel, avec la bande passante (FSB-DReViSeR-bandwidth) puis le débit (FSB-DReViSeR-throughput) comme critère de sélection de chemins de repositionnement. Nous procédons à une comparaison des deux approches afin d'en déduire laquelle des deux est la plus appropriée pour le choix de chemin de repositionnement.

IV.4.1. Performances de la méthode de Horiuchi et Tachibana améliorée

IV.4.1.1. Impact du nombre de repositionnements sur la QoS

Dans cette section, nous présentons l'influence du nombre de repositionnements sur la QoS globale dans un réseau virtuel. En effet, nous avons proposé dans la version améliorée des travaux de [Horiuchi et Tachibana \(2018\)](#), de réduire le nombre de repositionnements en éliminant ceux dus à des défauts de QoS inférieurs à une valeur seuil k . Pour les simulations, nous avons effectué pour chacun des réseaux de test, des tests pour différents délais d'initialisation du repositionnement : $k=2\text{sec}$, $k=6\text{sec}$ et $k=10\text{sec}$. Les figures [68](#) et [69](#) illustrent les résultats de cette étude pour un réseau de petite taille (réseau3 de 20 nœuds) et un réseau de grande taille (réseau4 de 60 nœuds) du chapitre [II](#).

Dans le réseau de petite taille (voir figure [68](#)), on constate que le taux de repositionnements est relativement bas par rapport à celui du réseau de grande taille (voir figure [69](#)). Cette observation se traduit par de nombreuses similitudes dans le nombre de repositionnements effectués après chaque variation du délai k de déclenchement du repositionnement. Celà peut s'expliquer par le fait que la présence d'un nombre important de noeuds induit aussi un trafic important ([Horiuchi & Tachibana, 2018](#)). Dans le réseau4 de 60 nœuds, il n'y a presque pas cette similitude. Cela signifie que, plus le nombre de nœuds est élevé, plus il est possible d'avoir des repositionnements et parmi ceux-ci, il y en a beaucoup qui peuvent être évités. Dans tous les cas, ces simulations montrent que notre approche de repositionnement qui impose un délai de repositionnement est meilleure que celle de [Horiuchi et Tachibana \(2018\)](#) qui admet beaucoup de repositionnements inutiles, et présente des temps d'indisponibilité élevés par rapport à la version modifiée.

Concernant le poids du trafic après repositionnement, les figures [70](#) et [71](#) donnent une comparaison de la méthode de gestion de la variation du trafic de [Horiuchi et Tachibana \(2018\)](#) avec sa version modifiée (Horiuchi et Tachibana complété) que nous avons proposée pour le réseau3 de 20 nœuds (voir figure [70](#)) et le réseau4 de 60 nœuds (voir figure [71](#)). Les deux méthodes ont presque les mêmes taux de réduction du trafic, que ce soit dans un réseau de petite ou de grande taille. Néanmoins, il y a des points de différence où l'approche de [Horiuchi et Tachibana \(2018\)](#) réduit plus la quantité de trafic que notre approche. Cette différence s'explique par le fait qu'à cause du délai que nous imposons, certains repositionnements ne sont pas effectués. Dans ce cas, le poids du trafic n'est pas beaucoup affecté.

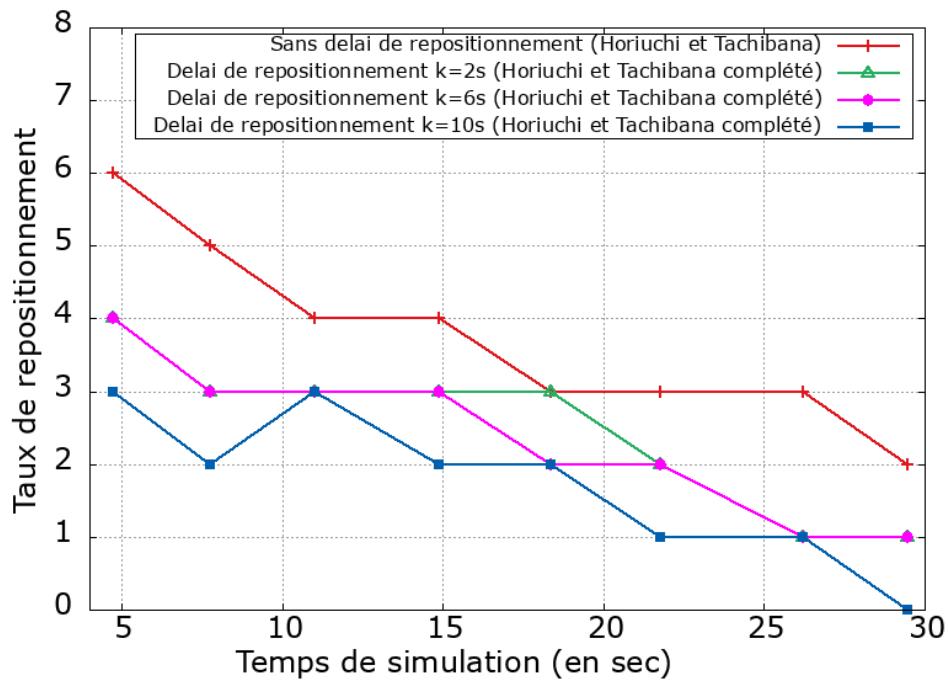


Figure 68 – Taux de repositionnement pour un réseau de 20 nœuds.

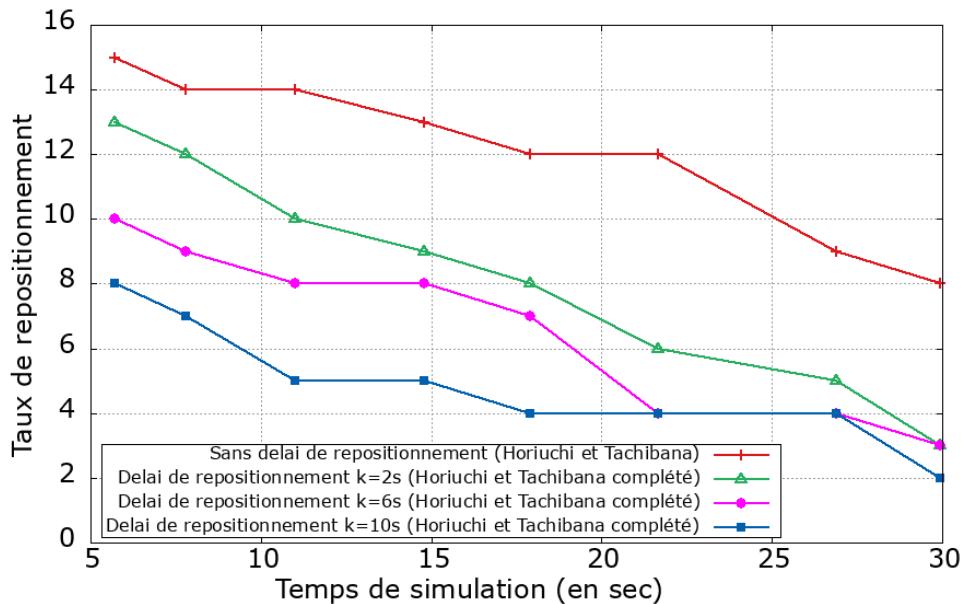


Figure 69 – Taux de repositionnement pour un réseau de 60 nœuds.

IV.4.1.2. Impact du temps de migration sur la QoS

Pour mesurer l'impact du temps de migration sur la QoS, nous avons observé pendant les simulations, le taux de migration et la durée moyenne des migrations sur un intervalle de temps de 0 à 30 secondes de simulation. Nous comparons notre approche qui intègre la migration, avec celle de Horiuchi et Tachibana (2018) qui ne l'intègre pas. Les figures 72 et 73 montrent dans l'ensemble que le temps

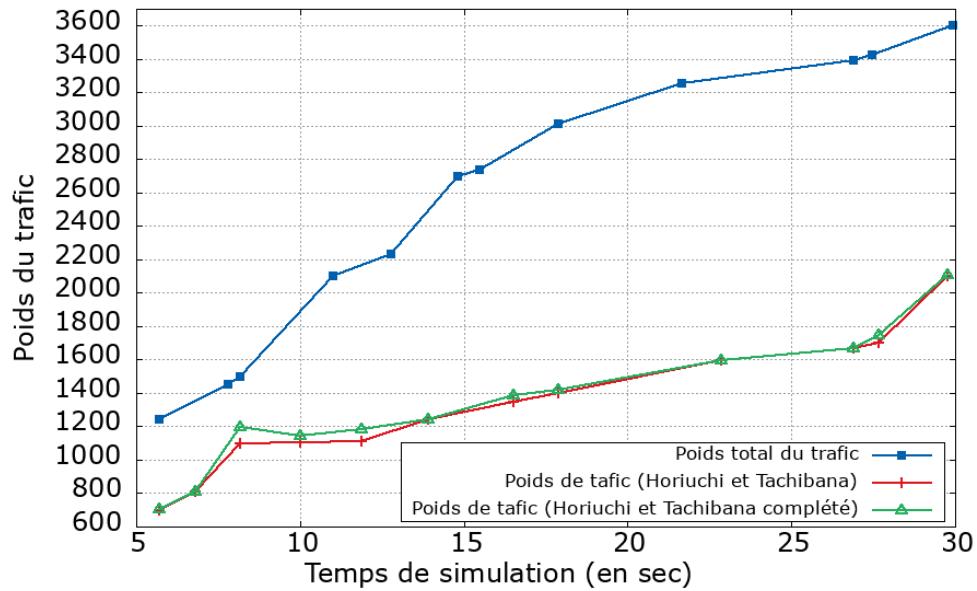


Figure 70 – Variation du poids de trafic pour le réseau de 20 nœuds.

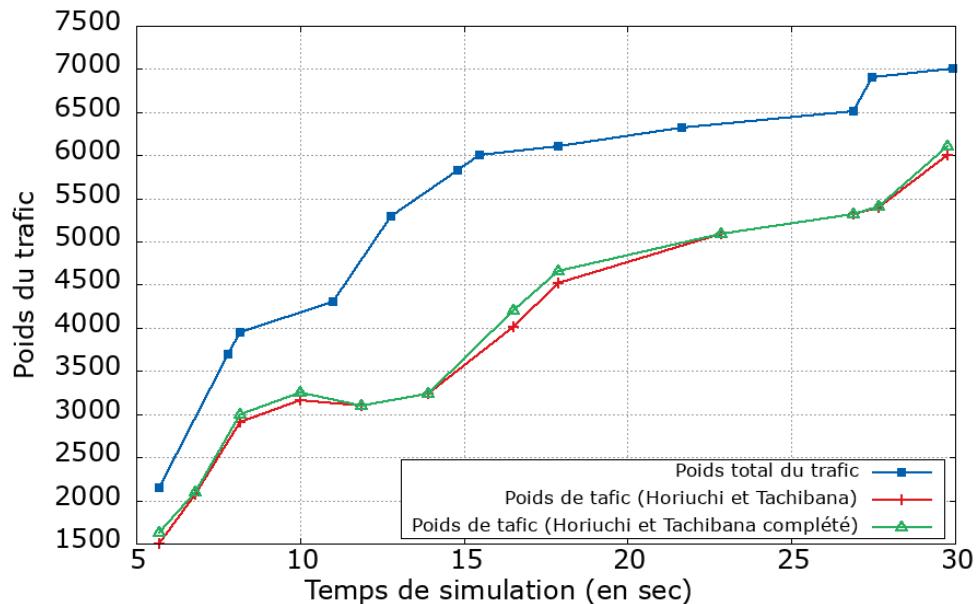


Figure 71 – Variation du poids de trafic pour le réseau de 60 nœuds.

moyen de migration de Horiuchi et Tachibana (2018) est plus bas que le nôtre. Ceci s’explique par le fait que nous considérons lors de chaque migration avec la méthode de Horiuchi et Tachibana (2018) que les données sont transférées en une seule fois sans phase d’arrêt et copie ; la conséquence est la diminution du temps global de migration. Mais, comme nous l’avons montré, cette façon de faire n’est pas très réaliste.

Sur la base de ces résultats, on peut conclure que l’approche de repositionnement de Horiuchi et Tachibana (2018) complétée que nous avons proposée, pos-

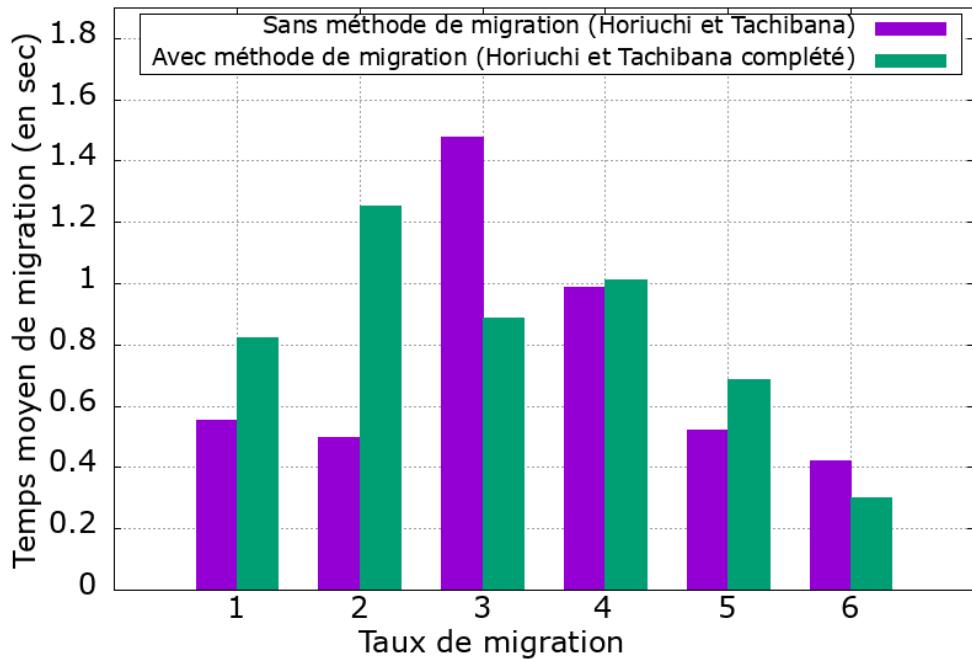


Figure 72 – Variation du temps de migration pour le réseau de 20 nœuds.

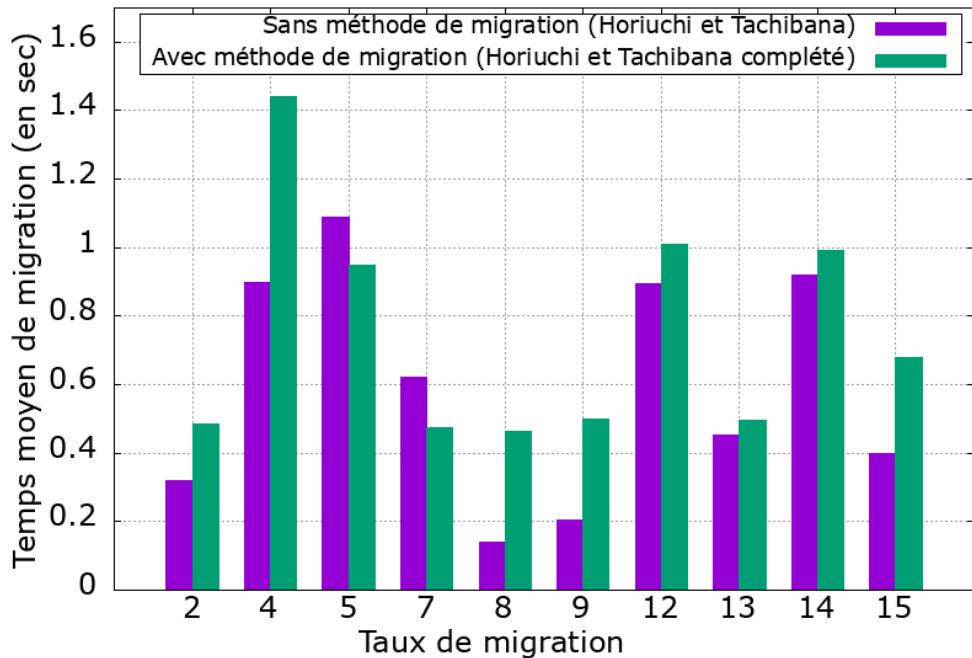


Figure 73 – Variation du temps de migration pour le réseau de 60 nœuds.

sède certes beaucoup de points de similitudes avec celle de Horiuchi et Tachibana (2018), mais elle la complète surtout en intégrant des éléments permettant de mieux apprécier les effets du repositionnement dans un réseau virtuel réel. Même si la contrepartie est un temps de repositionnement plus important lorsqu'on intègre une stratégie de migration.

IV.4.2. Performances du repositionnement par séparation de flux

IV.4.2.1. Impact de la séparation de flux avec usage de Bande passante comme critère de choix de chemins

Dans cette section, nous présentons une comparaison des délais de repositionnement d'un serveur virtuel exploitant notre méthode FSB-DReViSeR-bandwidth basée sur les graphes, avec l'approche de Horiuchi et Tachibana (2018) n'utilisant pas de technique de migration et celle utilisant une technique de migration ((Horiuchi & Tachibana, 2018) améliorée). Les figures 74 et 75 présentent respectivement les résultats obtenus pour le réseau3 (20 noeuds) et le réseau4 (60 noeuds). La technique de migration utilisée pour l'approche de Horiuchi et Tachibana (2018) est la méthode hybride de migration de machine virtuelle (Luo et al., 2008) sur laquelle s'appuie la nôtre. Les simulations ont été effectuées pour chaque technique et les temps moyens de repositionnement pour chaque approche ont été recueillis.

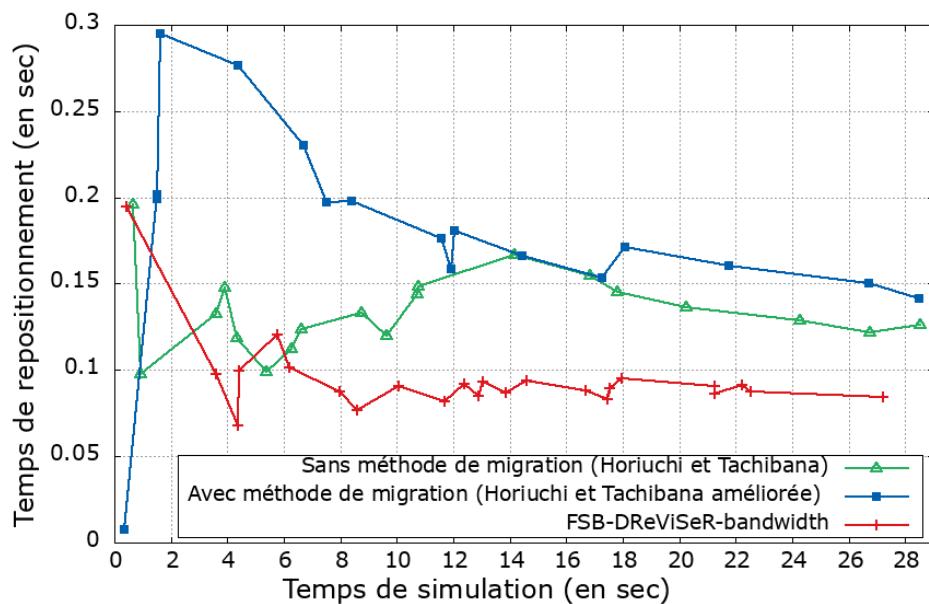


Figure 74 – Temps de repositionnement pour le réseau de 20 noeuds.

Nous constatons que dans le réseau de petite taille, notre méthode de repositionnement FSB-DReViSeR-bandwidth présente en général des temps de repositionnement plus courts que celles de Horiuchi et Tachibana (2018). En effet, cela s'explique par le fait que notre approche exploite d'une part les chemins dont les délais précalculés sont meilleurs et d'autre part, elle sépare le trafic de migration. Toutefois, dans certains cas (même s'ils sont rares), comme par exemple entre les temps de simulation t=11s et t=16s dans le réseau4 de 60 noeuds, l'approche de Horiuchi et Tachibana (2018) présente des meilleurs temps de repositionnement que notre méthode. Cette observation est due au fait que le trafic de migration pris

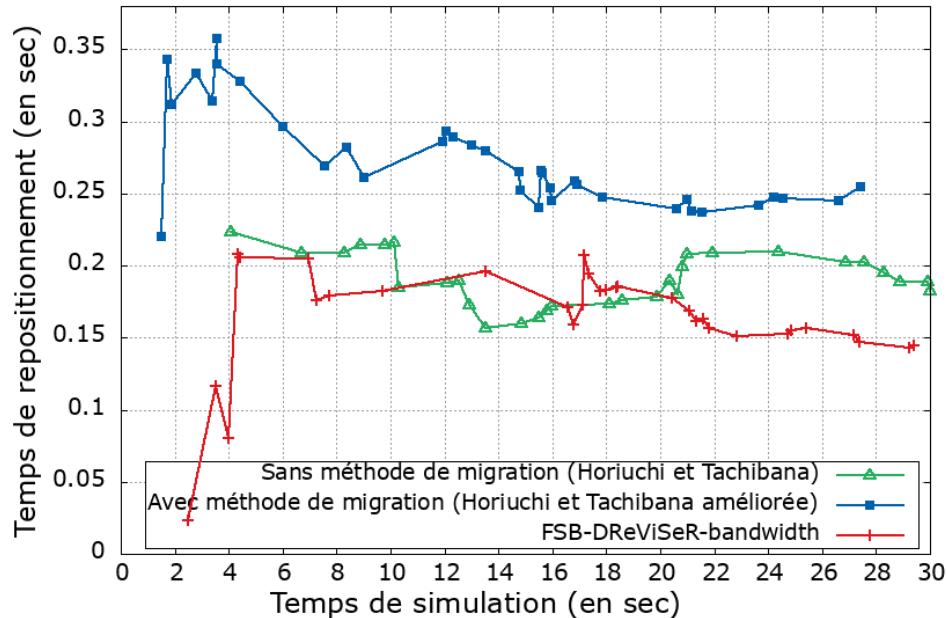


Figure 75 – Temps de repositionnement pour le réseau de 60 nœuds.

en charge par l’approche unicemin de Horiuchi et Tachibana (2018) modifiée, ne nécessitait pas une séparation.

Par ailleurs, notre approche de repositionnement FSB-DReViSeR-bandwidth offre des temps de repositionnement plus intéressants que les deux versions de la méthode de Horiuchi et Tachibana (2018) qui utilisent une topologie arborescente. L’élément qui augmente davantage le temps de repositionnement avec la méthode de Horiuchi et Tachibana (2018), c’est le temps de construction d’un arbre couvrant minimal avant d’effectuer le repositionnement. L’utilisation d’un seul chemin dans cette méthode pour effectuer la migration du service contribue aussi à augmenter ce temps de repositionnement global.

IV.4.2.2. Impact de la séparation de flux avec usage du débit comme critère de choix de chemins

Pour comparer les performances de notre approche FSR-DReViSeR-throughput utilisant le débit des liens (FSR-DReViSeR-throughput) comme un des critères de sélection des chemins de migration avec la version utilisant la bande passante (FSR-DReViSeR-bandwidth), nous avons provoqué des variations du flux de données suivant une loi uniforme (0,1); cela permet de rendre la variation aléatoire afin de se rapprocher plus de la réalité. Les temps de migration moyen ont été recueillis en fonction du nombre de migrations effectués, afin de savoir celle qui propose une meilleur QoS même en présence d’un trafic élevé.

Dans un réseau de petite taille, FSB-DReViSeR-bandwidth offre des délais de migration plus petits en général (voir figure 76). Mais la différence avec FSB-DReViSeR-throughput n'est pas très significative lorsqu'on est dans un réseau de grande taille (voir figure 76). Cela pourrait s'expliquer par le fait que, le trafic étant un peu plus important dans les réseaux de grande taille, alors le débit des liaisons tend généralement à se rapprocher de la bande passante des liaisons. Ainsi, en déterminant même la moyenne des variations entre le débit minimal (v_{min}) et le débit maximal (v_{max}), on obtient dans la majorité des cas, des valeurs très proches de la bande passante. Ce résultat laisse penser que FSB-DReViSeR-bandwidth et FSB-DReViSeR-throughput ont presque les mêmes performances lorsque la taille du réseau est assez importante (plus de 50 nœuds).

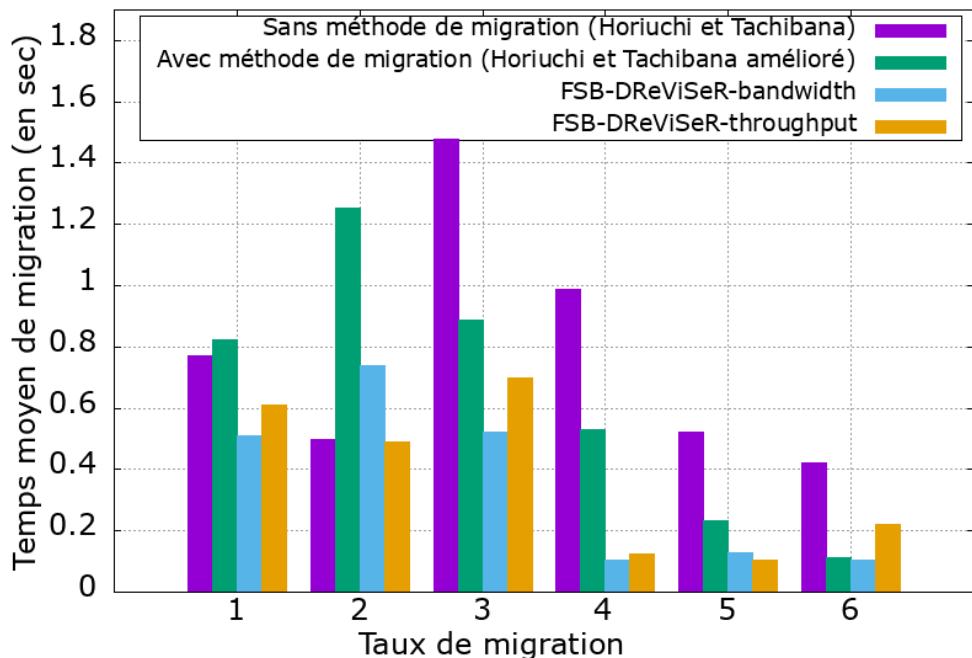


Figure 76 – Temps moyen de migration vs taux de migration pour le réseau de 20 nœuds.

IV.4.2.3. Impact de la séparation de flux sur la cohérence du service après repositionnement du serveur

L'un des problèmes de la séparation de flux, c'est le rassemblage des flux à la destination ; raison pour laquelle il faut réduire au maximum le nombre de séparations. Dans cette section, nous évaluons les performances de notre approche par rapport à son influence sur la cohérence du service après le repositionnement du serveur. De ce fait, nous nous intéressons au taux de perte de paquets de migration, que nous confrontons avec le nombre de repositionnements. Les figures 78 et 79 donnent un aperçu des résultats obtenus. Nous remarquons que, dans les réseaux

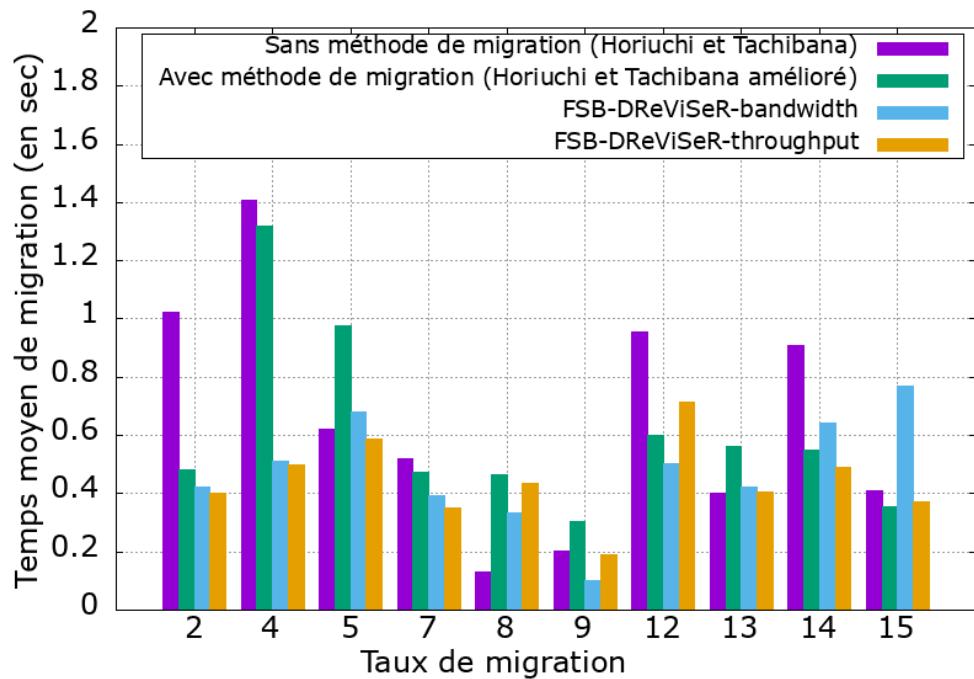


Figure 77 – Temps moyen de migration vs taux de migration pour le réseau de 60 nœuds.

de petite taille (figure 78) tout comme de grande taille (figure 79), le taux de perte de paquets est en général moins élevé avec les deux versions des méthodes de Horiuchi et Tachibana (2018) par rapport aux nôtres. La séparation de flux que nous utilisons, couplée à notre technique de migration, sont responsables de ce résultat. Mais, dans un réseau de grande taille, notre approche offre des taux de pertes de paquets meilleurs lorsque le nombre de repositionnements est important. On peut par ailleurs observer que, FSB-DReViSeR-throughput propose en général des temps de repositionnement meilleurs que FSB-DReViSeR-bandwidth, même si on note aussi des points d'équivalence.

Les résultats présentés permettent de conclure que notre approche de repositionnement dynamique de serveur par séparation de flux surpassé celle de Horiuchi et Tachibana (2018) sur plusieurs points (le temps de repositionnement, la cohérence du service, le temps d'indisponibilité du service virtuel); cette observation est faite surtout lorsque le réseau est de grande taille et le nombre de repositionnements important. Par ailleurs, parmi les deux approches que nous avons proposées, celle utilisant le débit comme un des critères de sélection des meilleurs chemins de repositionnement est meilleure.

IV.5. Conclusion

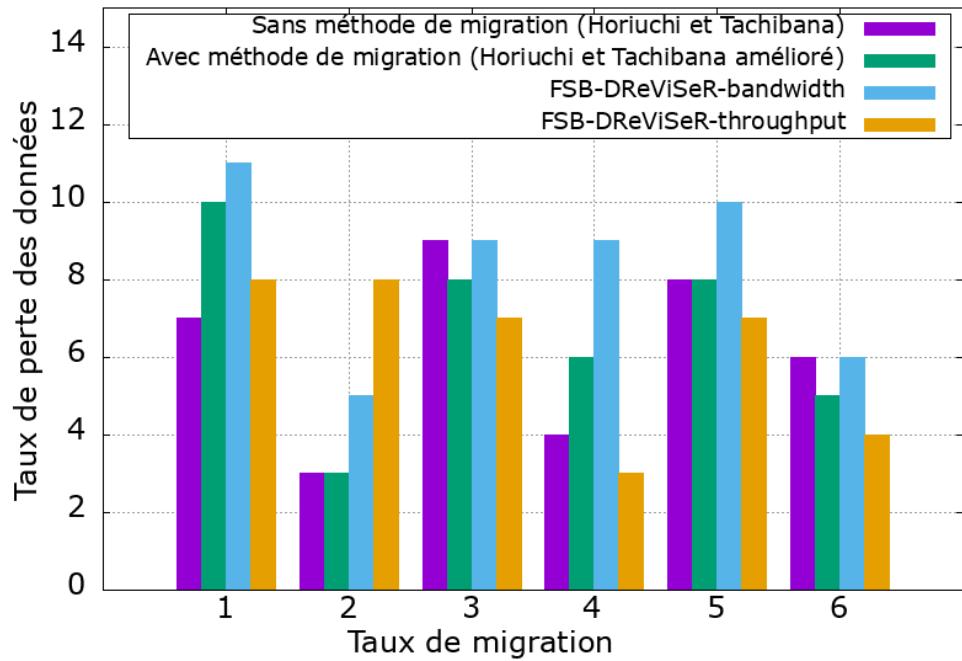


Figure 78 – Temps moyen de migration vs taux de migration pour le réseau de 20 nœuds.

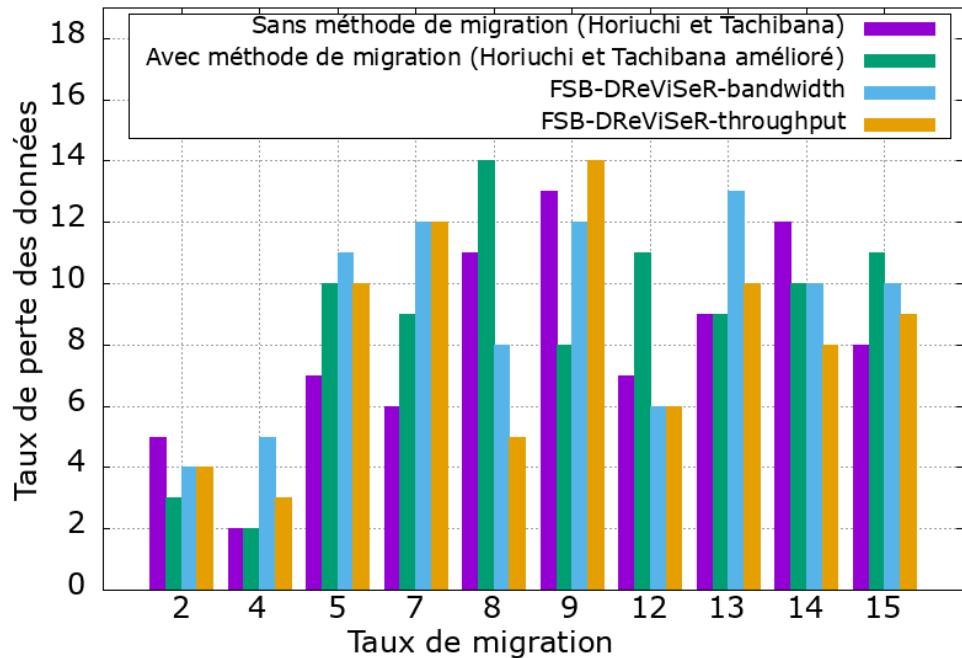


Figure 79 – Temps moyen de migration vs taux de migration pour le réseau de 60 nœuds.

Ce chapitre avait pour objectif d'améliorer la qualité de service offerte par les serveurs virtuels face aux mutations de trafic causant des surcharges au sein d'un serveur virtuel. Ces mutations de trafic sont dues aux activités et requêtes des utilisateurs dans le réseau. Compte tenu de la dépendance des ressources virtuelles du réseau physique, la gestion de ce trafic doit être associée à une politique d'utili-

lisation rationnelle des ressources disponibles. C'est ainsi que dans le contexte de réseaux virtuels hétérogènes, nous avons proposé plusieurs approches de repositionnement dynamiques des serveurs de services virtuels afin de réduire la charge de trafic au niveau de certains nœuds. Nous avons d'abord proposé dans le cadre d'une topologie arborescente, une méthode de repositionnement complétant celle de [Horiuchi et Tachibana \(2018\)](#) grâce à une technique de migration des données. Cette méthode améliorée permet de conserver l'état cohérent du serveur lors de sa restauration dans l'hôte de destination.

Dans le souci d'accélérer davantage le repositionnement et réduire ainsi les temps de latences qui détériorent la QoS, nous avons proposé deux approches de repositionnement de serveurs virtuels basées sur la séparation de flux. La première, FSB-DReViSeR-bandwidth, exploite la bande passante ainsi que la longueur des liaisons réseau, pour sélectionner les chemins les plus rapides afin d'acheminer le trafic de repositionnement du serveur virtuel. La seconde, FSB-DReViSeR-throughput, exploite plutôt le débit et la longueur des liaisons. Les simulations réalisées nous ont permis de conclure que l'approche centrée sur le débit est plus crédible et fiable. Par ailleurs, des comparaisons avec d'autres approches plus récentes comme celle de [Horiuchi et Tachibana \(2018\)](#) et sa version améliorée, ont montré que notre méthode de repositionnement satisfait mieux les critères de QoS dans une topologie de réseau en graphe, qui est plus proche de la réalité que celle en arbre.

CONCLUSION GÉNÉRALE

SOMMAIRE

Rappel de la problématique étudiée	138
Analyse critique des résultats obtenus	138
Quelques perspectives	140

Rappel de la problématique étudiée

Les nombreux avantages offerts par la virtualisation, on favorisé son intégration dans le monde des réseaux informatiques. Cette intégration s'est certes faite de manière progressive, mais presque tous les géants des réseaux aujourd'hui l'ont adopté pour une meilleure exploitation de leurs ressources réseaux. Mais, la dépendance des réseaux virtuels vis à vis du réseau physique en terme de ressources fait qu'il soit assez complexe de fournir une qualité de service satisfaisante aux utilisateurs. Ce problème de qualité de service se pose avec acquitté lorsque le trafic réseau est sujet à de grandes mutations liées à diverses pannes. Dans le cadre de cette thèse, nous avons proposé des solutions de gestion du trafic avec minimisation des ressources réseaux mobilisées (bande passante, débit) pour faire face aux situations de pannes de liaisons et de nœuds, ainsi qu'aux congestions dans les réseaux virtuels. L'objectif majeur de ces solutions est de garantir une bonne Qualité de Service (QoS) même dans les conditions difficiles de fonctionnement.

Analyse critique des résultats obtenus

Nous avons commencé par proposer dans le cadre de la panne d'une liaison virtuelle, une solution de reroutage de trafic exploitant la séparation du flux original en plusieurs parties acheminées sur plusieurs chemins adéquatement choisis

à l'avance. La conséquence immédiate de cette stratégie est la minimisation de la quantité de ressources prélevée sur le réseau physique pour supporter le rerouting. Ainsi, un maximum de pannes de liaison peut être pris en charge et maintenir pendant le plus longtemps possible une QoS satisfaisante dans le réseau. En plus, le faible latency (la gigue) des paquets, induite par notre méthode de séparation de flux combinée au précalcul des chemins, rend notre méthode de rerouting parfaitement adaptée aux réseaux multimédia qui ne tolèrent pas un temps de latency élevé.

Constatant ensuite que les chemins de rerouting précalculés par le contrôleur ne seraient plus optimaux pour router les flux lorsqu'une panne deviendrait persistante dans le temps, nous avons mis en place des méthodes de mise à jour de ces chemins. La plus grosse difficulté pendant cette opération de mise à jour des tables de routage des commutateurs, était d'éviter les pertes de trafic dans le réseau. Ainsi, une structure de message de mise à jour sous forme de vecteur de listes de triplets, joint à une technique de mise à jour séquentielle des nœuds, a permis d'apporter des solutions ; même si la structure de message utilisée nécessite une fragmentation préalable avant tout acheminement dans un réseau de grande taille. Ces solutions sont valables aussi bien dans le contexte de panne d'une seule liaison que de plusieurs liaisons, et même dans le cas d'une panne de nœud virtuel.

Dans la troisième partie de ce travail de thèse, nous avons présenté l'approche de repositionnement de service virtuel. L'objectif est de faire face aux surcharges de trafic au sein des serveurs virtuels dus aux requêtes des utilisateurs et leur mobilité d'un point d'accès réseau à un autre. Le repositionnement consiste à faire migrer un service de son nœud source vers un autre nœud ayant suffisamment de ressource pour prendre en charge la mutation de trafic. Étant donné que pendant le repositionnement, le service concerné est sujet à de nombreuses perturbations et temps de latency, nous avons proposé principalement deux méthodes de repositionnement nommées FSB-DReViSeR-bandwidth et FSB-DReViSeR-throughput. Ces deux méthodes exploitent la technique de séparation de flux en acheminant sur plusieurs chemins les données du service à repositionner. Cela permet de réduire le temps de repositionnement ainsi que les temps de latency observés lors du repositionnement. La stratégie de migration intégrée à nos approches permet de conserver en grande partie la cohérence du service après repositionnement. Ainsi, les métriques de QoS telles que la gigue, le faible taux de perte de paquets et le temps de transit, sont garanties par les méthodes proposées dans cette thèse.

Quelques perspectives

Les méthodes de gestion de trafic mises en place dans cette thèse pour garantir la QoS dans les réseaux virtuels permettent d'envisager plusieurs pistes d'améliorations et d'extension possible pour les travaux futurs.

1- Extension de l'approche de reroutage du chapitre II à une carte de trafic très dynamique

Dans l'approche de reroutage présentée au chapitre II, le contrôleur précalcule les chemins de reroutage sur la base d'une carte de trafic réseau prédéfinie. Mais cette carte de trafic évolue rapidement pendant la durée de vie du réseau, demandant ainsi à réévaluer régulièrement la qualité des chemins précalculés pour rerouter les flux en cas de panne de liaison. L'adaptation de notre stratégie de reroutage à une carte de trafic plus dynamique constitue une perspective à ce travail de thèse.

2- Gestion des dates conflictuelles entre les messages de mise à jour des tables de routage dans les environnements multicontrôleurs

Dans notre stratégie de mise à jour des nœuds dans le cadre des pannes persistantes de plusieurs liaisons, nous avons vu que plusieurs informations de mise à jour liées à des pannes différentes, peuvent se rencontrer simultanément au sein d'un même nœud. Nous avons alors proposé dans une telle situation, d'appliquer les informations de mise à jour de plus anciennes aux plus nouvelles. Mais, les dates de ces messages de mise à jour étant définies par le contrôleur, la question de mise en place d'une stratégie de datation plus efficace dans un environnement multicontrôleur est à envisager. La plus grande difficulté ici est de définir une date consensuelle entre celles proposées par les différents contrôleurs.

3- Intégration du handoving lors de la mise à jour des tables de routage

La prise en compte du phénomène de handoving dans le processus de routage et de mise à jour des nœuds dans un environnement multicontrôleur, constitue une autre perspective liée à ce travail de thèse. En effet, dans ce type d'environnement, chaque contrôleur a la responsabilité d'un ensemble d'équipements dont il gère le trafic grâce au plan de contrôle. Le Handover se produit lorsqu'un utilisateur migre de son domaine d'administration initial pour un autre. Dès lors, comment mettre à jour un nœud virtuel qui migre de son domaine d'administration initial pour un autre ? À ce moment, les questions de stratégie d'enregistrement et d'intégration temporaire ou définitive de cet utilisateur se posent également. Les politiques d'in-

tégration utilisées par les HLR (Home Location Register) et VLR (Visitor Location Register) dans le cadre de la téléphonie mobile pourraient être utiles pour aborder ce problème dans les environnements multicontrôleurs.

4- Repositionnement de serveur au-delà du domaine d'administration initial

L'extension des stratégies de repositionnement de serveur présentées dans cette thèse au-delà du domaine d'administration initial du nœud l'ayant abrité, constitue une autre perspective. En effet, les repositionnements effectués dans cette thèse supposent qu'à chaque fois le nouveau nœud devant recevoir le service hébergé, existe et peut se trouver dans le même domaine d'administration que le nœud virtuel source. Cependant, il peut arriver qu'aucun nœud du domaine d'administration initial ne possède suffisamment de ressources pour gérer le poids du trafic créant la congestion. Doit-on envisager un repositionnement de service au-delà du domaine initial et prendre ainsi le risque de détériorer davantage la QoS à travers l'augmentation considérable de la gigue ? Ou alors devrait-on plutôt maintenir une QoS faible en se contentant de traiter le peu de trafic possible avec les ressources du domaine d'administration initial ?

5- Intégrer la technique de partitionnement de graphe du chapitre II au repositionnement de serveur

L'association de la technique de partitionnement du graphe représentant le réseau en graphe bleu et graphe rouge dans notre stratégie de recherche des chemins de repositionnement peut également être envisagé. Cela permettrait d'isoler rapidement les nœuds n'ayant pas suffisamment de ressource de ceux qui en ont. L'objectif serait dans ce cas de trouver les meilleurs ponts permettant de relier le serveur virtuel source au nœud destination, et même sélectionner rapidement en cas d'existence de plusieurs destinations potentielles, la plus appropriée.

6- Meilleure métrique que le débit moyen

Une autre perspective à ce travail serait l'utilisation d'une métrique meilleure que celle du débit moyen dans la sélection des chemins de migration des données du serveur à repositionner. Comme nous l'avons dit au chapitre IV, la considération du débit moyen comme donnée représentant l'ensemble des variations du débit sur les arcs constituant un chemin de migration, n'est pas un choix optimal. Il faut donc trouver une caractéristique de position ou de dispersion statistique meilleure que le débit moyen et qui intègrerait la majorité des données de variation.

7- Repositionnement de plusieurs services virtuels hétérogènes

Le repositionnement simultané de plusieurs services congestionnés dans un réseau virtuel est également une piste de recherche alternative à ce travail. En effet, dans un réseau virtuel hétérogène offrant plusieurs services différents, leur congestion simultanée peut nécessiter des repositionnements. Dès lors, les questions de disponibilité de nœud de repli et de gestion de la dépendance fonctionnelle entre les services offerts se posent ; car dans un tel contexte, la migration d'un service entraîne la migration du service dont il est dépendant. La cohérence des deux services doit donc être garantie.

BIBLIOGRAPHIE

- 7th Framework Program, E. C. (2010). *Sail*. <https://sail-project.eu/index.html>. Auteur. (Accès le 29 novembre 2019)
- Albrightson, R., Garcia-Luna-Aceves, J., & Boyle, J. (1994). Eigrp—a fast routing protocol based on distance vectors. *Interop 94*.
- Almes, G., Kalidindi, S., & Zekauskas, M. (1999a). *A one-way packet loss metric for ippm* (Rapport technique).
- Almes, G., Kalidindi, S., & Zekauskas, M. (1999b). *A one-way packet loss metric for ippm* (Rapport technique). RFC 2680, September.
- Almes, G., Kalidindi, S., & Zekauskas, M. (1999c). *A round-trip delay metric for ippm* (Rapport technique). RFC 2681, september.
- Atlas, A. K., & Zinin, A. (2008). Basic specification for ip fast-reroute : loop-free alternates.
- Bakshi, K. (2013). Considerations for software defined networking (sdn) : Approaches and use cases. In *2013 ieee aerospace conference* (pp. 1–9).
- Bavier, A., Feamster, N., Huang, M., Peterson, L., & Rexford, J. (2006). In vini veritas : realistic and controlled network experimentation. In *Proceedings of the 2006 conference on applications, technologies, architectures, and protocols for computer communications* (pp. 3–14).
- Boucadair, M., Levis, P., Griffin, D., Wang, N., Howarth, M., Pavlou, G., ... others (2007). A framework for end-to-end service differentiation : Network planes and parallel internets. *IEEE Communications Magazine*, 45(9), 134–143.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.
- Bu, T., Gao, L., & Towsley, D. (2004). On characterizing bgp routing table growth. *Computer Networks*, 45(1), 45–54.

- Carapinha, J., & Jiménez, J. (2009). Network virtualization : a view from the bottom. In G. M. Parulkar & C. Westphal (Eds.), *Proceedings of the 1st ACM SIGCOMM workshop on virtualized infrastructure systems and architectures, VISA 2009, barcelona, spain, august 17, 2009* (pp. 73–80). ACM. Consulté sur <https://doi.org/10.1145/1592648.1592660> doi: 10.1145/1592648.1592660
- Chandra, P., Fisher, A., Kosak, C., Ng, T., Steenkiste, P., Takahashi, E., & Zhang, H. (1998). Darwin : Customizable resource management for value-added network services. In *Proceedings sixth international conference on network protocols (cat. no. 98tb100256)* (pp. 0177–0177).
- Chawda, K., & Gorana, D. (2015). A survey of energy efficient routing protocol in manet. In *2015 2nd international conference on electronics and communication systems (icecs)* (pp. 953–957).
- Chimento, P., & Ishac, J. (2008). Defining network capacity. *RFC5136, IETF, February*.
- Chowdhury, N., & Boutaba, R. (2009). Network virtualization : state of the art and research challenges. *Communications Magazine, IEEE*, 47(7), 20–26.
- Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., ... Warfield, A. (2005). Live migration of virtual machines. In *Proceedings of the 2nd conference on symposium on networked systems design & implementation-volume 2* (pp. 273–286).
- Consortium, P., et al. (2005). *Planetlab | an open platform for developing, deploying, and accessing planetary-scale services.* Consulté sur <http://www.planet-lab.org> (Accès le 23 novembre 2019)
- Crawford, M. (1998). Transmission of ipv6 packets over ethernet networks.
- Dasilva, D.-A., Liu, L., Bessis, N., & Zhan, Y. (2012). Enabling green it through building a virtual desktop infrastructure. In *2012 eighth international conference on semantics, knowledge and grids* (pp. 32–38).
- Demichelis, C., & Chimento, P. (2002). *Ip packet delay variation metric for ip performance metrics (ippm)* (Rapport technique). RFC 3393, November.
- Dong, G.-S., Shen, J., & Sun, L.-Q. (2017). Fast failure recovery in software-defined networks. In *2017 5th international conference on frontiers of manufacturing science and measuring technology (fmsmt 2017)*.

- Doumith, E. (2007). *Agrégation et routage de trafic dans les réseaux wdm multi-couches* (Thèse de doctorat non publiée). École Nationale Supérieure des Télécommunications de Paris.
- Dugaev, D., Zinov, S., Siemens, E., & Shuvalov, V. (2015). A survey and performance evaluation of ad-hoc multi-hop routing protocols for static outdoor networks. In *2015 international siberian conference on control and communications (sibcon)* (pp. 1–11).
- Farhad, H., Lee, H., & Nakao, A. (2015). Software-defined networking : A survey. *Computer Networks*, 81, 79–95.
- Fernandes, N., Moreira, M., Moraes, I., ... O. (2011, Oct). Virtual networks : isolation, performance, and trends. *Annals of Telecommunications*, 66, 339-355.
- Foundation, N. S. (2019). *What is geni ?* <https://www.geni.net/about-geni/what-is-geni/>. Auteur. (Accès le 29 novembre 2019)
- Guirlanger, J. (2015). *Sdn et openflow - randco*. <https://www.randco.fr/actualites/2014/sdn-et-openflow/>. (accès le 29 décembre 2019)
- Haider, A., Potter, R., & Nakao, A. (2009). Challenges in resource allocation in network virtualization. In *20th itc specialist seminar* (Vol. 18).
- Hedrick, C. L. (1988). *Routing information protocol* (Rapport technique).
- Hiertz, G. R., Denteneer, D., Max, S., Taori, R., Cardona, J., Berlemann, L., & Walke, B. (2010). Ieee 802.11 s : the wlan mesh standard. *IEEE Wireless Communications*, 17(1).
- Horiuchi, S., & Tachibana, T. (2018). Dynamic replacement of virtual service resources based on tree topology for mobile users in virtual networks. *Journal of Computers*, 13(12), 1335–1349. doi: 10.17706/jcp.13.12.1335-1348
- Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and openflow : From concept to implementation. *IEEE Communications Surveys & Tutorials*, 16(4), 2181–2206.
- Jain, R., & Paul, S. (2013). Network virtualization and software defined networking for cloud computing : a survey. *Communications Magazine, IEEE*, 51(11), 24-31.
- Jiang, H., Wang, Y., Gong, L., & Zhu, Z. (2015). Availability-aware survivable virtual network embedding in optical datacenter networks. *IEEE/OSA Journal of*

- Optical Communications and Networking*, 7(12), 1160–1171.
- Kamamura, S., Shimazaki, D., Hiramatsu, A., & Nakazato, H. (2013). Autonomous ip fast rerouting with compressed backup flow entries using openflow. *IEICE TRANSACTIONS on Information and Systems*, 96(2), 184–192.
- Keller, E., Lee, R. B., & Rexford, J. (2009). Accountability in hosted virtual networks. In *Proceedings of the 1st acm workshop on virtualized infrastructure systems and architectures* (pp. 29–36).
- Keshavamurthy, U., & Guruprasad, H. (2015). Vm migration : a survey. *Global Journal of Engineering Science and Researches*.
- Khan, M. M. A., Shahriar, N., Ahmed, R., & Boutaba, R. (2016). Multi-path link embedding for survivability in virtual networks. *IEEE Transactions on Network and Service Management*, 13(2), 253–266.
- Kherbache, V. (2016). *Ordonnancement des migrations à chaud de machines virtuelles* (Thèse de doctorat non publiée). Université Côte d'Azur.
- Koyanagi, Y., & Tachibana, T. (2014). *Dynamic resource allocation based on amount of traffic in virtual networks* (Vol. 113; Rapport technique № 473). Consulté sur <https://www.ieice.org/ken/paper/20140306aBkt/eng/>
- Lambert, A., Buob, M.-O., & Uhlig, S. (2009). Improving internet-wide routing protocols convergence with mrpc timers. In *Proceedings of the 5th international conference on emerging networking experiments and technologies* (pp. 325–336).
- Lim, A. O., Wang, X., Kado, Y., & Zhang, B. (2008). A hybrid centralized routing protocol for 802.11 s wmnns. *Mobile Networks and Applications*, 13(1-2), 117–131.
- Liu, J., Li, Y., & Jin, D. (2014). Sdn-based live vm migration across datacenters. In *Acm sigcomm computer communication review* (Vol. 44, pp. 583–584).
- Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y., & Chen, H. (2008). Live and incremental whole-system migration of virtual machines using block-bitmap. In *2008 ieee international conference on cluster computing* (pp. 99–106).
- Malkin, G. (1998). *Rfc2453 : Rip version 2*. RFC Editor.
- Malkin, G. S. (1993). Rip version 2 protocol analysis.
- Mekouar, L., Iraqi, Y., & Boutaba, R. (2010). Incorporating trust in network virtualization. In *2010 10th ieee international conference on computer and information*

- technology (pp. 942–947).
- Moy, J. (1997). Ospf version 2.
- Moy, J. T. (1998). *Ospf: anatomy of an internet routing protocol*. Addison-Wesley Professional.
- Murthy, S., & Garcia-Luna-Aceves, J. J. (1996). An efficient routing protocol for wireless networks. *Mobile Networks and applications*, 1(2), 183–197.
- Myoupo, J. F., Yankam, Y. F., & Tchendji, V. K. (2018). A centralized and conflict-free routing table update method through triplets' lists vector in sdn architectures. In *2018 ieee smartworld, ubiquitous intelligence & computing, advanced & trusted computing, scalable computing & communications, cloud & big data computing, internet of people and smart city innovation (smartworld/scalcom/uic/atc/cbdcom/iop/sci)* (pp. 1509–1515).
- Myoupo, J. F., Yankam, Y. F., & Tchendji, V. K. (2020). On the dynamic replacement of virtual service resources for mobile users in virtual networks. *Journal of Computers*, 15(1), 10-21. doi: 10.17706/jcp.15.1.10-21
- Nascimento, M. R., Rothenberg, C. E., Salvador, M. R., Corrêa, C. N., De Lucena, S. C., & Magalhães, M. F. (2011). Virtual routers as a service : the routeflow approach leveraging software-defined networks. In *Proceedings of the 6th international conference on future internet technologies* (pp. 34–37).
- Ng, W., Jun, D., Chow, H., Boutaba, R., & Leon-Garcia, A. (1999). Miblets : A practical approach to virtual network management. In *Integrated network management vi. distributed management for the networked millennium. proceedings of the sixth ifip/ieee international symposium on integrated network management.(cat. no. 99ex302)* (pp. 201–215).
- Niebert, N., El Khayat, I., Baucke, S., Keller, R., Rembarz, R., & Sachs, J. (2008). Network virtualization : A viable path towards the future internet. *Wireless Personal Communications*, 45(4), 511–520.
- N.M. Mosharaf Kabir Chowdhury, R. B. (2010). *A survey of network virtualization*. Elsevier, IEEE, 54, 862-876.
- Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking : Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3), 1617–1634.

- Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc on-demand distance vector (aodv) routing* (Rapport technique).
- Pham, T. S., Lattmann, J., Lutton, J.-L., Valeyre, L., Carlier, J., & Nace, D. (2012). A restoration scheme for virtual networks using switches. *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, 800-805. doi: 10.1109/ICUMT.2012.6459773
- Postel, J. (1990). Rfc 791 : Internet protocol, september 1981. *Darpa Internet Protocol Specification*, 24.
- Pujolle, G. (2014). *Les réseaux*. Editions Eyrolles.
- Pujolle, G. (2015). *Software networks*. Wiley Online Library.
- Rahman, M. R., Aib, I., & Boutaba, R. (2010). Survivable virtual network embedding. In *International conference on research in networking* (pp. 40–52).
- Rekhter, Y., Li, T., & Hares, S. (2006a, January). *A Border Gateway Protocol 4 (BGP-4)* (RFC). RFC Editor. Internet Requests for Comments. Consulté sur <https://www.rfc-editor.org/info/rfc4271>
- Rekhter, Y., Li, T., & Hares, S. (2006b). Ietf rfc 4271 : A border gateway protocol 4 (bgp-4). Reston : Internet Society.
- Ruth, P., Jiang, X., Xu, D., & Goasguen, S. (2005). Virtual distributed environments in a shared infrastructure. *Computer*, 38(5), 63–69.
- S. Bryant, S. P., & Shand, M. (2013). *Ip fast reroute using not-via addresses*. <https://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-11>.
- Seddiki, M. S. (2015). *Allocation dynamique des ressources et gestion de la qualité de service dans la virtualisation des réseaux* (Thèse de doctorat non publiée). Université de Lorraine.
- Servin, C., & Arnaud, J. (2003). *Réseaux et télécoms : cours et exercices corrigés*. Dunod. Consulté sur <https://books.google.cm/books?id=0QTuGwAACAAJ>
- Shahriar, N., Ahmed, R., Chowdhury, S. R., Khan, A., Boutaba, R., & Mitra, J. (2017). Generalized recovery from node failure in virtual network embedding. *IEEE Trans. Network and Service Management*, 14(2), 261–274. Consulté sur <https://doi.org/10.1109/TNSM.2017.2693404> doi: 10.1109/TNSM.2017.2693404

- Shahriar, N., Ahmed, R., Khan, A., Chowdhury, S. R., Boutaba, R., & Mitra, J. (2016). Renovate : Recovery from node failure in virtual network embedding. In *12th international conference on network and service management, CNSM 2016, montreal, qc, canada, october 31 - nov. 4, 2016* (pp. 19–27). Consulté sur <https://doi.org/10.1109/CNSM.2016.7818396> doi: 10.1109/CNSM.2016.7818396
- Sharma, S., Staessens, D., Colle, D., Pickavet, M., & Demeester, P. (2011). Enabling fast failure recovery in openflow networks. In *2011 8th international workshop on the design of reliable communication networks (drcn 2011)* (pp. 164–171).
- Sharma, S., Staessens, D., Colle, D., Pickavet, M., & Demeester, P. (2013). A demonstration of automatic bootstrapping of resilient openflow networks. In *2013 ifip/ieee international symposium on integrated network management (im 2013)* (pp. 1066–1067).
- Son, P. T. (2014). *Autonomous management of quality of service in virtual networks* (Thèse de doctorat non publiée). University of Technology of Compiegne.
- Specification, O. S. (2009). Version 1.0. 0 (wire protocol 0x01). *Open Networking Foundation*.
- Tchendji, V. K., Yankam, Y. F., & Myoupo, J. F. (2018). Conflict-free rerouting scheme through flow splitting for virtual networks using switches. *Journal of Internet Services and Applications*, 9(1), 13 :1–13 :15. Consulté sur <https://doi.org/10.1186/s13174-018-0085-4> doi: 10.1186/s13174-018-0085-4
- Thomas, M., Costa, D., & Oliveira, T. (2016). Assessing the role of it-enabled process virtualization on green it adoption. *Information Systems Frontiers*, 18(4), 693–710.
- Thorenoor, S. G. (2010). Dynamic routing protocol implementation decision between eigrp, ospf and rip based on technical background using opnet modeler. In *2010 second international conference on computer and network technology (iccnt)* (pp. 191–195).
- Touch, J. (2001). Dynamic internet overlay deployment and management using the x-bone. *Computer Networks*, 36(2-3), 117–135.
- uclp. (2019). *Uclp*. Consulté sur <http://www.uclp.ca> (Accès le 29 novembre 2019)
- Veerásamy, J., Venkatesan, S., & Shah, J. (1994). Effect of traffic splitting on link

- and path restoration planning. In *Global telecommunications conference, 1994. globecom'94. communications : The global bridge.*, ieee (Vol. 3, pp. 1867–1871).
- VMware. (2016). *White paper : Why businesses are adopting network virtualization* (Rapport technique N° 15-VMWA-3054). Auteur. Consulté sur <https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/solutionbrief/partners/intel/vmware-why-businesses-adopting-network-virtualization-white-paper.pdf>
- Wang, J., & Nelakuditi, S. (2007). Ip fast reroute with failure inferencing. In *Proceedings of the 2007 sigcomm workshop on internet network management* (pp. 268–273).
- Woo, A., Tong, T., & Culler, D. (2003). Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on embedded networked sensor systems* (pp. 14–27).
- Xi, K., & Chao, H. J. (2007). Ip fast rerouting for single-link/node failure recovery. In *Broadband communications, networks and systems, 2007. broadnets 2007. fourth international conference on broadband communications* (pp. 142–151).
- Yankam, Y. F., & Kengne Tchendji, V. (2020). Dynamic resource allocations in virtual networks through a knapsack problem's dynamic programming solution. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 31.
- Yankam, Y. F., Myoupo, J. F., & Tchendji, V. K. (2019, mar). A conflict-free routing tables update method for persistent multilink and node failures in SDN architectures. *Journal of Computer Science*, 15(3), 332–345. Consulté sur <https://doi.org/10.3844/jcssp.2019.332.345> doi: 10.3844/jcssp.2019.332.345
- Zhang, L., Berson, S., Herzog, S., & Jamin, S. (1997). Resource reservation protocol (rsvp)–version 1 functional specification. *Resource*.



ANNEXE

PUBLICATIONS ISSUES DE CETTE THÈSE

Revues internationales avec comité de lecture

1. Vianney Kengne Tchendji, Yannick Florian Yankam and Jean Frédéric Myoupo. *Conflict-free rerouting scheme through flow splitting for virtual networks using switches*, Journal of Internet Services and Applications, Vol. 9, No. 1, pp. 1-15, Springer, 2018. DOI : 10.1186/s13174-018-0085-4.
2. Yannick Florian Yankam, Jean Frédéric Myoupo and Vianney Kengne Tchendji. *A Conflict-Free Routing Tables Update Method for Persistent Multilink and Node Failures in SDN Architectures*, Journal of Computer Science, Vol. 15, No. 3, pp. 332-345, Science Publications, 2019. DOI : 10.3844/jcssp.2019.332.345.
3. Jean Frédéric Myoupo, Yannick Florian Yankam and Vianney Kengne Tchendji. *On the Dynamic Replacement of Virtual Service Resources for Mobile Users in Virtual Networks*, Journal of Computers, Vol. 15, No. 1, pp. 10-21, 2020. DOI : 10.17706/jcp.15.1.10-21.

Conférence internationale

1. Jean Frédéric Myoupo, Yannick Florian Yankam and Vianney Kengne Tchendji. *A Centralized and Conflict-Free Routing Table Update Method through Triplets' Lists Vector in SDN Architectures*, 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1509-1515, 2018. DOI : 10.1109/SmartWorld.2018.00261.

RESEARCH

Open Access



Conflict-free rerouting scheme through flow splitting for virtual networks using switches

Vianney Kengne Tchendji¹, Yannick Florian Yankam¹ and Jean Frédéric Myoupo^{2*}

Abstract

The weaknesses of the Internet led to the creation of a new network paradigm – network virtualization. Virtualization is a very successful technique for sharing and reusing resources, which results in higher efficiency. Despite its advantages, including flexibility in network architecture, virtualization imposes many challenges, such as physical resource allocation to virtual devices. An efficient allocation strategy for these resources can ensure good Quality of Service (QoS) in virtual networks, whether in node or link failure events. This paper presents a conflict-free rerouting scheme with efficient additional capacity usage for link and node failure resilience in a virtual network using switches. Combining an IP Fast Rerouting approach and flow-splitting strategy, this scheme provides short reaction time, stable performance and low complexity because the rerouting calculation and configuration are performed in advance. We show that rerouting by traffic splitting based on the entering arc and destination is sufficient to address all link-failure situations in the network, assuming that the network is two-link connected. After modelling the dimensioning problem as an Integer Linear Programme, we demonstrate through practical implementation of our rerouting scheme on different networks that the scheme can substantially minimize the additional capacity draw on the substrate network. A solution using multiple virtual planes is also provided to solve several conflict problems in the case of simultaneous multiple link failures.

Keywords: Network virtualization, Quality of service (QoS), Restoration scheme, Spare capacity, Traffic splitting

1 Introduction

Since its creation, the Internet has brought innovations and success to industry, economic and research fields; however, deployment of any new, radically different technology and architecture is becoming highly difficult, a situation that cloud computing can mitigate. That effect is what we call Internet ossification [1]. To fend off ossification, studies have proposed rethinking the architecture of the actual Internet [1]. However, network virtualization is the most promising approach to addressing current limitations of the Internet and supporting new requirements [2–5]. Its principle is to implement multiple virtual routers on each physical machine and to interconnect them through substrate network architecture. This implementation allows virtual networks to have different logical

topologies from that substrate network, and each of them behaves as a true network in which it is possible to implement different routing protocols and services. As in the substrate network, failures could occur in virtual networks; in this case, rerouting mechanisms can be implemented to forward traffic by using available resources in the virtual network or additional ones taken from the physical network. This additional resource could cause dysfunctional risks inside another virtual plane.

1.1 Previous work

The link and node failure problem has been investigated for a long time in the framework of physical networks but not in virtualized networks because of the new requirements brought by this technology [6] (e.g., architecture flexibility, mobility, and isolation). Our restoration scheme is pre-calculated. Therefore, the rerouting paths for all link failure cases are determined and recorded inside the controller in advance. When a failure

* Correspondence: jean-frederic.myoupo@u-picardie.fr

²Computer Science Lab-Mis, University of Picardie Jules Verne, Amiens, France

Full list of author information is available at the end of the article

occurs, the nodes apply the pre-calculated rerouting paths directly. Multiple methods based on the IPFRR (IP Fast Reroute) strategy have been proposed for transient failures. However, they have the following limitations:

- For rerouting schemes using a single path and additional capacity [7, 8], the limits of the physical resources are quickly reached, which paralyses the network.
- For loop-free alternative mechanism-based methods [9], we are not certain of rerouting traffic to all destinations; doing so only helps to reduce the number of lost packets in an IP network.
- Not-via addressing [10] and tunnelling [11] mechanisms require encapsulation and de-encapsulation of packets, whereas in a multiple routing configuration mechanism [12], the packets must carry configuration information. With the appearance of optic networks, methods that modify the packets are not recommended but can instead be used to optimize the usage of resources in network virtualization.
- Multipath rerouting [13] using spare capacity in the network can induce a capacity saving of up to 11% in randomly generated networks, but lack of spare capacity due to the existence of multiple virtual planes on a substrate network can undermine this result in network virtualization.

Additionally, [14–16] propose methods to find back-up paths that permit rerouting traffic in the case of link failure but not in the context of network virtualization. Proposed schemes also based on IPFRR use two port types: primary and back-up ports. The traffic will change from the primary port to a back-up port only when there is a failure on the primary port or the traffic comes from the primary port. This packet forwarding strategy uses a bridge to reconnect sub-graphs coming from a failure but does not consider conflicts, which can disturb traffic. In contrast to these works, our rerouting scheme uses not just one but multiple bridges, and it avoids conflict on rerouting paths. Our strategy extends the results in [7] and can minimize the dimensioning cost of the network. Recently, a restoration scheme was proposed in [7, 8] to handle both node and link failure problems by replacing all or parts of a network with switches managed by an external controller. The authors summarized the previous works [17–20] and noted their drawbacks. They derived a rerouting scheme negating these drawbacks. Their rerouting model uses only one path to reroute traffic and to minimize the additional capacity used by a virtual network; it cannot solve the multiple-link-failure problem. Moreover, their work in [7, 8] has drawbacks:

- Network modelling using only one rerouting path cannot significantly minimize additional capacity, which means that physical resource limits are quickly reached and the network is paralysed for a moment. Because the physical resource limits are quickly reached in [7, 8], the number of failures handled is also reduced.
- Based on their rerouting model, the authors in [7, 8] claim that a conflict-free rerouting scheme for the multiple-link-failure situation cannot be constructed even when the network remains connected.

The work in [21] presents a virtual network embedding model that allows a virtual link to map multiple substrate paths. This model can help to build a rerouting strategy using multiple planes.

1.2 Our contribution

We initially propose a new scheme to solve the link and node failure problems in a network of switches that avoids conflict on rerouting paths in contrast to protocols in [6, 7]. The new rerouting scheme we present here clearly uses not just one but multiple bridges, and it avoids conflict on rerouting paths. Our strategy uses a flow-splitting technique [22–25], extends the results in [7, 8] and can minimize the dimensioning cost of the network. Flow splitting is a method for restoring traffic from a failed link using multiple rerouting paths in the case of insufficient residual capacity. In this first contribution, we consider only one virtual plane. We propose a rerouting scheme that ensures that for any link or node failure, the traffic will be rerouted until it reaches the destination through an alternative path.

Given that there generally are no rerouting solutions that avoid conflicts in all network configurations, our second contribution is to avoid these situations. To reach this goal we use multiple planes to provide a rerouting strategy that avoids conflicts that could not be solved by using only one plane. Therefore, we introduce here a new scheme to solve the link and node failure problems in a network of switches. This new scheme negates the drawbacks of [7, 8] by showing that a conflict-free rerouting scheme for a multiple-link-failure situation can be constructed even when the network remains connected. The rerouting scheme described in this paper uses filters in switches to determine the next hop for the incoming flow. We provide each virtual network a specific filter. Then, the controller sets the flow path by programming the switches in the form of quadruplets (S, N, O, F) in which S is the source port (node), N the current node, O the output ports because flow can be split and forwarded through different paths to the same destination depending upon the spare capacity needed, and F the filter, which indicates the destination.

More precisely, for an incoming flow from a neighbour and a given destination, the scheme will assign the potential output ports. In the case of failure, only the upstream node must react by directing the disturbed traffic to one or many of its neighbours. The traffic is routed according to the filter programmed in each node of the network. Traffic can be split anytime at the level of each node if needed. Hence, the proposed scheme needs only a local reaction, making its implementation particularly easy in distributed environments. This local reaction helps the network operate normally and can solve the problem of transient failures. A transient failure is a failure whose duration is short, less than 10 min, whereas the duration of a persistent failure is longer [7]. When the failure is determined to be persistent, the controller can recalculate the routing table for all nodes in the network. To avoid the rerouted traffic of a failure causing disturbances to another part of network, additional capacity is added to all arcs. Because this additional capacity is added in the physical network and is exploited by several virtual layers, it is necessary to minimize it. The mathematical model that we propose in this paper can calculate the rerouting paths and optimize the total additional capacity injected into the network.

To the best of our knowledge, presently our work is the only one that shows how to effectively solve simultaneous multilink failures using flow splitting methods, thus providing an improvement in QoS of computer networks.

The rest of this paper is organized as follows. The next section presents motivations for traffic splitting. Section 3 provides a full description of our restoration scheme for a link failure configuration. Our mathematical model is described in Section 4. Section 5 presents numerical results of implementations. Section 6 studies the application of our rerouting scheme to the single-node-failure problem. Section 7 extends our work to simultaneous multiple-link-failure situations by providing a solution for some conflict problems. Finally, Section 8 ends the paper.

2 Motivation for traffic splitting

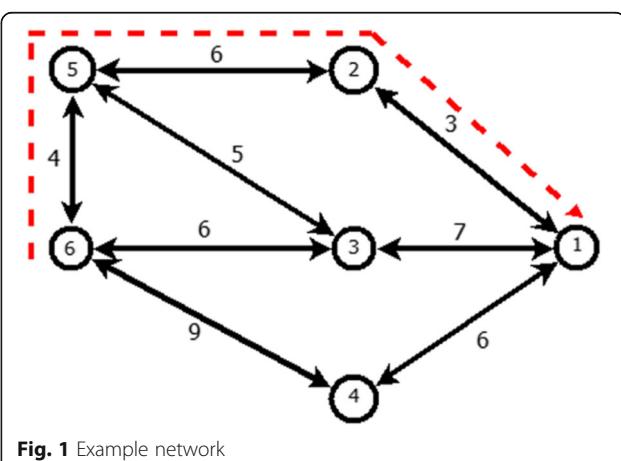
2.1 Improvements of computer networks Qos

Virtual networks use physical network resources to achieve their needs. In the case of link or node failure, spare capacity available in safe links is commonly used to restore traffic. However, this spare capacity might not be sufficient to carry the entering traffic; this situation represents a lack of spare capacity and is the main motivation for using flow splitting in the network. The idea is to split an original flow into multiple parts such that they can be forwarded easily through the network. This method induces numerous potential advantages as:

- Reduction of transit delay and packet loss rate, because the flows are more able to reach their destination nodes, thus improving the network QoS;
- improvement of the packets routing delays: since the original flow is split into several lighter-sized streams, they can be transported more quickly to the destination;
- improvement of load balancing distribution [26], leading to prevent or decrease congestion risk across the network. This is a well-known benefit of flow splitting in computer networks.
- extension of the lifetime of a network by allowing more flexible and efficient resources allocation;
- better economy of the substrate network resources supporting virtual networks. Since virtual networks are built on a physical network infrastructure, it is necessary to avoid an abuse of these resources with the risk of causing the hosted virtual networks malfunctions.

We illustrate this last point by an example. Figure 1 shows a network with 6 nodes and 8 links. The numbers carried by each arc represent spare capacity available on the links. There is only one path (of traffic) 6–5–2–1 going through link (5, 2). Other paths are not shown for clarity. Let the traffic of this path be 8 units, and let all links be of equal length. In link restoration, a faulty link is replaced by one alternative path.

Assume that faulty link (5, 2) is replaced by path 5–3–1. Because links (5, 3) and (3, 1) have only 5 and 7 spare capacity each, both links need, respectively, 3 and 1 more spare capacity to make the restoration possible. This example proves that the use of traffic splitting in the link restoration scheme can result in lower spare capacity requirements but in the context of a virtual network, it could be very important to reduce additional capacities added to the links to avoid disturbances due to the rerouting scheme.



Now, let us consider the traffic-splitting version of link restoration (Fig. 2). There are two alternative paths, 6–5–3–1 and 6–3–1, for link (5–2), and each alternative carries 4 units of traffic. (As in the model [22], the general design principles presented in this paper are valid for any unit of bandwidth capacity for virtual networks.) With this second method, there is no need for more spare capacity.

2.2 The packets re-ordering problem

Beyond its advantages, flow splitting also brings some difficulties, the best known of which are the following:

- Avoiding the packet re-ordering. When a stream is subdivided into the network, the different parts must be reassembled without losing no part (a potential TCP performance problem [27, 28]), making the use of flow splitting strategy very delicate. So, we should try to make reordering rare. Therefore avoiding this reordering until the packets reached the destination is crucial. However to overcome this problems some approaches were elaborated. Some strategies separate traffic at the level of flows. This approach removes the problem of reordering but at the cost of a restriction in the granularity with which we can split traffic [29]. Another one operates on bursts of packets (flowlets) carefully chosen to avoid reordering, but allowing a finer granularity of load balancing [30]. Some other algorithms [31, 32] minimize or eliminate reordering in some situations. But, some reordering problem should occur, and probably often enough to affect performance of IP networks;
- the problem of reassembling packet segments inside the destination node. Due to various reasons, such as multipath routing, route fluttering, and retranmissions, packets belonging to the same flow

may arrive out of order at a destination. The problem is how to know which packet comes before or after another one when we want to rebuild the original flow packets [27–29]. Some algorithms based on packet numbering in [29] can be used to at least minimize reordering.

In this work, flow splitting is implemented by building little flow of packets from an original one. To face reordering challenge, we use the numbering packets each time the flows are split, because this method does not modify significantly the packets headers.

3 New multipath link failure restoration scheme

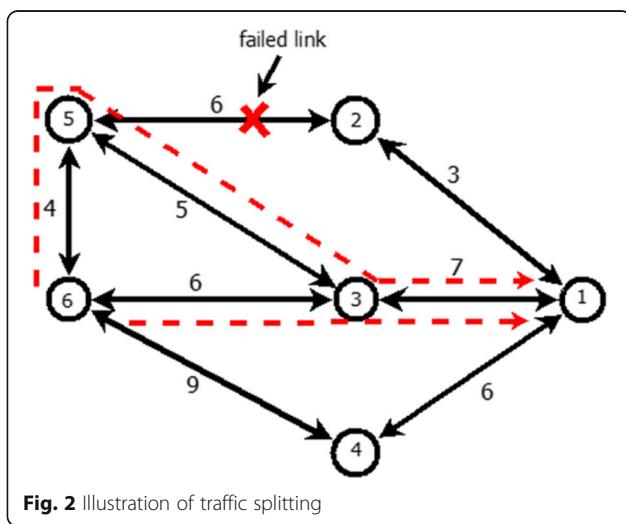
In this section, we present our rerouting scheme that addresses the case of a link failure. As presented in Section 2, traffic splitting occurs because spare capacity is lacking in the network, but implementation of that approach in our rerouting scheme helps to solve many other problems:

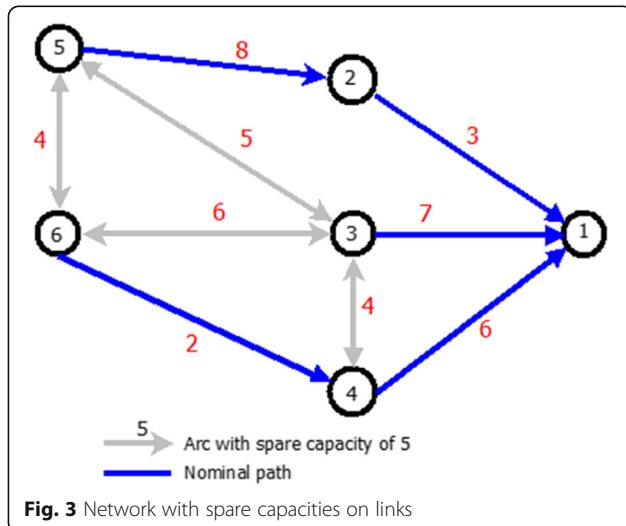
- Safeguarding of network resources by minimizing spare and additional capacity usage to manage more traffic
- Possible rerouting, however, is impossible to do with only one path, as shown in [7].

A routing tree called a nominal routing tree is associated with a given destination; this tree is constructed using the shortest path tree criterion. We assume that the routing is provided. In the case of a failure of an arc or edge (both arcs are then concerned) and a lack of spare capacity in links, we reroute the traffic through one or more alternative paths. When there is only one path used, our rerouting scheme is similar to [7]. According to the routing scheme, for two independent failures, if two rerouting paths to a given destination have a common arc, they must merge after this arc. This requirement holds for both nominal and rerouting paths. If two paths do not satisfy this requirement, we say that they are in conflict. Any routing scheme satisfying this requirement is said to be without conflict. In this strategy, only the extremity failed link nodes will know about the failure. The upstream nodes initiate the traffic diversion, whereas all other nodes in the network apply the filter for each incoming flow without any difference between disturbed and non-disturbed flows. Because the disturbed traffic is rerouted on multiple alternative paths and should satisfy the non-conflict requirement, the cost in terms of resources and of computational time is expected to be higher compared with conventional schemes using single path rerouting.

We illustrate this rerouting scheme in Figs. 3 and 4.

These figures represent a network with 6 nodes and 8 links.

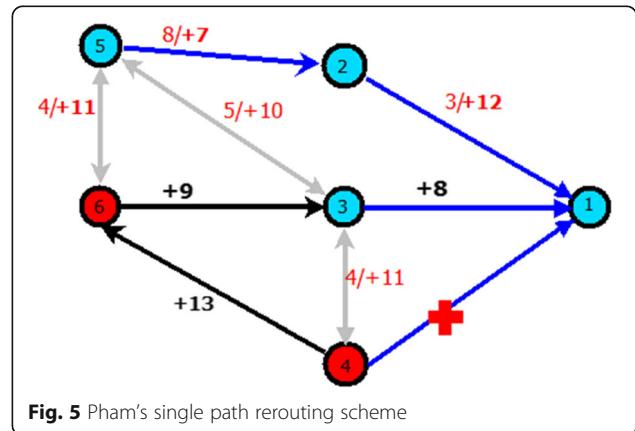


**Fig. 3** Network with spare capacities on links

The original graph with the spare capacity of each link is shown in Fig. 3.

Assume that the source node is node 6 and that the destination node is node 1. We also suppose that the failure link 4–1 generates a flow weight of 15 units to reroute. Figure 4 represents each link's weight and the additional capacity required on each arc if rerouting paths use this arc. If we reroute using only one alternative path, similar to [7], the path 4–6–3–1 will be selected as the rerouting path (see Fig. 5), which consumes $T = 13 + 9 + 8 = 30$ units of additional capacity.

When applying our rerouting scheme (Fig. 6), part 4–6 of the previous rerouting path will be preserved. From node 2, traffic can be split into two parts because there are two arcs leaving node 6 to node 1. Our splitting criterion is the amount of spare capacity available on links. Thus, a node will send some flow on the link that offers the greatest spare capacity, and the remaining flow will be sent on the other link. Therefore, 6 units of traffic

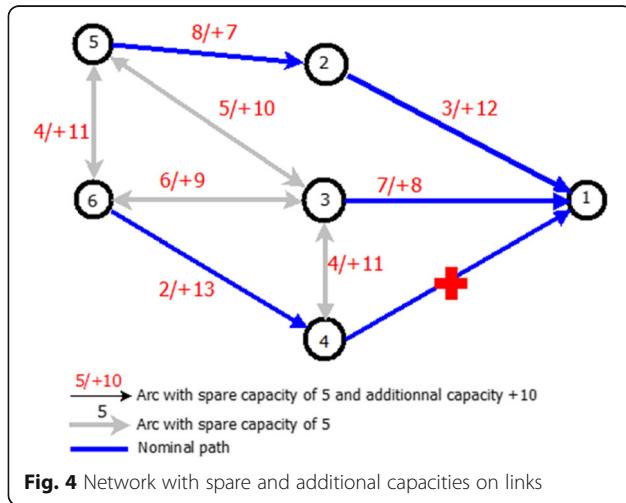
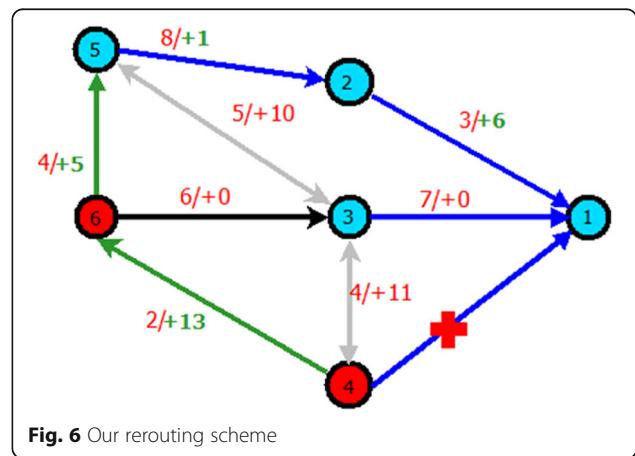
**Fig. 5** Pham's single path rerouting scheme

will be sent on arc (6 → 3) (which is a bridge), offering a spare capacity of 6, and 9 will be sent on arc (6 → 5) (another bridge), which offers a spare capacity of 4. In this example, we use an additional capacity of 5 on arc (6 → 5) in addition to its existing spare capacity. The rerouting paths are 4–6–3–1 using $13 + 0 + 0 = 13$ additional capacity, and 4–6–5–2–1 using $13 + 5 + 1 + 6 = 25$ additional capacity. Arc (4 → 6) belongs to the two rerouting paths and involves a total additional capacity usage of 13. Therefore, the total additional capacity needed for these two paths is 25, i.e., a total additional capacity savings of 17% relative to [7]. This analysis shows that our rerouting scheme can substantially minimize the additional capacity needed on links to reroute the traffic of a failed link. Because of the configuration of filters, traffic will be rerouted until it reaches the destination without any conflict.

4 Mathematical formulation

4.1 Description of our model

This section provides a rerouting mathematical model based on the following assumptions:

**Fig. 4** Network with spare and additional capacities on links**Fig. 6** Our rerouting scheme

- The graph is assumed oriented and symmetric.
- There are at least two disjoint arc paths between any two nodes of the graph.
- There is only one link failure at a time.

To resolve the question of the existence of a rerouting solution without conflict, we have the following theorem:

Theorem 1. *For any destination d , there is a rerouting plan without conflict using one or many alternative paths.*

The formal mathematical proof can be found in the Appendix.

As in [7], a similar mathematical formulation can be provided for our case, but we add in the equations the number of rerouting paths. Consider the following notation:

- Rt^d : set of arcs of the routing tree to destination d
- Ac_{ab} : additional capacity assigned to arc (a, b)
- Tn^d_n : total traffic for d that passes through node n . n is the node that detects the failure. In fact, the failure is characterized by a source n and a destination d because we use the routing tree for nominal routing. For a destination d , the failed arc is the one routing the traffic going to d and coming from n by nominal routing.
- F_i : indicates fictive nodes used to divert traffic in the case of failure. We introduce the fictive nodes F_i that will be used for all failures. For a given failure (n, d) , the traffic to d will be rerouted by i paths from F_i to d starting with arc (F_i, n) , $i = 1, 2, \dots$
- $SRed_n^d$: $SRed_n^d$ sub-tree of sink n . Recall that in the case of failure, the tree is divided into two parts, the isolated part, that is the Red part, and the Blue part. Alternative paths will reroute traffic from the Red part to the Blue part.
- $SBlue_n^d$: sub-tree of sink d , with $Rt^d - SRed_n^d SRed_n^d$
- $y_{efgh}^{dn} y_{efgh}^{dn}$: this binary variable indicates whether the e^{th} alternative path to destination d for a given failure contains arcs (f, g) and (g, h) ; node n is the node that detects the failure.
- x_{efgh}^d : x_{efgh}^d this binary variable indicates the rerouting scheme to destination d . It takes value 1 if there is a failure in which the e^{th} alternative path to destination d contains arcs (f, g) and (g, h) . Therefore, the variable takes value 1 if there are n and e where $y_{efgh}^{dn} y_{efgh}^{dn}$ is equal to 1.
- α_{eab}^{dn} : α_{eab}^{dn} this binary coefficient equals 1 if arc (a, b) belongs to one of the paths in the nominal routing from n to d except the failed arc.
- **Quadruplet**: All quadruplets (e, f, g, h) where e is the number of an alternative path in a rerouting

scheme, f, g, h are nodes of the graph; f can be the fictive node, and (f, g) and (g, h) are two adjacent arcs, with $f \neq h$.

- **Arc**: All arcs of the graph
- **L**: Set of links
- **N**: Set of nodes

The objective is to minimize the sum of additional capacity allocated to each arc; our objective function is provided by (1):

$$\min \sum_{(a,b) \in Arc} Ac_{ab} \quad (1)$$

$$\sum_{\substack{h=h_s \\ (n,h_s) \in Rt^d, h \text{ neighbor of } n, h=h_1}} y_{eF_0nh_s}^{dn} = 1, n \in N, d \in N, s = 1, 2, \dots, e = 1, 2, \dots \quad (2)$$

$$\sum_{\substack{h=h_s \\ h_s \in \text{neighbor of } g, h=h_1}} x_{efgh}^d \leq 1, (f, g) \in Arc, d \in N, s = 1, 2, \dots, e = 1, 2, \dots \quad (3)$$

$$y_{efgh}^{dn} = 0, d \in N, n \in N, j \in SRed_n^d, (f, g) \in Rt^d, (e, f, g, h) \in \text{Quadruplet}, e = 1, 2, \dots \quad (4)$$

$$\sum_{\substack{h=h_s \\ f \in N | (e, f, g_s, h_s) \in \text{Quadruplet}}} y_{efgh_s}^{dn} \leq 1, d \in N, n \in N, (g_s, h_s) \in Arc, g_s \in SRed_n^d, h_s \in SRed_n^d, e = 1, 2, \dots \quad (5)$$

$$\sum_{\substack{h=h_s \\ f_s \in \text{neighbor of } f}} y_{efgh_s}^{dn} = \sum_{\substack{h=h_s \\ h_s \in \text{neighbor of } g}} y_{efgh_s}^{dn}, d \in N, (f, g) \in Arc, f \neq f_s, f \neq n, g \in N-d, s = 1, 2, \dots, e = 1, 2, \dots \quad (6)$$

$$\sum_{e=1}^{e=k} \sum_{s=1}^{s=k} y_{engh_s}^{dn} = y_{eF_0ng}^{dn}, d \in N, n \in N, (n, g) \in Arc, k = 1, 2, \dots, e = 1, 2, \dots \quad (7)$$

$$y_{eghh_1}^{dn} - y_{efgh}^{dn} \geq 0, h \in SBlue_n^d - d, (h_1, h) \in Rt^d, (g, h) \in Rt^d, \forall d \in N, \forall (e, f, g, h) \in \text{Quadruplet}, e = 1, 2, \dots \quad (8)$$

$$\sum_{n \in N} \sum_{e=1}^{e=k} y_{engh}^{dn} \geq x_{efgh}^{dn} \geq \frac{\sum_{n \in N} y_{engh}^{dn}}{\text{cardinal}(N)}, (e, f, g, h) \in \text{Quadruplet}, n \in N, k = 1, 2, \dots, e = 1, 2, \dots \quad (9)$$

$$\begin{aligned}
& \sum_{d \in N | (n,m) \in Rt^d} \sum_{f \in \text{neighbor of } g, f \neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn} \cdot Tr_n^d \\
& + \sum_{d \in N | (m,n) \in Rt^d} \sum_{f \in \text{neighbor of } g, f \neq h} \sum_{e=1}^{e=k} y_{efgh}^{dn} \cdot Tr_m^d \leq Ac_{gh} \\
& + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{egh}^{dn} \cdot Tr_n^d + \sum_{d \in N | (m,n) \in Rt^d} \alpha_{egh}^{dm} \cdot Tr_n^d, \\
& (g, h) \in Arc, g \neq n, g \neq m, (n, m) \in L, k = 2, 3, \dots, e = 1, 2, \dots
\end{aligned} \tag{10}$$

$$\begin{aligned}
& \sum_{d \in N | (n,m) \in Rt^d} \sum_{i=1}^{i=k} y_{eF_i nh}^{dn} \cdot Tr_n^d \leq Ac_{nh} \\
& + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{nh}^{dn} \cdot Tr_n^d, (m, h) \in Arc, h \neq n, (m, n) \in L, e = 1, 2, \dots
\end{aligned} \tag{11}$$

$$\begin{aligned}
& \sum_{d \in N | (m,n) \in Rt^d} \sum_{i=1}^{i=k} y_{eF_i mh}^{dn} \cdot Tr_m^d \leq Ac_{nh} \\
& + \sum_{d \in N | (n,m) \in Rt^d} \alpha_{mh}^{dm} \cdot Tr_n^d, (m, h) \in Arc, h \neq n, (m, n) \in L, e = 1, 2, \dots
\end{aligned} \tag{12}$$

$$x_{efgh}^d \in \{0, 1\}, \forall (e, f, g, h) \in Quadruplet, \forall d \in N, e = 1, 2, \dots \tag{13}$$

$$y_{efgh}^d \in \{0, 1\}, \forall (e, f, g, h) \in Quadruplet, \forall d \in N, \forall n \in N, e = 1, 2, \dots \tag{14}$$

The objective function will allow us to evaluate the ratio between the additional capacity and the installed capacity.

Equation (2) is a constraint implying that there exist multiple paths resulting from flow splitting, which go from n to d for disturbed traffic.

Equation (3) ensures that there is no conflict in the rerouting, i.e., the incoming flows to node n to destination d must follow the same rerouting paths. If we use arcs $(i, k_1), (i, k_2), (i, k_3), \dots$ for rerouting to destination d , there is at most one output (k_s, j_s) for each.

To avoid loops and conflict problems, the alternative paths should not contain any arc of nominal routing in the red part of the network. Equation (4) ensures that condition.

Constraint (5) ensures that there will be no loop in the network. For a given destination and a given failure, an alternative path could contain a loop if the flows go from a node with a larger index number to another one with a smaller index. This constraint prohibits this type of problem.

Equations (6), (7), and (8) are the flow constraints for the continuity of the alternatives paths. Equation (6) is

the constraint of flow conservation. Referring to (7), the total amount of entering traffic in n is equal to the total outgoing traffic of g ; because of flow splitting being used for a given failure and a destination, we could have multiple incoming streams and possibly multiple outgoing streams.

Equation (8) ensures that in the blue part, if a path uses an arc of the nominal routing tree, it must continue until destination d .

Equation (9) is a constraint for the relationship between two rerouting paths that avoids a conflict (see the definitions of variables x and y). Because x and y are binary variables, with the same quadruplet (e, f, g, h) and same destination d , we can deduce from (9) that (x) will take the maximum value of (y) . We use the sum of failures divided by the cardinal to reduce the number of constraints.

Equations (10), (11), and (12) are the capacity constraints. For each failure of edge (n, m) , the constraint in (10) assumes rerouted paths for arcs (n, m) and (m, n) , and only trees that contain the arc failure are involved. They also consider the released bandwidth on the initial routing paths. Equations (11) and (12) are special cases of (9) for the nodes that detect the failure, node n and node m . Finally, (13) and (14) indicate that the variables take binary values.

4.2 Convexity of our model

The objective function of our model has the general form:

$$\min_{s.t. x \in C} f(x) \tag{15}$$

where C is the set Arc and f is a function over C giving the additional capacity needed for a chosen arc. The problem described by Eq. (15) is convex in the set C and the function f is convex.

Under the existence assumption of at least two paths between any pair of nodes of the graph and considering each arc of the set Arc as a segment, then the set C is convex as an intersection of convex subsets. According to [33], if the function f is affine, it is convex and the problems described by general Eq. (15) are usually stated convex problems with an implicit convexity. This implicit convexity is because there are more explicit formulations

Table 1 Restoration rate without conflicts

Networks	Number of nodes	Number of links	Number of failures	Restoration
Network1	5	7	7	7
Network2	10	18	18	14
Network3	20	31	28	23
Network4	60	81	70	61

Table 2 Restoration rate neglecting conflicts

Networks	Number of nodes	Number of links	Failures	Restoration
Network1	5	7	7	7
Network2	10	18	18	14
Network3	20	31	28	25
Network4	60	81	70	65

of convex problems such as convex optimization problems in functional form, which are convex problems of the form:

$$\min_{s.t. g_i(x) \leq 0, i=1,2,\dots,m, h_j(x)=0, j=1,2,\dots,p} f(x) \quad (16)$$

Where $f, g_1, \dots, g_m: \mathbb{R}^n \rightarrow \mathbb{R}$ are convex functions and $h_1, \dots, h_p: \mathbb{R}^n \rightarrow \mathbb{R}$ are affine functions. Each constraint of our model can be written under any of the forms of constraints of Eq. (16). This proves that our model is implicitly convex.

5 Implementation and simulation results

Based on the comparative study in [34], the OMNet++ network simulator has many advantages: Unlike NS-2 and NS-3, OMNet++ has extensive graphical user interface (GUI) and intelligence support, provide good computation times. The flexibility of the NED language used for describing the network architecture is appropriate to meet the great topology flexibility requirements of network virtualization. OMNet++ is also able to carry out large scale network, which is an important feature for our simulations. That is why the experiments have been conducted in the simulation environment OMNet++ running on a computer with the following configuration: Core i5 2.40 GHz, 4.00 GB RAM, 12 MB cache. We

applied our model to 4 networks: network1 (5 nodes and 7 links), network2 (10 nodes and 18 links), network3 (20 nodes and 31 links), and network4 (60 nodes and 81 links). These four networks satisfy the assumptions cited above. They contain a set of nodes with high degree for the estimation of the impact of multilink failures adjacent to the same node on those networks. It therefore shows the robustness of our strategy. The restoration rate of our rerouting scheme without conflict between rerouting paths is shown in Table 1. The simulations have been done on non-simultaneous multiple link failures for each tested network.

The data provided in Table 1 show that our rerouting scheme supplies rerouting solutions for almost all link failure situations considered. If the conflict constraint is neglected, we can find solutions for more failure configurations (see Table 2), but the variation is small (approximately 2%). In other words, conflict constraint does not significantly affect the number of failures handled.

Figure 7 graphically compares our rerouting scheme with that of [7]. This figure shows that the restoration rate gap between both methods increases with network size. This phenomenon is observed because the potential conflicts in a small network are not numerous, which means that cases of unsolvable conflicts are also not numerous.

We also perform an additional capacity consumption test for the previous four networks; the results are shown in Table 3. This table consists of five columns. Descriptions of the rightmost three columns are as follows: “Unused CA” represents the additional capacity available in the network; “Our used CA” represents the total of additional capacity used by our strategy for link-failure handling; and “Used by X” represents additional capacity used in the network by method [7]. The result units are expressed in seconds.

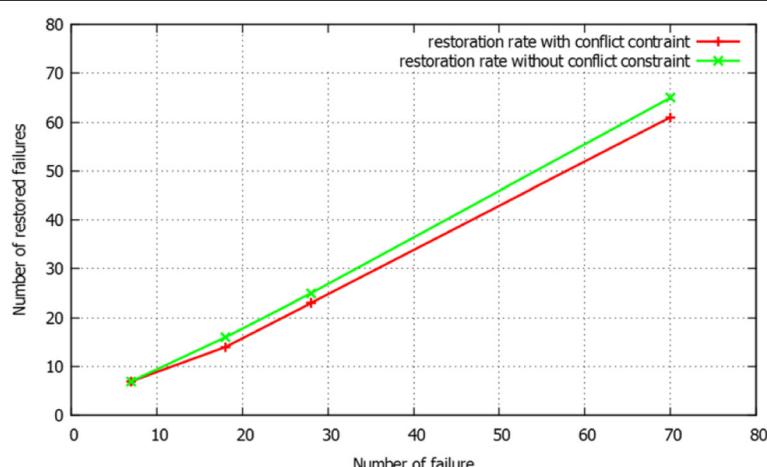


Fig. 7 Comparative graph of two versions of our rerouting scheme: with conflict constraint and without

Table 3 Comparison of our rerouting scheme with [7] concerning additional capacity used

Networks	Number of nodes	Number of links	Unused CA	Our used CA	Used by X
Network1	5	7	3342	1012	1624
Network2	10	18	7216	2433	2741
Network3	20	31	14,000	3802	4524
Network4	60	81	12,052	4110	5021

The results in Table 3 show that our rerouting scheme based on a traffic-splitting strategy uses less additional capacity than does the method presented in [7]. This difference is very important when the network size increases. Figure 8 provides us a better illustration of this difference. This figure shows that the additional capacity used for flow restoration increases with network size and network connectivity.

6 Node failure problem

We speak of node failure when some flow can no longer go through a given node in the network. This situation can be caused by overflow traffic in this node or a physical failure of the given node. Because of the two-link connectivity included in our hypothesis, a node failure leads directly to the outage of at least two or several links; in other words, node failure can be treated as a simultaneous multiple link failure. In this case, failure will be detected by all nodes connected to the failed node. Figure 9 presents the failure of node number 4.

When node 4 fails, flows coming from nodes 1 and 7 must be rerouted. The failure of node 4 implies a simultaneous failure of links (1–4), (7–4) and (4–6). Therefore, we must reroute the two flows (1–4) and (7–4) to

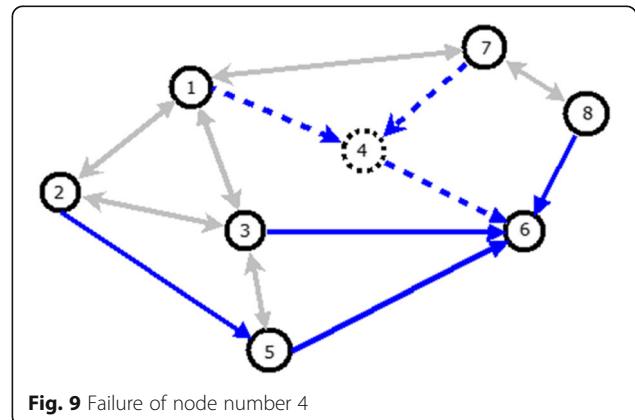


Fig. 9 Failure of node number 4

destination 6 without conflict. To solve this type of failure, two solutions are possible:

6.1 First solution resort to the controller

In the case of node failure, each switch that detects the failure sends a specific message called *packet-in message* to the controller that sets the rerouting order for the link failures related to these nodes. The idea of this rerouting approach is to solve these link failures as cascading failures. This order can be built on a node's label criteria. The nodes are labelled in a decreasing order as we approach the destination node. We could handle the failure detected by the node of a smaller label before another one with a larger label. Once the resolution order is fixed, the controller updates the routing tables of involved nodes as described in [7] by using another specific message called *packet-out message*. After this update, our rerouting scheme can be used to solve each link failure. The reaction time of this solution is too long, due to pro-activity; therefore, the principles of IPFRR are not satisfied with this solution.

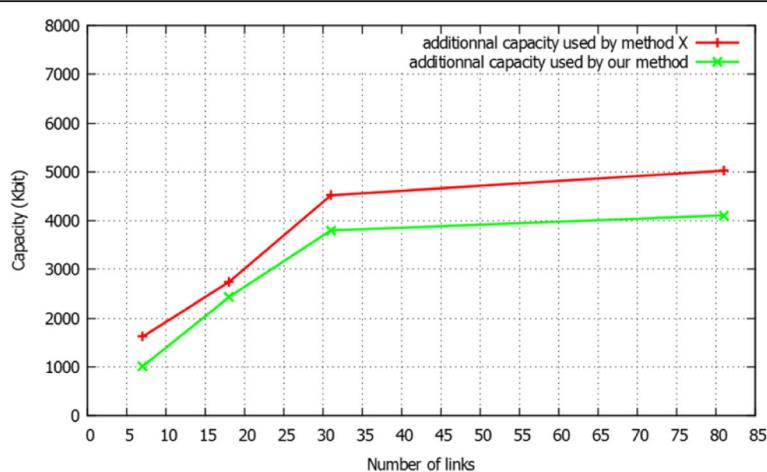


Fig. 8 Additional capacity used by different strategy

6.2 Second solution: No resort to the controller

Each link failure because of a node failure is handled locally and instantly by each node that identifies a link failure. The incurred risk in this strategy is the looping problem during flow rerouting; however, assuming our constraint imposing traffic from nodes with a lower label on another one with a higher label, cycles can be avoided. Our rerouting scheme for a simple link failure can be used to fix simple node failure situations. Recall that we speak about simple node failure when only one node fails at a time. Our rerouting strategy for simple node failure problems uses this approach. The following illustrates our rerouting scheme for a simple node failure with an example. Fig. 10 shows the nominal routing tree of an example network, and Fig. 11 presents link failures (dotted links) resulting from node 3's failure. Fig. 12 presents a cyclic problem resulting from a node failure, and Fig. 13 illustrates the efficiency of our solution to solve this cycle problem.

For each destination, we determine the nominal routing tree from each node towards this destination (see Fig. 10). The failure of node 3 generates simultaneous failures of links (5–3), (6–3) and (3–1) (dotted links). Nodes 5 and 6 will detect the breakdowns of links (5–3) and (6–3), i.e., we have two flows to reroute. These failures split the graph into two parts: the blue part and the red part (see Fig. 11).

Using our rerouting scheme, the flow coming from link (5–3) could be rerouted through arcs (5→2), (5→6) and (5→8). The flow of link (6–3) could follow arcs (6→8), (6→4) and (6→5). However, arc (5→6) can lead to node 3 through arc (6→3) or keep the rerouted flow in cycle 5–6–8–5 (see Fig. 12). Thus, arc (6→3) will be excluded from the list of potential paths for rerouting the flow coming from link (5–3) or node 3. If we consider the criteria related to management of the cycles, arc (5→6) will be considered in rerouting the paths of link (5–3), which will not be true of arc (6→5) (see Fig. 13).

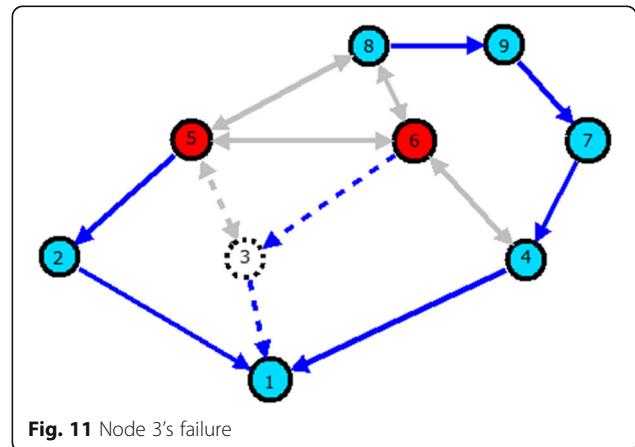


Fig. 11 Node 3's failure

Similarly, for rerouting link (6–3), arc (5→6) could also be used rather than (6→5).

Consequently, the possible rerouting paths will be 5–2–1, 5–8–9–7–4–1 and 5–6–4–1 for flow from the failure of link (5–3); concerning the flow from the failure of link (6–3), the possible rerouting paths could be 6–8–9–7–4–1 and 6–4–1. We can conclude that local reaction required by IPFRR strategy can also be preserved when addressing simple node failure situations through our rerouting scheme.

To achieve the local connectivity recovery, there is a filter similar to an agent, running inside each switch (example of OpenFlow switches) used in network architecture like ours. This agent detects the port states and acts as needed. For classical switches, there are control mechanisms provided to check that ports status.

Multiple link failures studied in the case of simple node failure involve links adjacent to that node, but we also have cases of simultaneous multiple link failures not adjacent to the same node.

7 Simultaneous multilink failures

We speak about simultaneous multiple link failures when several links fail at the same time. The case

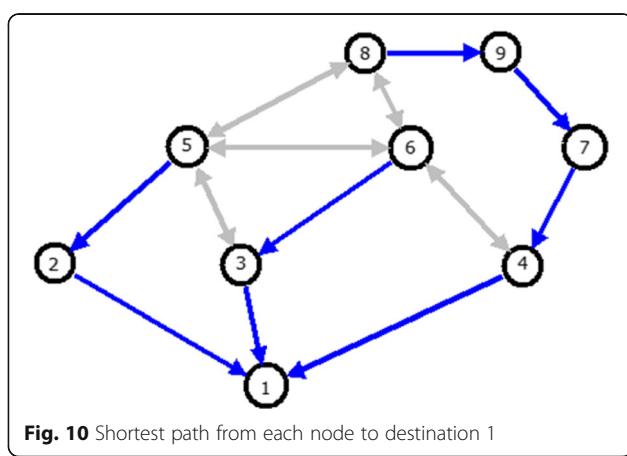


Fig. 10 Shortest path from each node to destination 1

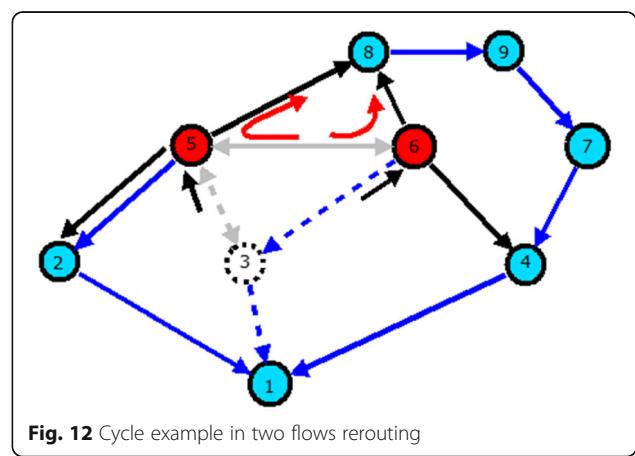
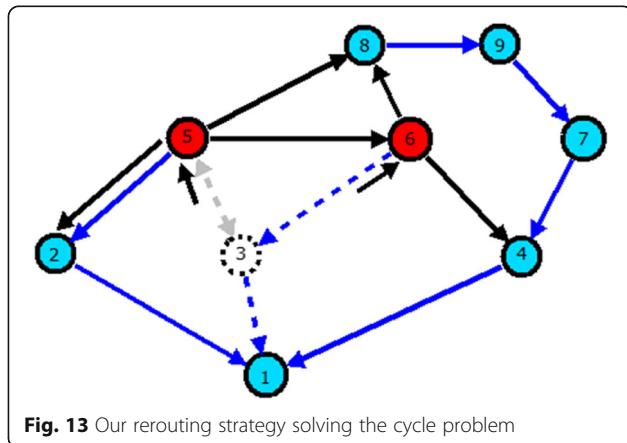


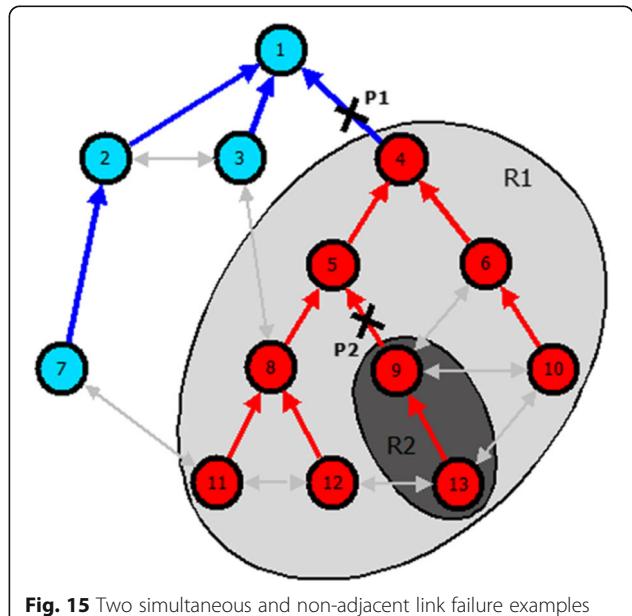
Fig. 12 Cycle example in two flows rerouting

**Fig. 13** Our rerouting strategy solving the cycle problem

considered in this section concerns non-adjacent links to the same single node. In this case, there are multiple nodes, each of which detects a link failure as in the simple node failure case. This type of failure can also be handled using either of two methods:

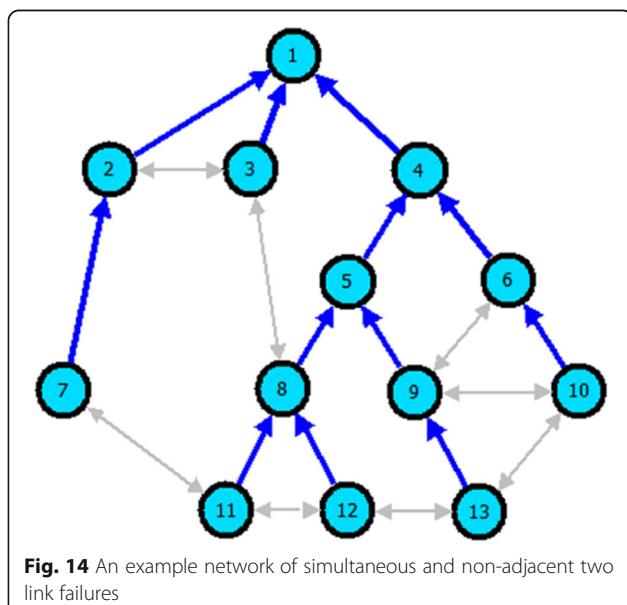
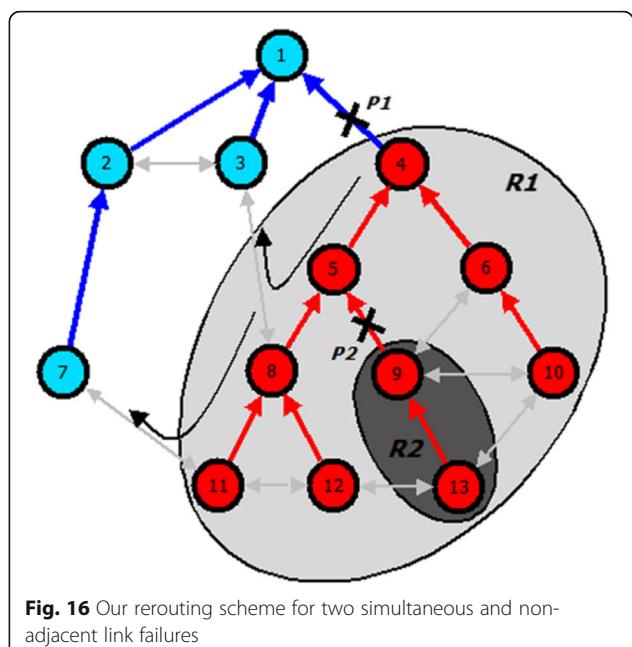
7.1 First method: Treat only one link failure at a time

In this approach, despite many link failures occurring at the same time, they are handled as non-simultaneous link failures; therefore, failures are treated sequentially. This method is used in [7], in which a rerouting scheme is proposed to solve the problem for the case of two links failing simultaneously. As stated in Section 7 about the node failure problem, the limit of this strategy is its slowness in rerouting.

**Fig. 15** Two simultaneous and non-adjacent link failure examples

7.2 Second method: Treat all link failures at the same time

This approach is similar to the second one presented in Section 7 for the node failure problem, and it enables all nodes that detect a failure to initiate the rerouting process. Our rerouting scheme for node failure can also be used here. When several link failures occur simultaneously during the rerouting process, we can use flow splitting each time to find spare capacity lacking in the network.

**Fig. 14** An example network of simultaneous and non-adjacent two link failures**Fig. 16** Our rerouting scheme for two simultaneous and non-adjacent link failures

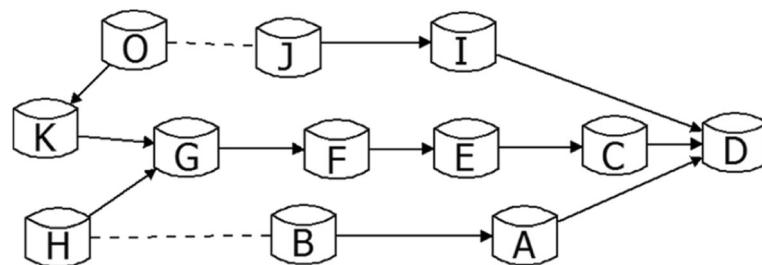


Fig. 17 Unsolvable conflict when using only one plane

Consider the example network of Fig. 14 to illustrate our rerouting strategy for the case of simultaneous and non-adjacent two-link failures.

The nominal routing tree is shown, and the destination node is labelled 1.

Figure 15 shows two link failures named p_1 and p_2 occurring at the same time.

The failures p_1 and p_2 create the red parts R_1 and R_2 . p_1 is detected by the node labelled 4, and p_2 is detected by the node labelled 9. The rerouting scheme of p_1 can be through bridges (8–3) and (11–7) leading to the paths 4–5–8–3–1 and 4–5–8–11–7–2–1 (see Fig. 16). Flows can be split at node numbers 8, 11, 2 and 3. Concerning rerouting of p_2 , link (12–13) can be considered a bridge that connects the red part R_1 to R_2 in addition to (8–3) and (11–7). After the link failure pair (p_1, p_2), if another occurs (pair (p_3, p_4) for example), the rerouting will be done based on the previous one to avoid conflict.

However, concerning this simultaneous multiple link failure, there are several conflict configurations that require particular attention as shown in Fig. 17. For the configuration example shown in Fig. 17, [7] affirms that the conflict problem illustrated is insoluble. Indeed, the rerouting scheme provided in [7] uses only one path for

rerouting, with management of conflict between the paths similar to our strategy. We prove that this potential unsolvable conflict claimed by [7] can be solved by using multiple planes. The principle is to cross from one plane to another when there is a risk of unsolvable conflict when using a single plane.

Consider the configuration example given by [7] in Fig. 17, in which the authors claim that there is no rerouting solution. Two simultaneous link failures situations are considered: first, we have simultaneous link failures (A–D) and (C–D). Second, we have simultaneous link failures (E–C) and (I–D).

According to [7], when (A–D) fails, the only available rerouting path is A–B–H–G–F–E–C–D because if we choose path H–G–K, there would be a conflict at node G. When (C–D) fails, the traffic that comes from failure (A–D) will be rerouted by C. To reroute the traffic of failure (C–D), node C must transfer the traffic back to G; then, there are two possibilities: use G–H–B–A–D as the rerouting path, or transfer the traffic through link (G–K). We cannot use G–H–B–A because the traffic would be transferred indefinitely between A and C. Therefore, we must use F–G–K as the rerouting path in this situation.

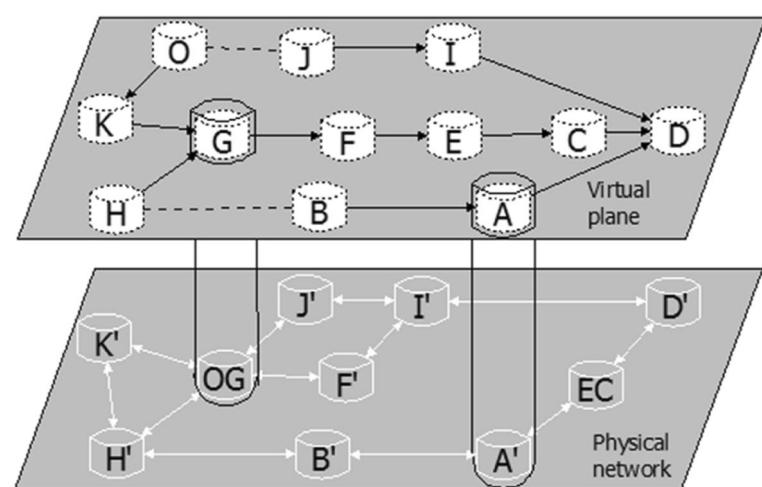


Fig. 18 Solution through multiple planes

Concerning the second situation in which the two links (E-C) and (I-D) fail at the same time, when (I-D) fails, using the same reasoning as in the previous case, the only available rerouting path is I-J-O-K-G-F-E-C-D, according to [7]. Because we used F-G-K in the previous case of link (E-C)'s failure, we must also transfer the traffic through F-G-K for this case to avoid conflict. Because both failures occur at the same time, the traffic will be transferred indeterminately between C and I; therefore, the traffic cannot be rerouted in this situation. That property is why [7] affirms that there is no rerouting scheme without conflict for destination D in this configuration.

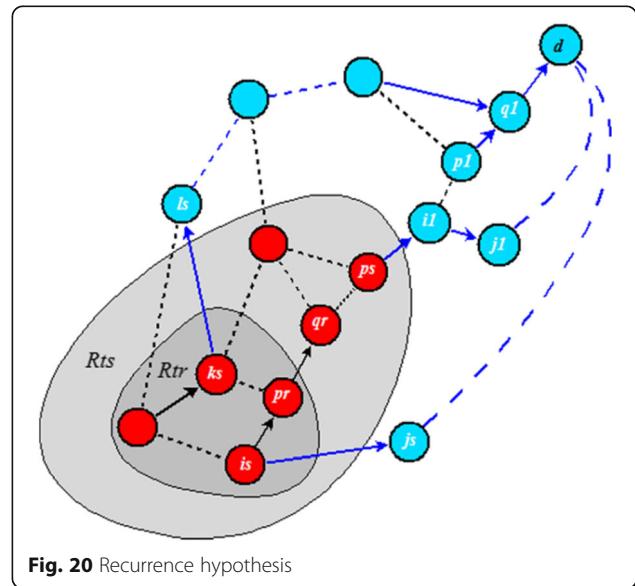
Now, consider our rerouting scheme using multiple planes. For the same configuration example above, our rerouting solution is shown in Fig. 18. In this figure, virtual nodes E and C are hosted by physical node EC; virtual nodes O and G are lodged by physical node OG. The network topology with unsolvable conflict is located in a virtual plane.

Let us transpose the topology of Fig. 17 into the physical plane as illustrated by Fig. 18.

Assuming that the nodes which detect failures are nodes E and I in the case of simultaneous link (E-C)'s and (I-D)'s failures and considering the topology's heterogeneity in the virtual networks, node E detects that a traffic redirection through path F-G-K-O-J will be deviated on node I and cause a cyclic problem. To solve that problem, we use another plan for traffic coming from both link failures. We will choose paths E-EC-D'-D and I-I'-D'-D for link (E-C)'s and (I-D)'s failure restoration. Thus, our approach can solve unsolvable conflicts presented in [7] by making use of multiple planes.

8 Conclusion and future work

Our aim in this paper was to propose a rerouting approach to handle the single link node failure and simultaneous multiple link failure problems in a network of

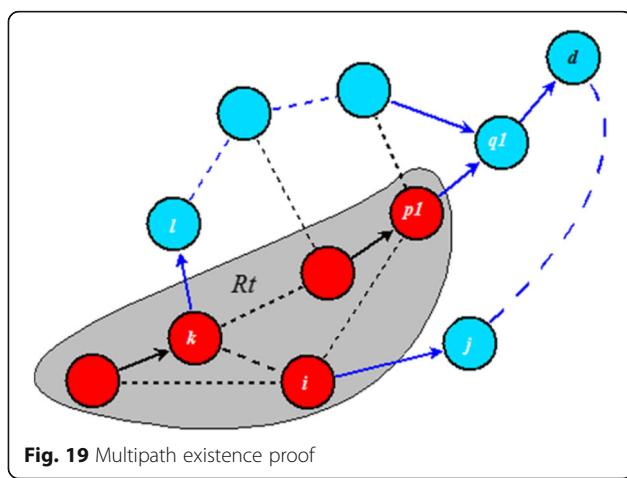


switches in the context of network virtualization. We proposed a conflict-free rerouting scheme that can ensure that, whatever the case of link or node failure, traffic will be rerouted to the destination. The proposed method is based on local reaction of nodes placed at the extremities of the failed link, whereas the other nodes need not know about the failure or take any particular action. Thus, the implementation is particularly easy. The flow splitting strategy used when there is insufficient spare capacity on links helps to reduce additional capacity added to the network. We proved that there exists a restoration scheme without conflict in the network and provide a mathematical model that permits calculation of the rerouting scheme with optimization of the sum of additional capacities needed for one virtual plane. We also proposed a rerouting solution using several planes to solve cases of potentially unsolvable conflicts when we use only one plane. Further work will address congestion management into the nodes implied in the rerouting and routing table updates without disturbing the network.

9 Appendix

The following is a formal proof of theorem 1. stated in Section 5. This proof is similar to the one presented in [7], with the difference that we are showing the existence of multiple paths rather than one path as in [7].

Consider a routing tree Rt to a destination d as shown in Fig. 19; then d is the sink of Rt . The dotted links represent the possible existence of nodes between nodes connected through that link. Let (p_1, q_1) be an arc of Rt . We assume that this arc fails and that we must find a rerouting scheme without conflict. We note that (q_1, p_1) does not belong to Rt , which means that, for a given



destination, the link failure problem can be treated as an arc failure. Without loss of generality in this proof, we will consider the problem of arc failure (p_1, q_1). p_1 is the sink of a sub-tree Rt_1 whose nodes are coloured in red. The other vertices in the tree are coloured in blue. We assume that all vertices are part of the tree Rt and that this assumption is true for any destination d .

Based on [7], there exists a rerouting path connecting both the red part and the blue part. Inside the red part of sub-tree (p_1, q_1), under the assumption of two-link connectivity, there are at least two paths going from any node of that red part to an arc connecting both the red and blue parts. Thus, if traffic splitting is performed on any of those nodes, it will be possible to reroute the flow through at least two different paths until it reaches destination d . In other words, two paths μ and ν exist from p_1 that visit some vertices of Rt_1 and connect a vertex of Rt_1 , which is red, to a blue vertex. This connection can be done through arcs (i, j) and (k, l) offering sufficient spare or adequate additional capacity. These arcs act as a bridge between the red and blue part. For a red vertex of Rt_1 affected by the failure, the descended traffic will first follow paths μ and ν inside the red part to p_1 . It will then follow the original routing tree from p_1 to i or k and use arc (i, j) or (k, l) as a bridge to reach destination d through blue vertices located in the blue part of the routing tree. Due to dimensioning issues, traffic can also be rerouted into the blue part over at least two alternative paths until destination d . According to the rerouting choices, the rerouting paths associated with the link (p_1, q_1) failure in the red and the blue parts are without conflict.

Consider the $n-1$ arc failures of the tree; we choose different arcs (i, j) to connect the red part to the blue part. First, we prove the existence of a bridge connecting both parts; second, we demonstrate the absence of conflict between different paths. The arcs of the tree are numbered in decreasing order as we approach sink d . We choose arcs (i, j) and (k, l) in successive order of increasing numbers. Let (pr, qr) be an arc under consideration. We assume that we have chosen arc pairs $((i_1, j_1), (k_1, l_1)), ((i_2, j_2), (k_2, l_2)), \dots ((i_{r-1}, j_{r-1}), (k_{r-1}, l_{r-1}))$ for rerouting. pr is the root of tree Rt_r . We consider two cases. The first case is with arc (ps, qs) as the failure, and $s < r$. In this case, arcs (i_s, j_s) and (k_s, l_s) have their extremities i_s and k_s inside tree Rt_s , and their extremities j_s and l_s are out of trees Rt_s and Rt_r , which are included in Rt_s (see Fig. 16). Then, we choose arc (i_s, j_s) as arc (i, j) and (k_s, l_s) as arc (k, l) for tree Rt_r . In the second case, there is no rerouting arc with this property. We choose any arcs (i_r, j_r) and (k_r, l_r) that connect Rt_r to its complement. Each of these cases offers at least two path possibilities in the red part of the network.

Let us prove the absence of conflict in our rerouting scheme. By recurrence of the number of rerouted arcs, let us assume that we have already rerouted $n-1$ arcs in the tree. Each rerouted arc has generated at least two rerouting paths. By the recurrence hypothesis, there is no conflict for the first $n-1$ reroutings. We must verify that the n^{th} rerouting also has no conflict with the first $n-1$ reroutings.

There is no conflict by construction concerning the rerouting of the outside part of tree Rt_r , which is the part in common with the classical routing. Although this n^{th} rerouting uses the same arc as the previous ones do, in this part, the rerouting will follow the same path until destination d ; therefore, there is no conflict. A conflict could occur if the splitting strategy is located in the blue part of the network. We verify no conflict exists for the part in which it goes in the opposite direction of the tree, which verifies that in the two cases cited above, there is no conflict. In the first case, when we have chosen arcs (i_s, j_s) and (k_s, l_s) in tree Rt_r , there was no conflict in that part of the tree because Rt_r would use the same arcs (i_s, j_s) and (k_s, l_s) as the passing bridges between its red part and its blue part (see Fig. 20). In the second case, the part climbing up the tree has nothing in common with the other rerouting arcs. In this case, there would exist (i_s, j_s) and (k_s, l_s) . Therefore, there is no conflict in this case. We can conclude that the property remains true to the order n . The absence of conflict in our rerouting scheme can be proved by reoccurrence.

Acknowledgements

We thank the anonymous reviewers whose valuable comments and suggestions have significantly improved the presentation and the readability of this work.

Authors' contributions

JFM suggested this work. VKT and YFY carried out analysis and performed the experiments. VKT and YFY wrote the first draft of this work and worked for the revised version. JFM revised the first draft and worked on the revised version. In addition, all authors read and approved the work.

Authors' information

VKT and YFY are with the department of Mathematics and Computer Science of the University of Dschang, Cameroon. JFM is with the Computer Science Lab. MIS of the university of Picardie Jules Verne, Amiens, France.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon. ²Computer Science Lab-Mis, University of Picardie Jules Verne, Amiens, France.

Received: 13 October 2017 Accepted: 25 April 2018

Published online: 18 June 2018

References

- Chowdhury NM, Boutaba R. Network virtualization: state of the art and research challenges. *IEEE Commun Mag*. 2009;7:20–6.
- Alkmim GP, Batista DM, da Fonseca NLS. Mapping virtual networks onto substrate networks. *J Internet Serv Appl*. 2013;4:3. <https://doi.org/10.1186/1869-0238-4-3>.
- Bays LR, Oliveira RR, Barcellos MP, Gaspary LP, Mauro Madeira ER. Virtual network security: threats, countermeasures, and challenges. *J Internet Serv Appl*. 2015;6:1. <https://doi.org/10.1186/s13174-014-0015-z>.
- Cheng X, Su S, Zhang Z, Shuang K, Yang F, Luo Y, Wang J. Virtual network embedding through topology awareness and optimization. *Comput Netw*. 2012;56:1797–813.
- Cheng X, Su S, Zhang Z, Wang H, Yang F, Luo Y, Wang J. Virtual network embedding through topology-aware node ranking. *ACM Comput Commun Rev*. 2011;41:38–47.
- Fernandes NC, Moreira MD, Moraes IM, Ferraz LH, Couto RS, Carvalho HE, Campista ME, Costa LH, Duarte OC. Virtual networks: isolation, performance, and trends. *Ann Telecommun*. 2011;66:339–55.
- Pham TS, Lattmann J, Lutton JL, Valeyre L, Carlier J, Nace D. A restoration scheme for virtual networks using switches: International Workshop on Reliable Networks Design and Modeling. USA: IEEE Press; 2012. p. 800–5.
- Pham TS. Autonomous management of quality of service in virtual networks: PhD Thesis. Compiègne: the university of Technology of Compiègne; 2004.
- Atlas AK, Zinin A (2008) Basic specification for IP fast-reroute: loopfree alternates. <https://tools.ietf.org/pdf/rfc5286.pdf>. Accessed 20 July 2017.
- Bryant S, Shand M, Previdi S (2011) IP fast reroute using not-via addresses. <https://www.ietf.org/proceedings/62/slides/rwg-w3.pdf>. Accessed 10 July 2017.
- Ho K-H, Wang N, Pavlou G, Botsaris C. Optimizing post-failure network performance for IP fast reroute using tunnels. In: Proceedings of the 5th international ICST conference on heterogeneous networking for quality, reliability, security and robustness, article no 44. Hong Kong: ACM digital Library; 2008.
- Kvalbein A, Hansen A, Cicic T, Gjessing S, Lysne O. Fast IP network recovery using multiple routing configurations, vol. 2006: Proceedings IEEE INFOCOM; 2006. <https://doi.org/10.1109/INFOCOM.2006.227>.
- Zalesky A, LeVu H, Zukerman M. Reducing spare capacity through traffic splitting. *IEEE Commun Lett*. 2004;8:594–6. <https://doi.org/10.1109/LCOMM.2004.833800>.
- Wang J, Nelakuditi S. IP fast reroute with failure inferencing. In: ACM proceedings of the 2007 SIGCOMM workshop on internet network management. Kyoto: ACM Digital Library; 2007. p. 268–73.
- Kang X, Chao HJ. IP fast rerouting for single-link/node failure recovery. In: BROADNETS 2007, fourth international conference on broadband communications: Networks and Systems. USA: IEEE Press; 2007. p. 142–51.
- Kang X, Chao HJ. IP fast reroute for double-link failure recovery. In: Proceedings of the 28th IEEE conference on global telecommunications. Piscataway: GLOBECOM; 2009. p. 1035–42.
- Sgambelluri A, Giorgetti A, Cugini F, Paolucci F, Castoldi P. Openflow based segment protection in ethernet networks. *J Opt Commun Netw*. 2013;5:1066–75. <https://doi.org/10.1364/JOCN.5.001066>.
- Staessens D, Colle D, Pickavet M, Demeester P. A demonstration of automatic bootstrapping of resilient openFlow networks. In Poceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM 2013). Ghent: IEEE Xplore Digital Library; 2013. pp. 1066–7.
- Sharma S, Staessens D, Colle D, Pickavet M, Demeester P. OpenFlow: meeting carrier-grade recovery requirements. *Comput Commun*. 2013;36:656–65. <https://doi.org/10.1016/j.comcom.2012.09.011>.
- Kamamura S, Shimazaki D, Hiramatsu A, Nakazato H. Autonomous IP fast rerouting with compressed backup flow entries using OpenFlow. *IEICE Trans Inf Sys*, pp. 2013;96:184–192.
- Yu M, Yi Y, Rexford J, Chiang M. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):17–29.
- Veeramany J, Venkatesan S, Shah JC. Effect of traffic splitting on link and path restoration planning. In: The global telecommunications conference, 1994 IEEE GLOBECOM, vol. 3: Communications: The Global Bridge. USA: IEEE Press; 1994. p. 1867–71.
- Fischer S, Kammenhuber N, Feldmann A. REPLEX: dynamic traffic engineering based on wardrop routing policies. In: Proceedings of the of the ACM CoNEXT'06. Lisboa: ACM Digital Library; 2006. p. 1–12.
- OpenFlow multipath proposal. http://openflowswitch.org/wk/index.php/Multipath_Proposal. Accessed 26 Oct 2015.
- Cao Z, Wang Z, Zegura E. Performance of hashing-based schemes for internet load balancing. In: Proceedings of INFOCOM'00, vol. 1. Israel: Tel Aviv; 2000. p. 332–41.
- Prabhavat S, Nishiyama H, Ansari N, Kato N. On the performance analysis of traffic splitting on load imbalancing and packet reordering of bursty traffic. In: IEEE international Conference on Network infrastructure and digital content, IC-NIDC, USA: IEEE Press. 2009. p. 236–40
- Bennett JC, Partridge C, Shectman N. Packet reordering is not pathological network behavior. *IEEE/ACM Trans Networking*. 1999;7(6):789–98.
- Laor M, Gendel L. The effect of packet reordering in a backbone link on application throughput. *IEEE Netw*. 2002;16(5):28–36.
- Leung KC, Li VO, Yang D. An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges. *IEEE Trans Parallel Distrib Syst*. 2007;18(4):522–35.
- Kandula S, Katabi D, Sinha S, Berger A. Dynamic load balancing without packet reordering. *ACM SIGCOMM Comput Commun Rev*. 2007;37(2):51–62.
- Adisesu H, Parulkar G, Varghese G. A reliable and scalable striping protocol. *ACM SIGCOMM Comput Commun Rev*. 1996;26(4):131–41.
- Partridge C, Milliken W. Method and apparatus for byte-by-byte multiplexing of data over parallel communications links: Patent number: 6160819. Assigned to GTE Internetworking Incorporated, December 2000, Cambridge, Massachusetts, USA.
- Boyd S, Vandenberghe L. Convex optimization, vol. 34. UK: Cambridge university press; 2004.
- Khana AR, Bilalb SM, Othmana M. A performance comparison of network simulators for wireless networks. USA: Cornell University, Library. 2013;arXiv: 1307.4129.

Submit your manuscript to a SpringerOpen® journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com

A Conflict-Free Routing Tables Update Method for Persistent Multilink and Node Failures in SDN Architectures

¹Yannick Florian Yankam, ²Jean Frédéric Myoupo and ¹Vianney Kengne Tchendji

¹Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon

²Computer Science Lab-MIS, University of Picardie Jules Verne, Amiens, France

Article history

Received: 21-12-2018

Revised: 31-12-2018

Accepted: 11-03-2019

Corresponding Author:

Jean Frédéric Myoupo
Computer Science Lab-MIS,
University of Picardie Jules
Verne, Amiens, France
Email: jeanmarc.myoupo@u-picardie.fr

Abstract: The large-scale network management abilities of centralized architectures, such as Software-Defined Networking (SDN), are attracting increasing interest from major computer networking companies. In this architecture type, the nodes follow the rerouting rules predefined by the control plane in the controller. However, when a link or node failure becomes persistent over the time, the controller must recompute the routing and rerouting rules and update the relevant nodes to maintain an acceptable quality of service (QoS). In this study, we propose a conflict-free mechanism for updates of routing and rerouting tables of nodes by the controller. This mechanism works without disrupting the current traffic if both persistent multilink and node failures occur. We describe an efficient strategy for choosing the nodes to update and define an update scheme for these nodes. We show through the simulations of our updating scheme on various networks that our strategy improves QoS by reducing packet routing delays and the data loss rate in case of persistent multilink and node failures in the network. We also compare our results to those of several existing studies.

Keywords: Network Virtualization, Software-Defined Networking (SDN), Centralized Architecture, QoS, Network Recovery

Introduction

For a long time, the difficulty of managing largescale network infrastructure (addressing, bandwidth, throughput, etc.) has been increasing in computer networks. The adoption of network virtualization technology has made this problem increasingly urgent and timely. The reason is that network virtualization allows easy creation of virtual networks using the virtual components built from an already existing physical IP network. However, in recent years, the Software-Defined Networking (SDN) approach has been presented as the most suitable solution for such a problem (Farhad et al., 2015; Hu et al., 2014). In this technology, the control plane (Hu et al., 2014) is decoupled from the data plane. The control plane is the part of the network that defines the network management policies, while the data plane merely forwards data according to the rules defined by the control plane. The latter is centralized within the main equipment called the controller and the data plane (Hu et al., 2014) is kept on network devices (e.g., switches and routers). Such decoupling of planes provide flexibility and programmability, satisfying the cloud automation needs and making it easier to add new services

in communication networks (Azodolmolky et al., 2013). The controller that hosts the control plane defines network management policies, e.g., the routing rules and integrates them inside network devices. When a failure occurs in the network, the devices forward the packets according to the rules defined in their flow tables (Rothenberg et al., 2012; Pham, 2014; Papan et al., 2017) in advance by the controller. This is the IPFRR (IP Fast ReRoute) scheme known for its fast rerouting capabilities.

However, when a failure has been deemed persistent, these rules become obsolete and no longer allow for a good QoS in the network. The network QoS mostly suffers in case of multilink and node failures. A failure is determined to be persistent if its duration is more than ten minutes (Pham, 2014). In this case, the preceding routing rules are no longer optimal and become obsolete. Then, it is necessary to recalculate the routing paths and update the routing tables of the devices. The most important challenges to be overcome by the controller in this failure scenario are the following:

- Construct new routing and rerouting rules based on the existing rules, to handle the failures
- Select the nodes to update

- Choose the efficient update order that will be applied to the nodes; in particular, it needs to be decided whether all the involved nodes are to be updated simultaneously or sequentially
- Decide whether to send all the computed rules at the same time; this challenge concerns the packet size and the maximum transmission unit (MTU) value of the network's interfaces; in large-scale networks, the packet size of the updates can increase easily with the number of nodes

Related Studies

A review of the routing tables update problem in the literature yields few relevant ideas that have already been stated in basic routing protocols, such as Open Shortest Path First (OSPF) (Moy, 1998), Routing Information Protocol (RIP) (Perkins *et al.*, 2003; Hedrick, 1988) and Enhanced Interior Gateway Routing Protocol (EIGRP). Perhaps this is why, to the best of our knowledge, many of the centralized (Muruganathan *et al.*, 2005), distributed (Moy, 1998; Perkins *et al.*, 2003) and even hybrid (Rothenberg *et al.*, 2012; Lim *et al.*, 2008) rerouting schemes provided in computer networks simply define the rerouting paths calculation and do not specifically address the routing tables update challenge. Nevertheless, the above distributed protocols are not individually efficient in centralized architectures, such as SDN using the OpenFlow protocol (Pham, 2014).

Lim *et al.* (2008) proposed a rerouting scheme based on a tree topology computation in Wireless Mesh Networks (WMN). In that tree-based routing scheme, numerous messages are exchanged between the root node and its child nodes when a routing table update is needed in case of a link or a node failure.

Moreover, each node that needs to update its routing table must send a Route Request (RREQ) message to the root first; the root replies with an update message packet called Route Set (RSET) (its structure is shown in Fig. 1). In a large-scale network, this approach could be very harmful.

In a centralized SDN architecture, when a persistent link failure has been identified, Pham (2014) suggests an update strategy based on an IPFRR (Papan *et al.*, 2017) mechanism that avoids looping. In the cited study, the authors propose updating the nodes from the destination of a routing tree to the node that detected the failure. However, this scheme has the following limitations:

- Overloading of the rerouting path, which may affect the handling of other link failures and load balancing in the network
- Involving nodes for which an update is not necessarily needed; this increases the conflict risks to be managed in the rerouting path
- Only stating the order of updating the routing tables from one node to another, but not stating how the topology will be recomputed and how the update's information will be sent by the controller

Our Contribution

The main motivation of this paper is to improve the computer network's QoS by providing solutions to the drawbacks cited above for the approaches of (Pham, 2014; Lim *et al.*, 2008). The contribution of this paper is the continuation of our work that previously appeared in Myoupo *et al.* (2018). More precisely, our contribution is a method of updating routing and rerouting tables of nodes by a controller. In contrast to the approach in Myoupo *et al.* (2018) in which only a single link failure is considered, this new approach also considers multilink and node failures; we describe it through three aspects:

- Determination of nodes involved in the update process: We show that by updating certain particular nodes, we can obtain lower packet routing delays in case of a persistent link and node failure than in case of a transient failure
- Construction of an update packet structure: We present a packet structure in the form of a vector of lists, inspired by the Link State Update (LSU) and Link State Advertisement (LSA) packets' structure of the Open Shortest Path First (OSPF) protocol (Moy, 1998). The structure of the update packet must be considered carefully to reduce the size of update packets which can facilitate easier forwarding of the updated routing information over the network
- Definition of a nodes' update scheme: We opt for both simultaneous and sequential updates of the involved nodes. Simultaneous update refers to particular nodes called critical nodes that are not directly linked by links; sequential update refers to the nodes on the original rerouting path used by the IPFRR strategy applied in Pham (2014) to deal with a transient failure. This approach helps reduce the rerouting paths' overloading

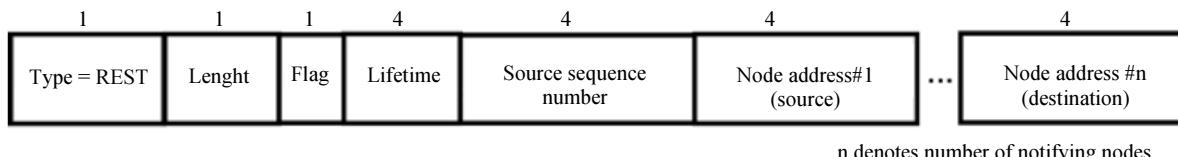


Fig. 1: Sample line graph using colours that contrast well both on screen and on a black-and-white hardcopy

The remainder of this paper is organized as follows. In section 2, we propose several hypotheses related to this study. Section 3 presents our update strategy for a single persistent link failure. Section 4 presents our update strategy for a persistent and non-simultaneous multilink failure. Section 5 presents our approach to the case of a virtual node failure. Our solution for the case of a physical node failure is presented in section 6. Section 7 describes numerical results of simulations. Finally, section 8 concludes the paper.

Hypotheses

In this work, we assume that the network is represented by a graph with bidirectional links. Each edge is associated with a weight that represents the bandwidth or the flow on that edge. The weight of nodes is not considered and it is assumed that for any pair of nodes, there are at least two disjoint paths, making it possible to connect them.

We assume that there is already a rerouting scheme used for handle link and node failures. This scheme is based on Pham (2014), which exploits a shortest path tree called the nominal routing tree (Fig. 2a). It assumes that in case of a link failure, only one node detects the failure (node S in Fig. 2b) and reroutes packets according to the rules described in its routing table, as shown in Fig. 2b (black path). The node that detects failure is located in the red nominal routing tree that hosts the failure and the destination node (node D in Fig. 2b) of the original nominal routing tree is located in the blue nominal routing tree.

In case of node failure, all the links connected to that node fail. Then, multiple nodes detect the failure and each reroutes packets according to the rules

provided for a single link failure. We consider a link to fail when flows can no longer use that link. A switch detects a link failure as unavailability of the port connected to that link.

Our Loop-Free Routing Tables Update Method using a Vector of Lists of Triplets in Case of a Single Link Failure

Here, we recall the results of Myoupo *et al.* (2018) that will be useful in the following.

Let us consider a network represented by graph $G(N, L)$, where N represents the set of nodes and L is the set of links. Let us consider the following definitions before they are used later:

- G_r : The red part of the network that hosts a failure
- G_b : The blue part of the network that hosts the destination node of a nominal routing tree
- N_f : The node that detects a link failure when it occurs
- *border node*: A node of the red part connected directly to a node of the blue part
- *critical node*: A border node nearest to the node that detects a transient failure

Our Critical Nodes' Detection Mechanism

We consider a persistent link failure (p, q) (Fig. 3). Let $|N| = \text{Cardinal}(N)$ be the number of nodes and $|L| = \text{Cardinal}(L)$ be the number of links. We also assume that node D is the destination of the flows of $G(N, L)$. The link failure (p, q) generates the subgraph G_r , which is a routing tree with node p as the root (Fig. 3). For each node of G_r , there is a path linking it to p because of the tree topology.

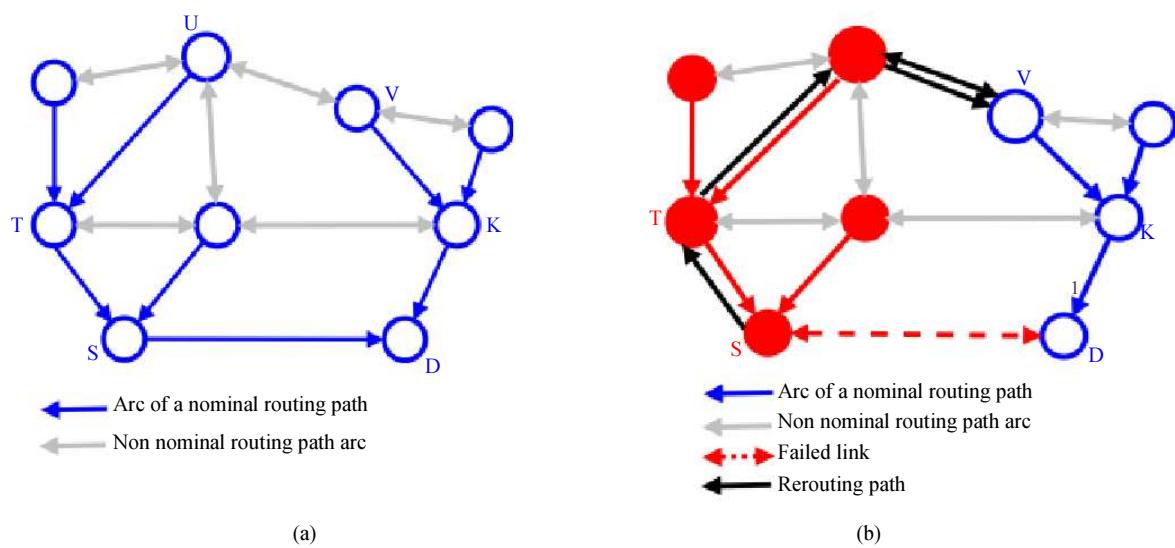


Fig. 2: Rerouting scheme used for link failure. D indicates the destination (a) Nominal routing tree (b) Rerouting path

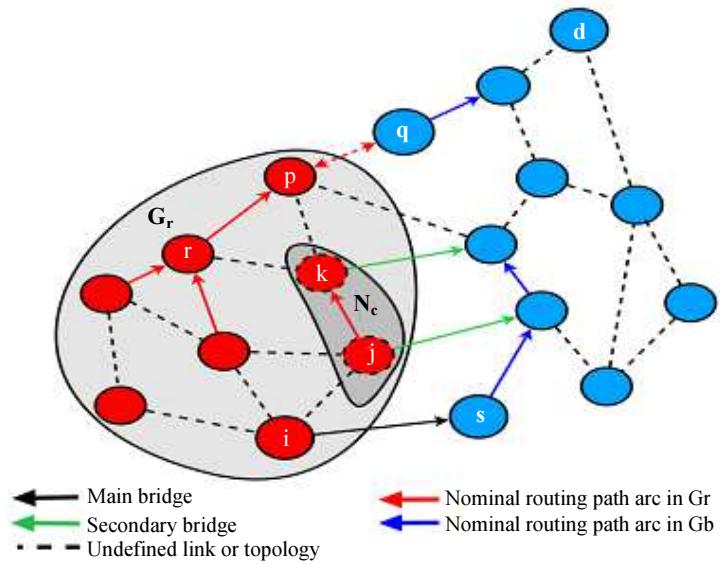


Fig. 3: Our mechanism for detecting the nodes to be updated

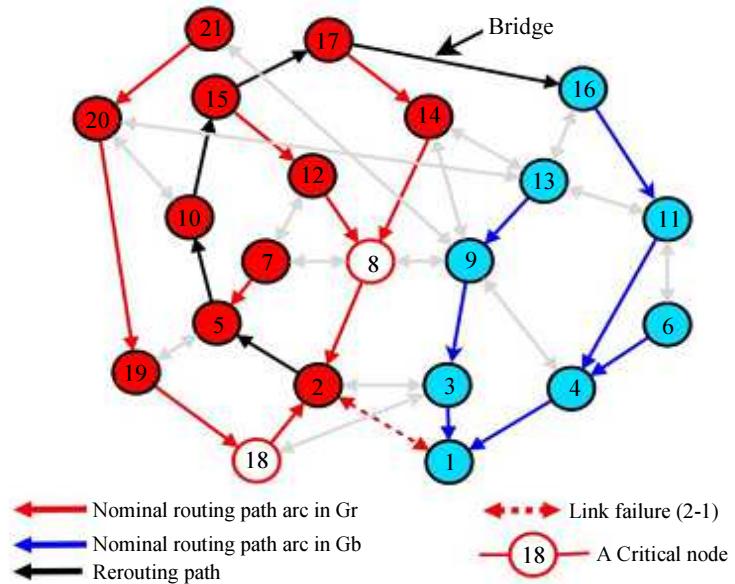


Fig. 4: Our mechanism of detecting the nodes to be updated

The dotted links indicate the possible presence of a topology. According to Pham (2014), there is at least one subgraph rooted in node r , which contains a set of nodes r_i and connects an extremity i of G_r to node s of G_b ; this is the bridge (i, s) . Similarly, according to the configurations, there are subtrees N_c of G rooted in nodes k_i ($i = 1, 2, \dots$) and composed of nodes $k_j \neq i$ that directly connect G_r to G_b .

Our detection strategy uses only one bridge (instead of several) from each branch of the nominal tree G_r to retain the network structure. The search process starts

from node $p = N_f$. We look for the first node of each routing tree's branch that offers a bridge to reach G_b . That node k is a critical node. To reduce the loop risk in the routing, only these critical nodes among the border nodes will be updated. The exploration of a tree branch stops when a critical node has been found. For instance, in the case of Fig. 4, the critical nodes are 8 and 18.

General Update Principle from one Node to Another

While the critical nodes are located in different tree branches, they are all independent from each other; then,

the controller can update those critical nodes simultaneously through packet-out messages, without the loop risk. In addition to these critical nodes, the nodes of the rerouting path used to handle the preceding transient failure also need to be updated because of rerouting. As to the nodes of the rerouting path, the update will be performed gradually, starting from the extremity node i of the bridge located in G_r towards node p that detected the link failure.

As the routing tables of the nodes in the rerouting path are used for transferring the rerouted flows, the update for these nodes will consist of applying these rerouting configurations in their routing tables as the new routing rules. The critical nodes' routing tables will be updated to allow for these critical nodes to use a secondary bridge; the rerouting paths of these nodes must also be rebuilt by considering these new routing tables' configurations. Consequently, the rerouting strategies previously developed by the controller will be recomputed by this method and reinstated in the various nodes of G_r . Otherwise, the routing tree would neither remain nominal nor be optimal, which would increase the packet routing delays.

Internal Updates of Nodes in the Rerouting Path

The internal updates of nodes are performed using a message containing a vector of lists of triplets that contains the updated data provided by the controller. This message's structure is similar to that of LSU messages of the OSPF protocol (Fig. 5). Each entry in the update vector is a list of triplets (a, b, c) , where a represents the entry gate of the flow to be modified, b is the new exit gate of this flow and c is the exit gate that

will be used for forwarding the remainder of the message to the next node in the vector. In case of a switch, c will be a MAC address and in the case of a router, it will be the next hop's IP address. For the nodes that need to update many entries in their routing table, only the last triplets in their update lists will have the value $c \neq 0$. While the update message can easily expand with the number of nodes to be updated, considering the MTU value, the update packet of the nodes will be fragmented into smaller packets by the controller before being transmitted. Each fragment will be assigned to its appropriate group of nodes, as shown in Fig. 6. Each group of nodes of the rerouting path that receive an update message use it only when it receives a permission message from the controller or from the preceding node in the rerouting path; to achieve this latency, the SDN switches or routers are made of agents, as it is the case in Pham (2014) with filters. Flow management, analysis of the ports' status and initiating the appropriate actions as specified by the controller are some features of these agents. When a node wants to use the update message, it reads the field k 's value, recovers its list of triplets, performs its updates, increments the value of k and returns the update message to the next node in the vector. The last triplet list has the value of c equal to 0; this signifies the end of update message transmission between the nodes of the rerouting path. When each node in a group receives its update triplets list, it checks if $c = 0$ in the last triplet and in such a case, updates its routing table and destroys the update message; otherwise, it forwards this update message to the next node.

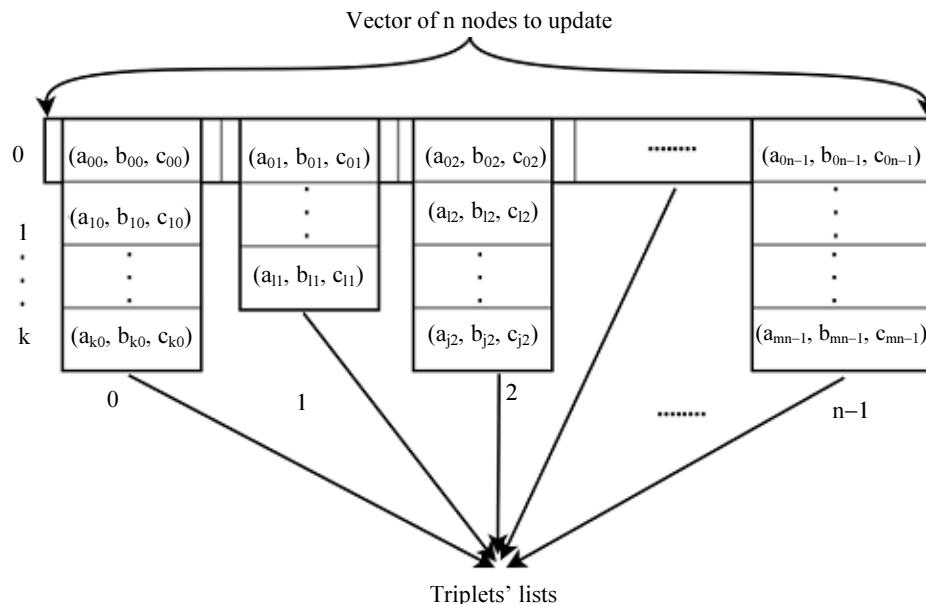


Fig. 5: Update structure of a vector of lists of triplets

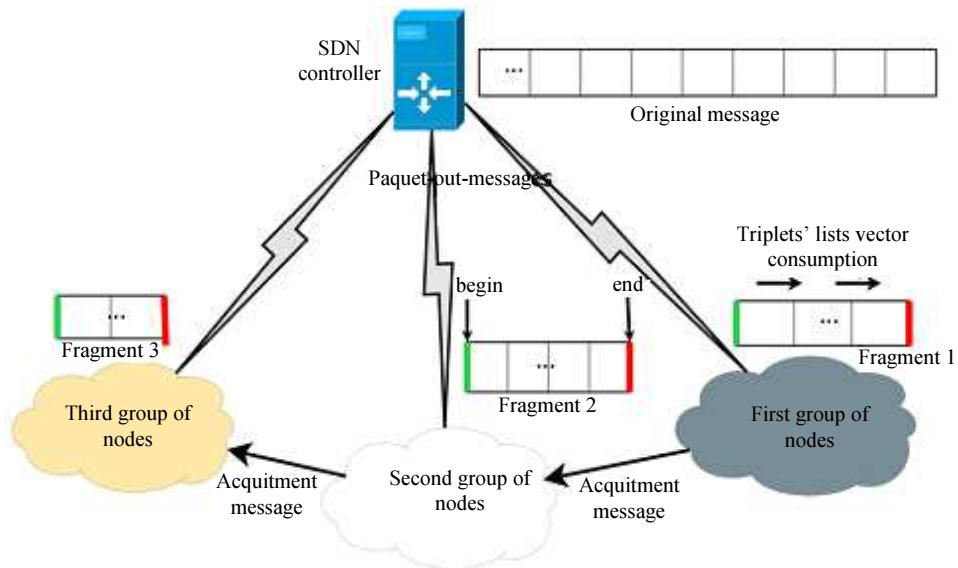


Fig. 6: Fragmentation of the update message

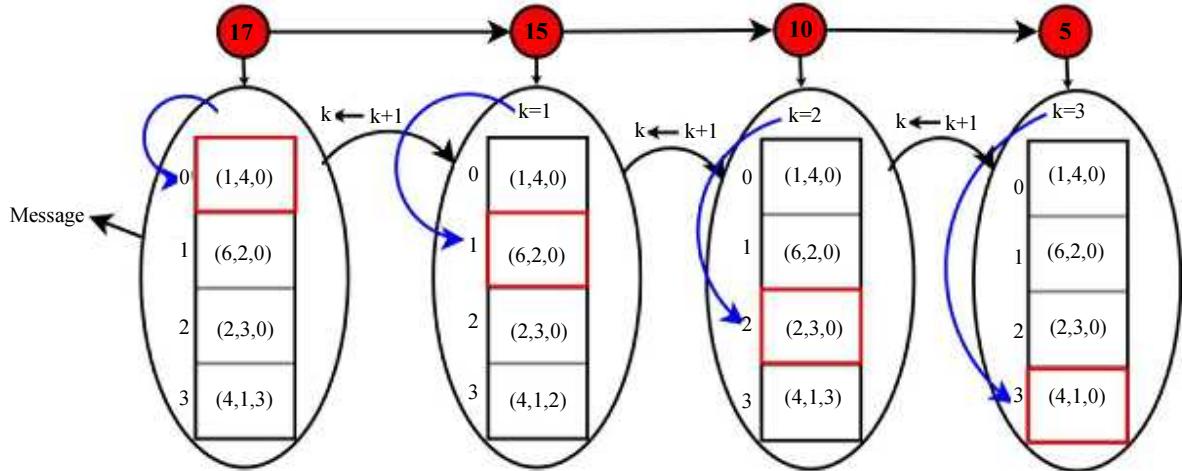


Fig. 7: Internal updates of nodes using a vector of lists of triplets

The use of this message structure is illustrated for the successive updates of a group of nodes 17, 15, 10 and 5 in Fig. 7. In that figure, node 5 is the last one to be updated in the group.

Reconstruction of Rerouting Tables' Rules using the New Routing Configurations

While rerouting rules are computed based on the new routing table's configurations, rerouting rules of each updated node must also be updated. Using the approach of Pham (2014) on a virtual nominal routing tree built by the controller, the controller computes the new rerouting rules; during this step, the network uses the prior rerouting rules as in the Cisco routers, resulting in poor QoS. To minimize the latencies during computations, the operations for determining the critical nodes and

computations of the new routing and rerouting rules must all be done in advance in the controller before initiating any tables' update process. For the same reasons, the updates to the routing tables and rerouting tables in each node can be performed at the same time assuming that the controller has already provided the rules.

Our Routing Tables Update Approach for non-Simultaneous and Persistent

Multilink Failures Multiple link failures are called non-simultaneous when they occur sequentially. Each of these failures can persist over time (for instance, more than 10 min in Pham (2014)); then, they are deemed persistent failures. Persistence of each failure will be detected by the controller consecutively. In this section, we propose several

resolution approaches that could be considered for updating the routing tables and explain our strategy.

First Solution: Consecutive Update of Nodes as Link Failures are Deemed Persistent

This solution suggests solving persistent link failure cases as they are detected by the controller. Each detected case is solved by applying the proposed approach to the persistent single link failure case. This solution is inspired by Pham (2014), which is related to the rerouting rules' computation for cases of simultaneous and transient multilink failures.

Figure 8 illustrates this solution for persistent and nonsimultaneous failures of two links. In this figure, let us assume that the link failure (3, 1) is deemed persistent before (8, 2) is by the controller.

Then, the controller immediately recalculates the rerouting path to use the path 3-9-4 to reroute the flow and launches the update of nodes 3 and 9; this update introduces the new configuration presented in Fig. 8c. The controller starts the rerouting path computation (path 8-14-13) directly for link failure (8, 2) by considering this new configuration and sends the update message once the computation is complete.

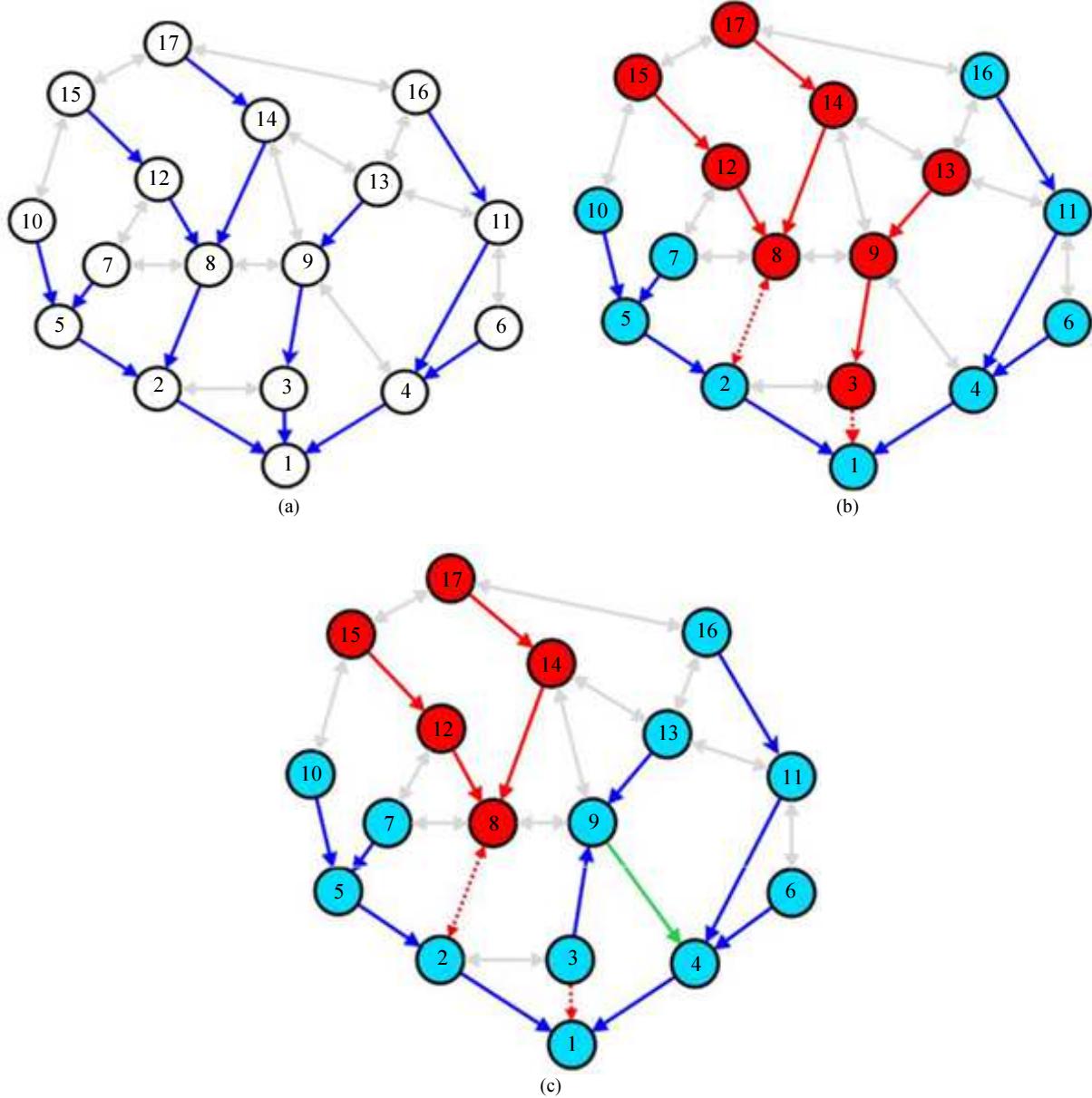


Fig. 8: Updates' structure of a vector of lists of triplets (a) Original nominal routing tree with node 1 as the destination (b) Persistent non-simultaneous multilink failure (c) New configuration after persistent failure management

Second Solution: Synchronized Update of a Node's Routing Tables as Persistent Link Failures are Gradually Detected

Consider a network subject to a set of nonsimultaneous link failures. The main drawback of the first solution is the difficulty of managing multiple update messages for different link failures when they arrive simultaneously at the same node. To solve this question, we prove the following theorem:

Theorem 1

For any pair of non-simultaneous link failures, there exists at least one common node in subgraphs G_r and G_b corresponding to these two failures.

Proof

Let VN be a virtual network represented by a graph $G(|N|, |L|)$ where $|N|$ is the number of nodes and $|L|$ the number of links. Let d be the destination of the flows of a Nominal routing tree of G. Let (l_1, l_2) be a pair of link failures. Let n_1 be the node that has detected l_1 and n_2 the one that has detected l_2 . l_1 subdivides G into two subgraphs G'_1 and G'_2 , with $n_1 \in G'_1$ and $d \in G'_2$ such that $\bigcup_{i \in \{1,2\}} G'_i = G$. Similarly, l_2 divides G into G''_1 and G''_2 such that $n_2 \in G''_1$ and $d \in G''_2$.

Let us consider $G'_1 \cap G''_1 = \emptyset$ and $G'_2 \cap G''_2 = \emptyset$. We have $d \in G'_2$ and $d \in G''_2 \Rightarrow d \in G'_2 \cap G''_2$; but $G'_2 \cap G''_2 = \emptyset$. So $d \in \emptyset$, which is absurd. Hence $G'_2 \cap G''_2 \neq \emptyset$. $G'_1 \cap G''_1 = \emptyset \Rightarrow n_1 \notin G''_1$ and $n_2 \notin G'_1$. But, the failure l_1 creates the subdivision $S_{1|G}$ with $G'_1 \subset S_{1|G}$ and $G'_2 \subset S_{1|G}$. Similarly, l_2 creates the subdivision $S_{2|G}$ such that $G''_1 \subset S_{2|G}$ and $G''_2 \subset S_{2|G}$. Since each failure induces a subdivision of the graph G into two subgraphs, we have: $S_{1|G} \subset S_{2|G}$ or $S_{2|G} \subset S_{1|G}$. Thus, $n_1 \in S_{2|G} \Rightarrow n_1 \in S_{1|G} \Rightarrow n_1 \in G'_1 \cap G''_1$ which is absurd because $G'_1 \cap G''_1 = \emptyset$.

Theorem 1 states the existence of common nodes in different subtrees generated by n non-simultaneous and consecutively persistent link failures. These common nodes are the most likely nodes to cause conflicts in the routing tables update' process between multiple messages. This theorem allows us to consider two approaches of solutions to this synchronization problem:

Approach 1

Wait for a packet-in message from the network before sending any update message for another link failure. When the updates' data for a given failure is being applied in the network, the controller waits to receive a packet-in message from the last node group to be updated, before sending the update packets for another failure. Considering Fig. 6 related to the updates' message fragmentation, the last group of packets is that of group 3, i.e., the packet-in message sent to the controller at the end of the update process for the

handled failure will come from this group. In this group, it is the last node to be updated that will send this message, e.g., node 5 of Fig. 7 in section 3. This node will know that it is the last one by examining the value of c that is zero in the last update triplet. This approach has the advantage of avoiding the cases of conflicts and loss of update data for different failures; however, it has the major drawback of increasing latency.

Approach 2

Assign a label referencing the priority level of update messages for each failure.

The controller adds into the update message for a given failure a priority order in the set of update messages related to other persistent detected link failures. This priority order is a field that is a date or an integer, the value of which is incremented each time a new persistent failure is detected. Hence, there is no need to know in advance the existence of any potentially persistent failures in the network to use this field. To avoid information loss, the controller must keep up to date internally the current value of this field. This approach helps avoid the assignment of the same label to two update messages concerning different failures.

When multiple update messages are encountered within the same node, the value of the priority field is used to respect the precedence order; this is done especially when it is necessary to simultaneously apply many conflicting update data to a common entry of the relevant node's routing table. Then, the message with the smallest priority value is used before other messages with larger values. Non-conflicting update data related to the same entries are applied simultaneously. This approach reduces latency by allowing multiple update messages of different failures to be forwarded simultaneously; the preceding solution does not allow this.

Approach 2 is what we propose for a node's update and synchronization problems in cases of non-simultaneous and persistent multiple link failures. This approach enhances that used for the case of a single persistent link failure proposed in section 3, with several adjustments:

- A field is added to the update message structure to manage update synchronization in the common network parts delimited by several failures
- We assume at least two persistent link failures in the network

Our approach provides the following advantages:

- Preserving the consistency of paths in the network
- Maximizing handling of non-simultaneous link failures through the update synchronization method
- Allowing us to treat the cases of simultaneous failures of two non-adjacent links to the same node

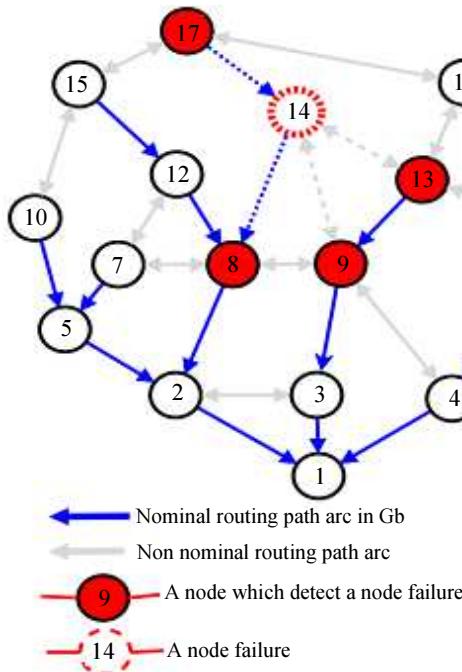


Fig. 9: Failure of node 14

The Persistent Virtual Node Failure Case

A node (router or switch) failure, whether physical or virtual, implies the impossibility of transferring data from any incoming port to an outgoing port of that node. This failure scenario directly leads to a simultaneous failure of several links. Figure 9 shows the failure of node 14 that induces failures of links (14, 17), (14, 13), (14, 9) and (14, 8).

We consider for the moment a single node failure in a virtual plane. Then, a node failure can be considered as:

- Simultaneous multiple link failures adjacent to the same node, as shown in Fig. 9
- Internal damage of the node

In both situations, it is impossible to transit flows from one input port of the equipment to another. The node failure's persistence is detected based on that of failures of links adjacent to it. For various reasons, the persistence of these failures can be detected and addressed by the controller in several ways:

Solution 1: Successive Detection of Persistent Adjacent Links' Failures

Persistence detection of multiple failed links is performed successively. In this case, the new rerouting paths are computed using the method of Pham (2014). The update mechanism used is approach 2 presented for the case of non-simultaneous multilink failures.

Solution 2: Simultaneous Detection of Persistent Link Failures Adjacent to the Same Node

In this case, the link failures that caused the node failure are all deemed persistent at the same time. According to the IPFRR strategy (Pham, 2014; Papan et al., 2017), the nodes that will detect the failure of node 14 are 17, 8, 9 and 13 (Fig. 9). These detector nodes will be the first to be updated to avoid the cycles in the rerouting; the next nodes are those of the rerouting path found by using the strategy of Pham (2014). The updates of the detector nodes will be primarily to divert flows for failed links to available links. For the case of Fig. 9 where node 1 is the routing tree destination, paths 17-1611-4-1, 8-2-1, 9-3-1 and 13-9-3-1 can be used to treat the link failures (14,17), (14,8), (14,9) and (14,13), respectively. The general update principle for this second solution is as follows:

1. The controller uses its global network view to detect the failed node from the list of failed adjacent links
2. For each persistent link failure detected, the controller builds the updates' messages based on our strategy proposed in section 2 to handle a single persistent link failure. A date field is included in each update message to manage synchronizations problems
3. THE controller sends the updates' messages to the nodes that detected the given node failure

Our Strategy for a Persistent Physical Node Failure

The network virtualization paradigm allows the sharing of a physical infrastructure via the creation of virtual networks that are hosted by this infrastructure (Mosharaf Kabir Chowdhury and Boutaba, 2010). Then, a physical node damage involves the failure of multiple virtual nodes belonging to the same plane or to different planes according to the mapping model that exists between the substrate network and the virtual networks being hosted (Nashid et al., 2016). Figure 10 shows a network virtualization environment including two virtual networks VN1 and VN2.

Nodes 3 and 4 of VN1 and node 5 of VN2 are all hosted within node 2 of the substrate network. When substrate node 2 fails, it directly induces the failure of nodes 3 and 4 of VN1 and node 5 of VN2; the physical and virtual links also fail at the same time.

In this type of configuration, the solutions proposed in Nashid et al. (2016) or Pham (2014) can be used for rerouting. In addition, various approaches to multilayer rerouting proposed in the framework of multilayer optical networks (Doumith, 2007) allow us to consider several routing tables update approaches:

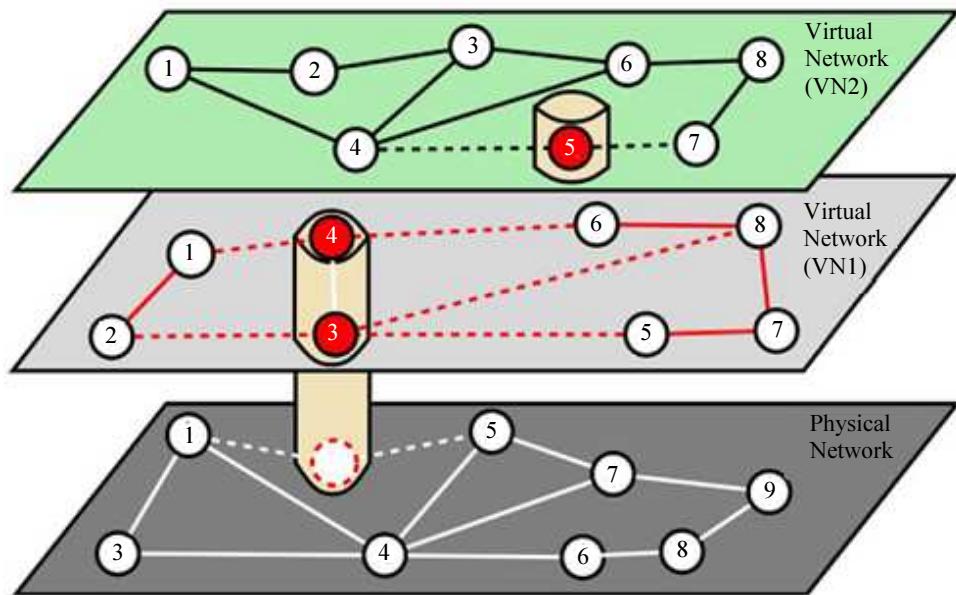


Fig. 10: Physical node failure

Solution 1: Uncoordinated Update

This solution uses our previously proposed strategy for virtual node failure in each virtual network. As a result, the controller triggers the update for each virtual plane affected by the failure of the physical node.

This approach suffers from an excessive protection bandwidth reservation that could decrease the network performance, especially in case of residual bandwidth deficiency in the network.

Solution 2: Coordinated Update

The aim is to define a sequential update process for the set of virtual networks affected by the substrate node failure. However, we first update the necessary nodes in the physical network (location of the physical node failure) before examining the hosted virtual planes: This strategy ensures the integrity of the substrate network, which affects the QoS in the network through its shared resources (bandwidth, throughput, etc). In the physical network, we can use our routing tables update scheme proposed for a virtual node failure.

Considering the dimensioning network problem and the virtual network mapping on the physical network, a physical node failure can lead to a simultaneous failure of multiple virtual nodes; e.g., this is the case for virtual nodes 3 and 4 of virtual network VN1 in Fig. 10. This failure's heterogeneity in the virtual networks makes the coordinated update process very difficult. Then, some update priority criteria for the affected virtual planes are needed to efficiently manage conflicts in the presence of multiple simultaneous persistent failures. These criteria are defined by the infrastructure provider

according to the intended objectives. In this paper, we define two priority criteria: The number of failed virtual nodes and the traffic weight. Hence, depending on the infrastructure provider's objectives, we can propose the following solutions for managing update for a set of affected virtual planes:

- Prioritize the virtual nodes of the network with the fewest failed virtual nodes. This approach has the advantage of minimizing the routing conflicts. In the case of Fig. 10, the nodes of virtual network VN2 present only one node failure, while VN1 is affected by two node failures. Then, the nodes of virtual network VN2 will be updated before those of VN1
- Focus on the virtual network with the highest traffic weight: The advantage of this approach is the customers' loyalty through a traffic-oriented QoS

Implementation and Simulation Results

The OMNet++ network simulator environment has been used to implement and evaluate our routing table's update scheme. This network simulator provides an extensive graphical user interface (GUI), intelligent support, short computation times and an effective implementation of large-scale networks (Bilalb and Othmana, 2012). Moreover, the flexible NED language offered by OMNet++ allows full network topology customization even when the simulation is running, which meets the needs of our study. In the simulations, we consider the different networks in Table 1. These networks satisfy our hypotheses and have been randomly generated to

better assess the ability of our update strategies to apply to any network that satisfies our hypotheses.

The simulations have been performed on a computer with the following configuration: Core i5 2.40 GHz CPU, 4.00 GB of RAM and 12 MB of cache.

The objective of our simulations is to compare QoS of the network in the absence of persistent link failures and in the presence of such failures to show the efficiency of our routing tables update strategy for the QoS improvement in the presence of failures. To this end, we performed simulations on high data rate networks (1.2 Mbits/sec to 512 Mbits/sec) of various scales and focused on packet routing delays and the data loss rate. We also compare our results to several existing studies. For both link and node failures, the data were obtained by performing several tests on the networks in the following order:

- In the absence of failures
- In the presence of a transient link failure
- In the scenario where the above link or node failure was deemed persistent

The analysis of packets' routing delays in various nodes and various network instability scenarios allowed us to appreciate the QoS variability. Consequently, in the case of a single persistent link failure, the average of packet routing delays in network 3 is presented in Fig. 11. We note that the average packet routing time in the presence of a persistent link failure is lower than those observed in the presence of a transient failure; this occurs

because some packets are forwarded through the critical nodes. In addition, there is a high gap between the TBR (Tree-Based Routing) approach and the packet routing delay in the network without a failure. This significant difference could be explained by the TBR approach relying on the data of neighbours to construct its updating information; since data of certain neighbours might not reach the destination node, the update data can be wrong. Our approach actually uses all the information from neighbouring nodes to build its update data, allowing optimal paths that reduce the packet transit time.

Figure 11 reveals the results for a large-scale network (60 nodes); however, this result is similar to those for other tested networks (network1 and network 2) that are small. Network 3 is interesting to us because its size is more compatible with the SDN scale.

Considering the node failure problem, Fig. 12 presents the packet transit delays in the network. We note that the previous observations made for a persistent single link failure also apply here. The packet transit delays with our routing tables update method are lower than those in the literature. Then, in case of a persistent node or multilink failure, our method improves the packet transit delays.

Table 1: Simulation networks

Network	Number of nodes	Number of links
Network1	10	18
Network2	20	31
Network3	60	90

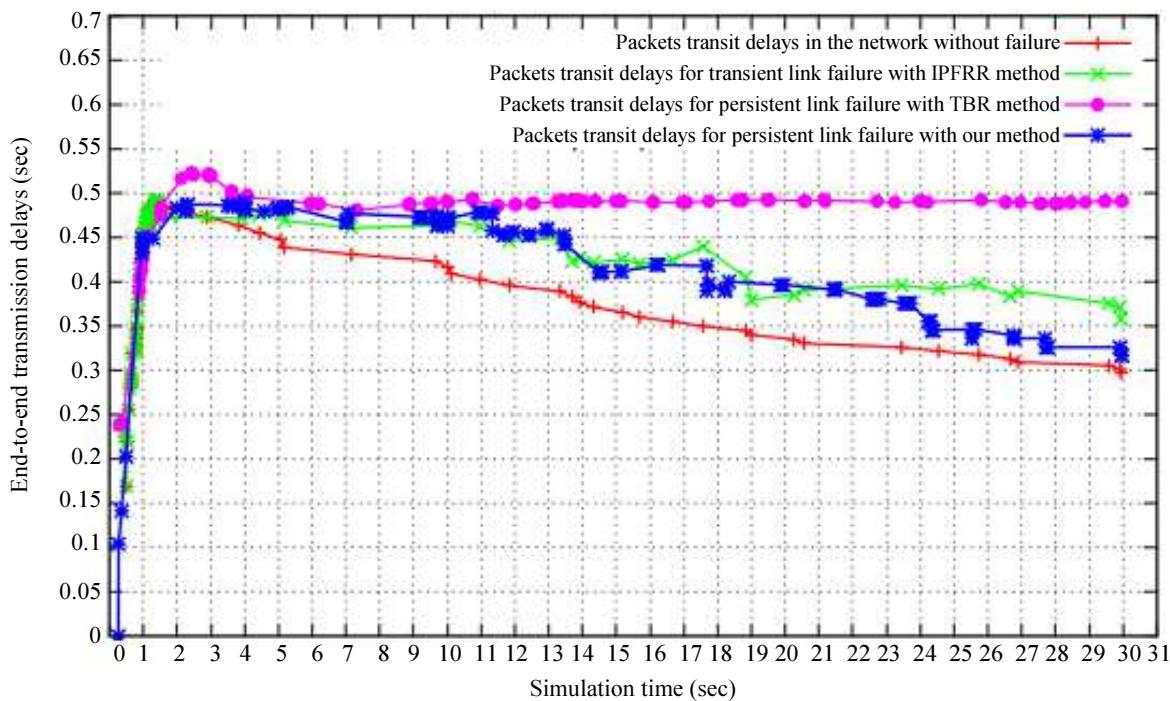


Fig. 11: Packets routing delays' variations for a persistent single link failure in network3

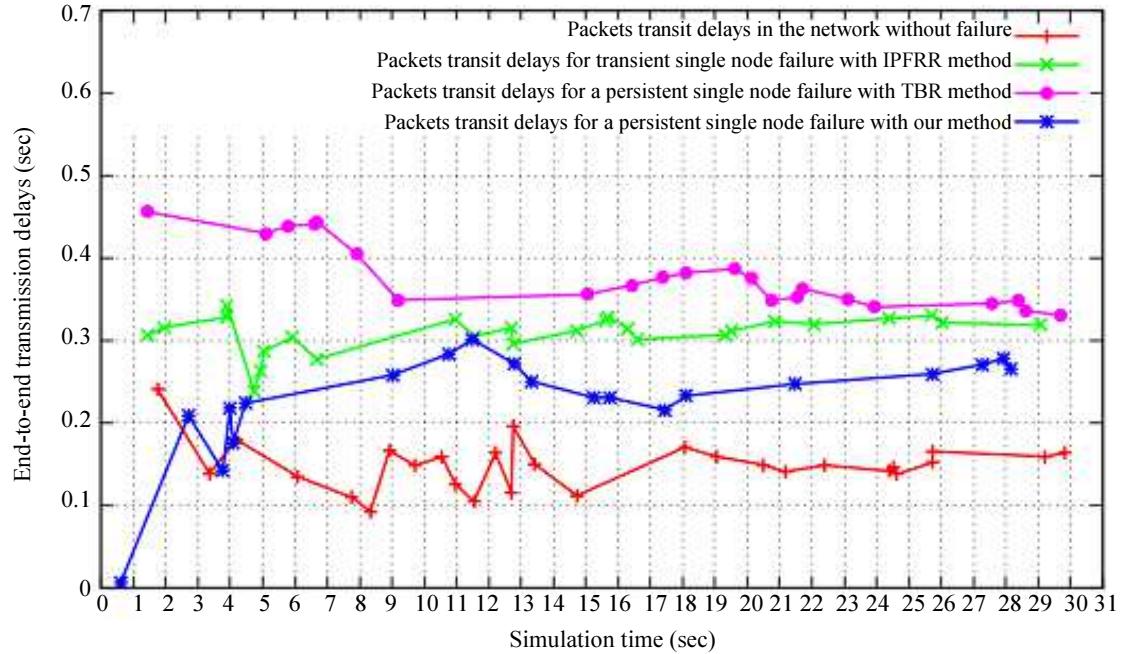


Fig. 12: Packets routing delays' variations for a persistent single node failure in network3

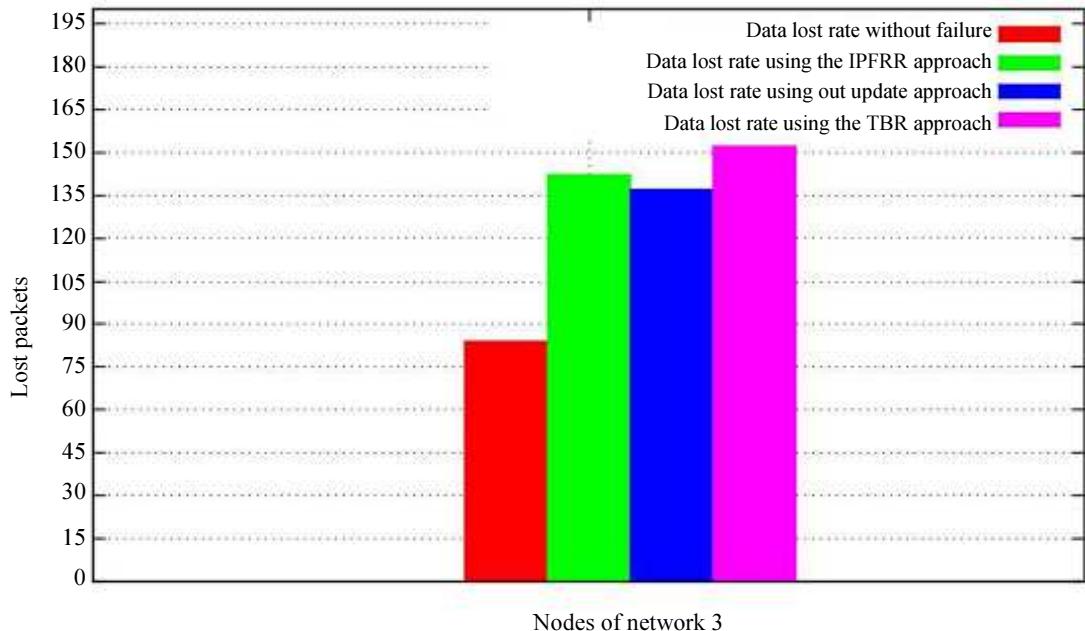


Fig. 13: Comparison of the data loss rate of our update strategy

Considering the data loss rate, we obtain the results displayed in Fig. 13 for a single link failure and Fig. 14 for a single node and multilink failures. Both figures show that a link or node failure highly increase the data loss rate in the network. This data loss rate is more important for the case of a persistent single node failure than the persistent single link failure. We note a strong increase by approximately 60% of this data loss rate

(Fig. 14) compared to those observed in the absence of failures. This phenomenon could be explained by the lack of resources needed to reroute the traffic. Nevertheless, our approach helps decrease this data loss rate to 24% (compared to the IPFRR method) and 43% (compared to the TBR method). This improvement means that our approach is better than TBR and IPFRR in terms of the data loss rate.

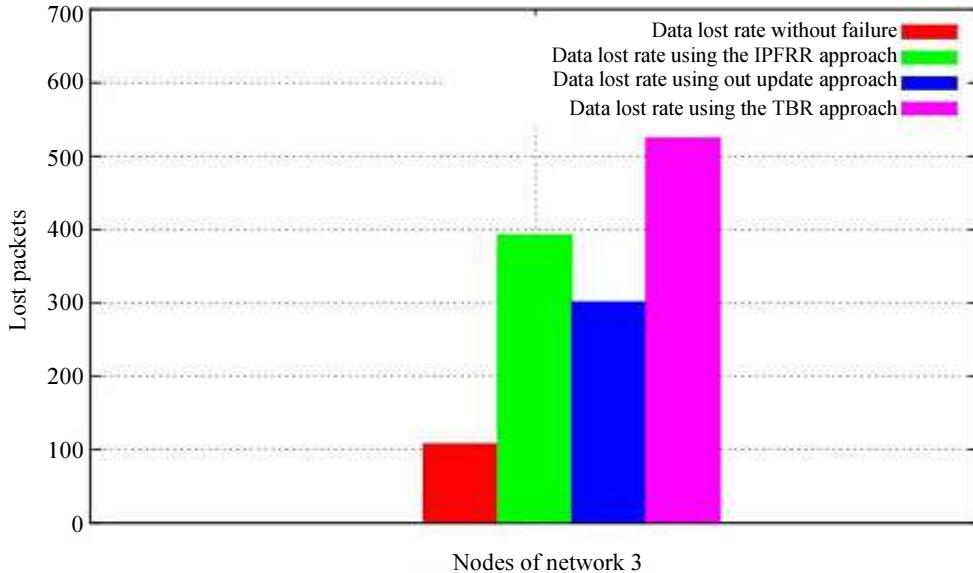


Fig. 14: Comparison of the data loss rate of our update strategy to those of Pham (2014) and TBR for network3 in case of node failure

Conclusion

In this paper, our aim was to propose a routing and rerouting tables' update mechanism to overcome nodes' reconfiguration challenges in case of persistent single link, multilink and node failures in a virtual network. Hence, we proposed an update strategy that efficiently identified the nodes to be updated and defined an update scheme for those nodes. This scheme, originally built on a vector of lists of triplets inspired by OSPF's LSU packet structure has been adapted to treating the cases of persistent multilink and node failures. The simulations we performed show that our routing tables update method helps reduce packet routing delays and the data loss rate, which are two important metrics in computer networks. Consequently, this study provides solutions to the routing tables' update challenge in the SDN architectures, to offer a good QoS at all times to the endusers and avoid losing customers to infrastructure providers (InP). In addition, the ideas presented in this paper and related to the management of multiple virtual planes in case of a node failure can help the InP better allocate resources for network recovery.

To improve this work, further research could specifically address the flow congestion problem in the network when there are numerous active users' requests. Even though our update approach is effective, update messages can be stopped by certain network congestion cases. In these cases, our method will not be very useful.

Acknowledgement

We thank the anonymous reviewers whose valuable comments and suggestions have significantly improved the presentation and the readability of this work.

Author's Contributions

Yannick Florian Yankam: The author contributed to the review the various published articles in the field, contributed to the hypothesis the date analysis, writing of the manuscript and final approval.

Jean Frédéric Myoupo: The author designed the research proposal, organized the study, designed of the proposed work plan, contributed to the hypothesis, writing of the final manuscript and final approval.

Vianney Kengne Tchendji: The author contributed as the research guide, technical corrections, reviewing it critically and final approval.

Ethics

This work is original and not published elsewhere. The authors confirm that they have read and approved the manuscript and there is no conflict of interest. Also, the authors confirm that there are no ethical issues involved.

References

- Azodolmolky, S., P. Wieder and R. Yahyapour, 2013. SDN-based cloud computing networking. Proceedings of the 15th International Conference on Transparent Optical Networks, Jun. 23-27, IEEE Xplore Press, Cartagena, Spain, pp: 2181-2206. DOI: 10.1109/icton.2013.6602678
- Bilalb, S.M. and M. Othmana, 2012. A performance comparison of open source network simulators for wireless networks. Proceedings of the IEEE International Conference on Control System, Computing and Engineering, Nov. 23-25, IEEE Xplore Press, Penang, Malaysia, pp: 34-38. DOI: 10.1109/iccsce.2012.6487111

- Doumith, E., 2007. Agrégation et routage de trafic dans les réseaux WDM multicouches. Ph.D Thesis, École Nationale Supérieure des Télécommunications (Télécom ParisTech), France.
- Farhady, H., H.Y. Lee and A. Nakao, 2015. Software defined networking: A survey. *Comput. Netw.*, 81: 79-95. DOI: 10.1016/j.comnet.2015.02.014
- Hedrick, C.L., 1988. Routing information protocol. RFC Editor.
- Hu, F., Q. Hao and K. Bao, 2014. A survey on software-defined network and OpenFlow: From concept to implementation. *IEEE Commun. Surveys Tutorials*, 16: 2181-2206. DOI: 10.1109/comst.2014.2326417
- Lim, A.O., X. Wang, Y. Kado and B. Zhang, 2008. A hybrid centralized routing protocol for 802.11s WMNs. *Mobile Netw. Applic.*, 13:117-131. DOI: 10.1007/s11036-0080038-4
- Mosharaf Kabir Chowdhury, N.M. and R. Boutaba, 2010. A survey of network virtualization. *Comput. Netw.*, 54: 862-876. DOI: 10.1016/j.comnet.2009.10.017
- Moy, J.T., 1998. OSPF: Anatomy of an Internet Routing Protocol. 1st Edn., Addison Wesley Professional, ISBN-10: 0201634724, pp: 339.
- Muruganathan, S.D., D.C.F., Ma, R.I. Bhasin and A.O. Fapojuwo, 2005. A centralized energy-efficient routing protocol for wireless sensor networks. *IEEE Commun. Magazine*, 43: S8-S13. DOI: 10.1109/mcom.2005.1404592
- Myoupo, J.F., Y.F. Yankam and V.K. Tchendji, 2018. A centralized and conflict-free routing table update method through triplets' lists vector in SDN architectures. Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, Oct. 8-12, IEEE Xplore Press, Guangzhou, China, pp: 1509-1515. DOI: 10.1109/SmartWorld.2018.00261
- Nashid, S., R. Ahmed, A. Khan, S.R. Chowdhury, R. Boutaba and J. Mitra, 2016. ReNoVatE: Recovery from node failure in virtual network embedding. Proceedings of the 12th International Conference on Network and Service Management, IEEE Xplore Press, Montreal, QC, Canada, pp: 19-27. DOI: 10.1109/cnsm.2016.7818396
- Papan, T., P. Segec, M. Moravcik, J. Hrabovsky and L. Mikus *et al.*, 2017. Existing mechanisms of IP fast reroute. Proceedings of the 15th International Conference on Emerging eLearning Technologies and Applications, Oct. 26-27, IEEE Xplore Press, Stary Smokovec, Slovakia, pp: 1-7. DOI: 10.1109/iceta.2017.8102516.
- Pham, T.S., 2014. Autonomous management of quality of service in virtual networks. Ph.D Thesis, University of Technology of Compiègne (UTC), France.
- Perkins, C., E. Belding-Royer and S. Das, 2003. Ad hoc on demand distance vector (aodv) routing. RFC Editor.
- Rothenberg, C.E., M.R. Nascimento, M.R. Salvador, C.N. Araujo Corrêa and S. Cunha de Lucena *et al.*, 2012. Revisiting routing control platforms with the eyes and muscles of software-defined networking. Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks, Aug. 13-13, ACM Press, Helsinki, Finland, pp: 13-18. DOI: 10.1145/2342441.2342445

On the Dynamic Replacement of Virtual Service Resources for Mobile Users in Virtual Networks

Jean Frédéric Myoupo^{1*}, Yannick Florian Yankam², Vianney Kengne Tchendji²

¹ Computer Science Lab-MIS, University of Picardie Jules Verne, 33 rue St. Leu 80039 Amiens, France.

² Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon.

* Corresponding author. Tel.: +33 322825915; email: jean-frederic.myoupo@u-picardie.fr

Manuscript submitted November 10, 2019; accepted January 8, 2020.

doi: 10.17706/jcp.15.1.10-21

Abstract: In virtual networks, network traffic is managed through the resources of a substrate network. When users move from one access point to another, the traffic that is generated increases in variation, and the virtual service resources within each virtual network must be maintained at an acceptable quality of service (QoS). However, the resources that are offered by this virtual service resource may become insufficient to manage this traffic, harming the QoS. In this paper, we propose a virtual service resource replacement method that consists of migrating a service from one virtual node to another, offering sufficient resources to provide a good QoS when the traffic changes. Our method improves those of the literature by dealing with the cases of equal traffic weight at the node level and the service migration strategy. Our approach reduces the service migration time and the number of virtual service resource replacements compared to those of the literature.

Key words: Virtual network, virtual service resource replacement, traffic change, mobile user, service migration.

1. Introduction

Virtual networks are networks that are built on the resources of a physical network and are used to provide services. These virtual networks have special properties such as architectural flexibility and isolation ([1], [2]) that they provide to the users, and various services that are hosted in servers are called virtual service resources. A virtual service resource in a virtual machine supports the incoming and outgoing traffic that is generated by mobile users relative to the requested service. Consequently, the quality of service (QoS) depends on the amount of traffic involved.

However, the traffic that is associated with a given service is not constant; rather, it increases and decreases depending on the number of users and the network structure mutation. Sometimes, the resources of the virtual server become insufficient to handle these changes wisely, harming the QoS. Therefore, there is a need for policies that allow the virtual network to continuously ensure the QoS to end users and deal with these traffic fluctuations and topology changes. These topology mutations come from the addition and removal of virtual machines in the network. Several possible solutions can be explored to deal with such a problem:

- Increase the resources of the virtual service resource so that it can support the amount of generated traffic. This solution is related to the well-known resource allocation and Virtual Network Embedding (VNE) problems in the virtual networks. These problems have been widely studied in the literature [3]-

[5]. The common disadvantage of the methods proposed in these works is that they progressively reduce the physical resources that are available for the benefit of virtual networks;

- move the virtual service from a virtual service resource to another one with more available resources: this solution has been very little explored so far [6]. This approach does not require additional resources from the substrate network. Our contribution in this paper is based on this approach.

With regards to the virtual service resource replacement, the most important challenges to be overcome in this scenario are the following:

- Find the new node that will receive the virtual service resource;
- migrate the service to the new node;
- restructure the virtual network after a topology change. When the network topology change is from a tree to a graph structure, a tree topology needs to be rebuilt if we want to use method [6] as the virtual service resource replacement approach;
- design a multiple virtual services resources migration approach in a VN.

1.1. Related Works

Revisiting the dynamic replacement of virtual service resources in the literature, we found that this approach has been explored very little to the best of our knowledge.

In the field of virtual networks with a tree topology, [7] proposes a dynamic replacement of virtual network resources to satisfy the quality of service of the provided service when the traffic changing comes from a new node that is added in the network. The idea is to migrate the virtual service resource successively by one hop from the overloaded node to another one that has enough resources, until the new destination node is reached. At each migration to a node, it is determined whether the quality of service is satisfied before proceeding to the next node. The new node is the node at one hop of the new added node. Fig. 1 shows this replacement method. When the new node n_k (Fig. 1.a) is added in the topology, the virtual service resource that is initially located in node n_2 is replaced in node n_6 at one hop of node n_k through the shortest path (Fig. 1.b). The main drawback of this method is that it takes a lot of time to replace a virtual network service, which harms the QoS.

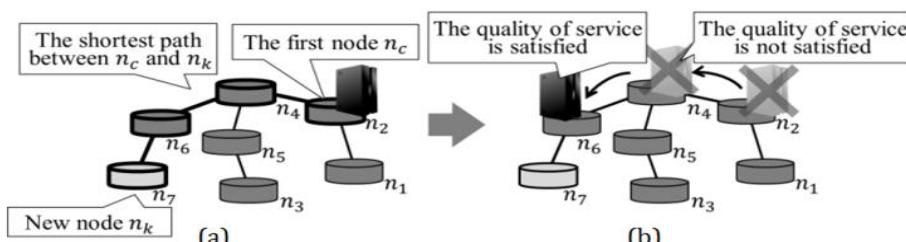


Fig. 1. A virtual service resource replacement method by moving the virtual network service by one hop.

To overcome the drawback of [7], [6] proposes a dynamic virtual service resource replacement method that does not check if the quality of service is satisfied at each node between n_c and n_k . This results in a reduced virtual service resource replacement time. The method proposed in [6] deals with three key challenges:

- a virtual service resource replacement from one virtual node to another;
- a tree topology redesign when the network topology changes after a node is added into or removed from a virtual network with tree topology;
- the replacement of 2^n virtual service resources.

However, the solutions proposed in [6] have the following limitations:

- no replacement solution when all of the leaf nodes of the tree topology have the same traffic weight;
- too much virtual service replacement depending on traffic variations;
- the service is not replaced by the initial node when traffic decreases or returns to normal. The replacement had been done to face a traffic change in the network. The virtual service should be replaced by its initial virtual node to satisfy the QoS when the network traffic returns to normal;
- the migration strategy (pre-copy, post-copy, hybridization) of the virtual service from one node to another is not studied or discussed. The service migration time is significant and greatly influences the QoS ([8], [9]) of the virtual networks when we are dealing with virtual network resource replacement.

1.2. Our Contribution

The objective of this paper is to improve the QoS of virtual networks when the traffic changes by providing solutions to the limits cited above concerning the approach proposed by Horiuchi and Tachibana [6]. Our contribution is summarized in four points:

- a selection strategy of a QoS-satisfying node when all of the leaf nodes in the tree topology have the same traffic weight. This selection is based on the amount of traffic of the parent node. The selected leaf node is one whose parent node has the largest traffic weight;
- avoid automatic virtual resource replacement at the first sign of QoS dissatisfaction. A bad QoS state can be temporary. In this case, a virtual service replacement is not necessarily required. Therefore, we define a QoS dissatisfaction delay as one that is beyond the virtual service resource replacement that is performed;
- automatic replacement of the virtual service to its initial host node after solving the problem that is caused by traffic variation. This allows a better use of the virtual network resources;
- a virtual service migration strategy from the source node to the destination node. We propose a hybrid migration approach because it has the advantage of minimal downtime and data copying as well as the retention of the replaced service data consistency ([8], [10]).

The remainder of this paper is organized as follows: In Section 2, we describe in detail the approach of [6] with its drawbacks. Section 3 presents our contribution in more detail. The simulation results and discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2. Presentation of Former Work

In this section, we describe the Horiuchi and Tachibana approach [6] with its drawbacks.

2.1. The Virtual Service Resource Replacement Approach of Horiuchi and Tachibana and its Drawbacks

Consider a virtual network $G = (N, L)$ as a tree, where $|N|$ denotes the number of nodes and $|L|$ the number of links. Each node n_i has a quantity of resources m_i , and each link l_{ij} between the nodes n_i and n_j has a traffic weight m_{ij} . The variables γ_i and γ_{ij} are the amount of traffic passing through the node n_i and link l_{ij} , respectively. The idea is to select in the tree, the node of which allows the management of the greatest amount of traffic in the topology. The search algorithm starts by selecting the leaf node with the minimum amount of traffic; then, the weight of this traffic is added to that of its parent node in the tree. After that, this leaf node is logically removed from the research process. This process is repeated until the tree contains two nodes; in this case, the new node to select for the virtual service replacement is the one with the greatest traffic weight. An illustration of this selection approach is presented in Fig. 2. At the end of the search process for the case of Fig. 2, the remaining nodes are n_1 and n_2 . The node n_1 is selected as the new node for the virtual service since its amount of traffic is 40 compared to n_2 which has an amount of traffic of 35.

The first limit of this approach is that there is no replacement solution when all of the leaf nodes of the tree

have the same traffic weight. In this case, the algorithm fails from the beginning, and no result is produced at the end of the search process. Fig. 3 illustrates this limit. All of the leaf nodes F, E and C have a current traffic weight that is equal to 12. Which one should be selected? The approach of Horiuchi and Tachibana ([6]) does not answer this question.

The second limit is that there are too many virtual service replacements, depending on the traffic variations. Each time the QoS is not satisfied because the amount of traffic changes, a virtual service replacement is performed. These multiple replacements create instability in the offered network service as well as a load imbalance in the nodes of the virtual network.

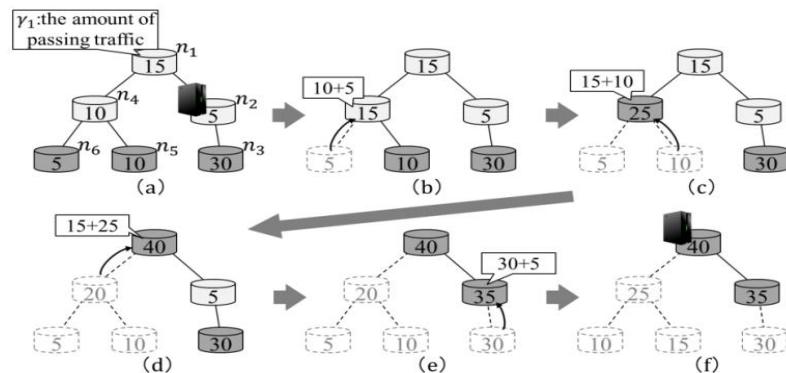


Fig. 2. The virtual service resource replacement method of Horiuchi and Tachibana.

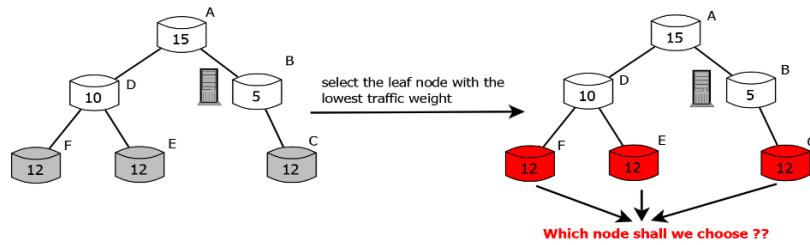


Fig. 3. Limit 1: The equal traffic weight challenge.

Third, the virtual service resource should be replaced with its original virtual node once the amount of traffic has returned to normal. The virtual service replacement in another node leads to the use of the resources of this node that are planned in principle for a specific service. The exploitation of these resources in an unexpected way can create some disturbances in the service proposed by the node; thus, it is necessary to temporarily use the resources to manage the traffic changing and then replace the service in its starting node. This would avoid overusing the resources of some nodes. This returning aspect of the virtual service to its original location after replacing the service is not considered by Horiuchi and Tachibana.

Finally, a service migration strategy is not discussed. A virtual service resource replacement involves the transfer of a data set from one virtual machine to another; these data are characteristic of the state of the migrated service so that it is restored more efficiently in the new physical or virtual node that will receive it. These characteristics are: the amount of volatile memory required, the non-volatile memory, the status of the virtual CPUs, the status of the connected devices and the status of the active connections [9]. It takes time for these elements that characterize the service to be routed from a virtual node to another, and this process requires the implementation of an adapted migration approach to maintain the consistency of the service provided to end users. Depending on the case, it can be considered as a migration of the service files or a migration of the operating system hosting the virtual service. In both cases, the migration time affects the service availability. Therefore, this migration time should be considered and minimized.

2.2. Dynamic Replacement of 2^n Virtual Service Resources of Horiuchi and Tachibana and its Drawbacks

The objective is to replace multiple virtual service resources in a virtual network. This is an extension of the previous approach that was proposed for one virtual service resource replacement.

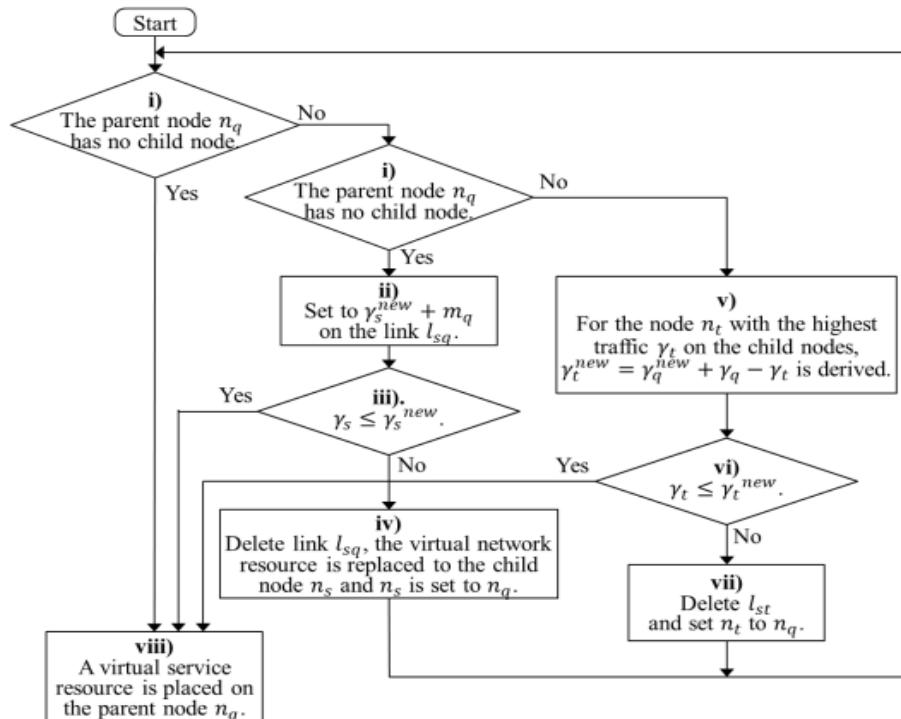


Fig. 4. Flowchart for 2^n virtual service resource placement.

A virtual network is logically divided into two virtual networks according to the traffic weight, using the algorithm of section 2.1; in each subdivided virtual network, a virtual service resource is placed using the flowchart in Fig. 4, where γ_i is the amount of traffic in node n_i and γ_i' is the traffic variation of node n_i . According to this diagram, the node that is chosen as the host of the virtual service resource is the one that has enough resources to support the amount of traffic that is generated by all its children nodes. The virtual service resource in each subdivided virtual network offers the same virtual service as the original virtual network and is replaced according to the method described in Section 2.1.

For example, considering Fig. 5, we get a network with the two extreme nodes n_4 and n_6 by applying the algorithm that was described in Section 2.1. These two nodes are the root nodes of the sub-trees graph A and graph B, which represent segments of the original virtual network.

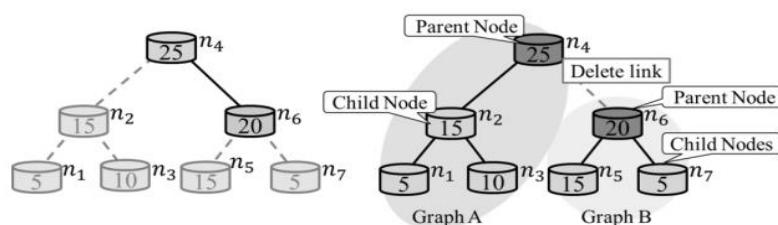


Fig. 5. Segmentation of the virtual network into two network segments.

The subdivision into sub virtual networks helps to reduce the traffic load of the original virtual node that hosted the amount of traffic for the proposed service.

The first drawback of this extended replacement approach is that there is no replacement solution for a number of virtual service resources $R \neq 2^n$. The dynamic replacement approach of [6] uses the subdivision of the original virtual network into 2^n segments according to the amount of traffic to perform a dynamic replacement of 2^n virtual service resources. This assumes that any traffic weight can be satisfied through at most 2^n virtual service resources, but this is not always verified. Thus, for any traffic weight requiring subdivision into an odd number of sub virtual networks where $R \neq 1$, Horiuchi and Tachibana [6] does not provide a solution.

Second, the heterogeneity of services is not considered in the 2^n virtual service resource replacement approach of [6]. User-generated traffic can involve several different services that are distributed across one or more virtual networks. In this case, replacing a virtual service resource can lead to the replacement of another dependent one, based on the type of traffic that is generated. A user can be simultaneously connected on a Video on Demand (VoD) service and a Voice over IP (VoIP) service. In [6], the authors consider a single virtual network service and not the dependency between multiple services.

3. Improvements of the Horiuchi and Tachibana Approach

In this section, we provide some solutions to the drawbacks of the virtual resource replacement approach in [6]. These solutions address the problems of equal traffic weight in the leaf nodes of the tree topology, automatic replacement and the establishment of a data migration mechanism of the virtual service.

3.1. Our Solutions to the Equal Traffic Weight Challenge in the Leaf Nodes

To address this problem with the tree topology, we can proceed as follows: select the leaf node whose parent node has the largest traffic weight. From here, two scenarios can arise:

If the direct parent nodes have equal traffic weights, we go back in the tree until we find a level where the traffic weights of the nodes are different. Once this level is reached, one of the leaf nodes of the node with the largest traffic weight is selected; then, we add the traffic weight of this leaf node to that of its parent node. We remove the previously selected leaf node from the research process and repeat this process until there are only two nodes in the tree. Finally, the node that is selected as new location for the virtual service resource is the one with the highest traffic weight.

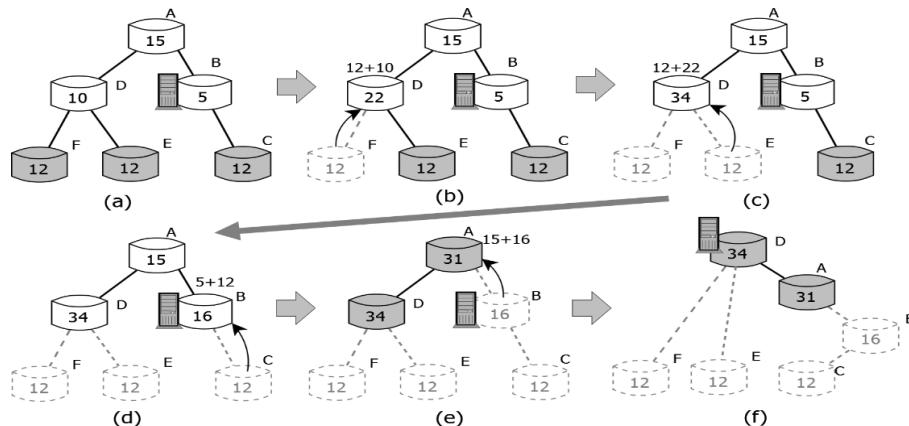


Fig. 6. First case for the problem of equal traffic weight in the leaf nodes with parents of different traffic weight.

If we have a tree with a root node followed directly by the leaf nodes (with equal traffic weights), we randomly select any leaf node (see Fig. 7). Generally, if the tree topology height is $h=1$, then any leaf node can be selected.

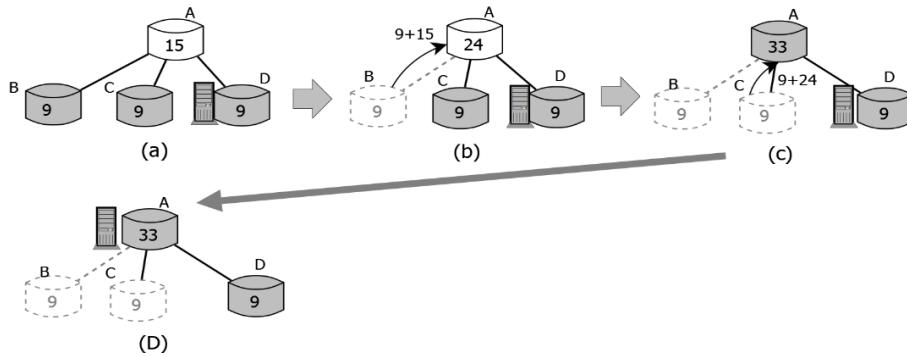


Fig. 7. Second case for the problem of equal traffic weight in the leaf nodes with tree topology height $h = 1$.

3.2. A Solution to Reduce the Number of Replacements

In [6], when the quality of service is not satisfied because of traffic change, a virtual service resource replacement is immediately performed; this replacement, which aims to improve the quality of service, may deteriorate it further because of the time that is needed to effectively replace a virtual service resource and the service disruptions in the network [11]. If multiple replacements are done simultaneously, the QoS will be greatly affected. Thus, a virtual service resource replacement approach is necessary, but only in the case of emergency. We define a delay beyond the replacement process that is effectively initiated. The case of the virtual service replacement emergency could be defined through some criteria that we mention here:

- *The type of service offered:* Depending on the services that are offered by the virtual networks, their resources and flow management policies requirements, we must consider their nature and the priorities between these services [4]. Thus, when the amount of traffic changes, the immediate reaction time is relative to the nature of the service that is involved. In addition, certain services such as VOIP and VOD are less tolerant to latency times than others; consequently, these services require a reaction time that is relatively fast in situations of failure.
- *The quantity of available resources in the considered virtual environment:* It can be possible that when we want to replace a virtual service resource, we do not find a node that is able to control the amount of traffic involved; therefore, it is necessary to wait before performing a replacement in this situation.

The reaction time that is set up to address this improvement path would avoid unnecessary replacements due to temporary traffic overweight. This reaction time must be a network setting that is defined by the administrator based on the type of service.

3.3. The Data Migration Mechanism

There are several mechanisms for migrating virtual entities from one node to another [12]:

- *Cold migration (or stop-and-copy)* [11]: In this approach, the service is first stopped, and then its data are copied to the new node; once the copy is complete in the destination node, the service is started in that node. The main advantage of this method is that it ensures the faultless migration of the server memory. In addition, the state of the server memory is not changed on the source host. On the other hand, the downtime and start times on the destination node are longer.
- *Live migration* [11], [13]: There are three approaches of live migration: pre-copy, post-copy and hybrid post-copy.

The pre-copy approach (see Fig. 8) mainly consists of transmitting all of the virtual service resource memory pages to the destination host while the service is running. After a given copy level, the service is stopped on the source host and the rest of the modified memory pages are copied to the destination host. The major problem occurs when the maximum interruption time is too low for sending the last modified

pages to the destination node [9]. In this case the migration time can be very high.

In contrast to the pre-copy, the post-copy (see Fig. 9) starts with the immediate shutdown of the virtual machine on the source host. Then, the subsidiary data are transferred before activating the service on the destination host. The major disadvantage compared to the pre-copy is based on the robustness of this approach. Indeed, since the VM is directly awakened on the destination host in an inconsistent state, the failure of one of the nodes, source or destination during migration causes the inevitable loss of the integrity of the VM memory state.

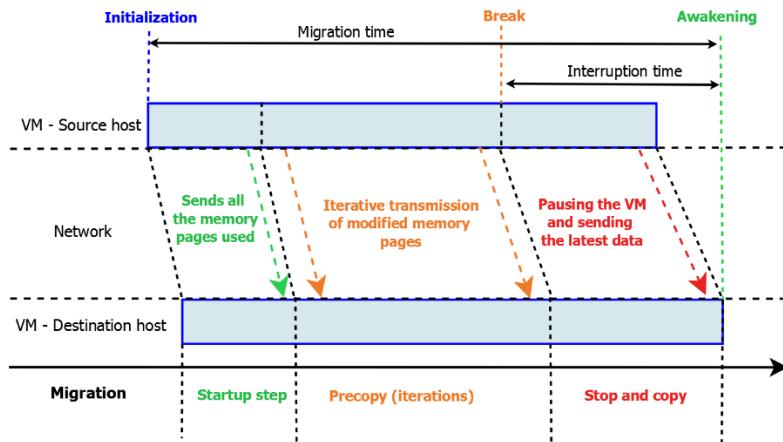


Fig. 8. Pre-copy algorithm.

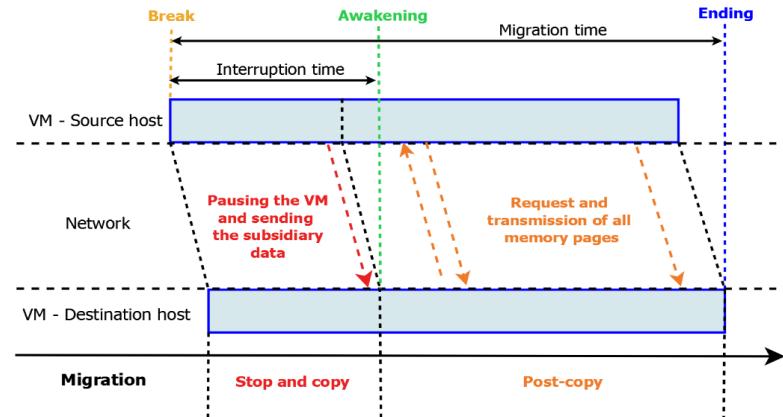


Fig. 9. Post-copy algorithm.

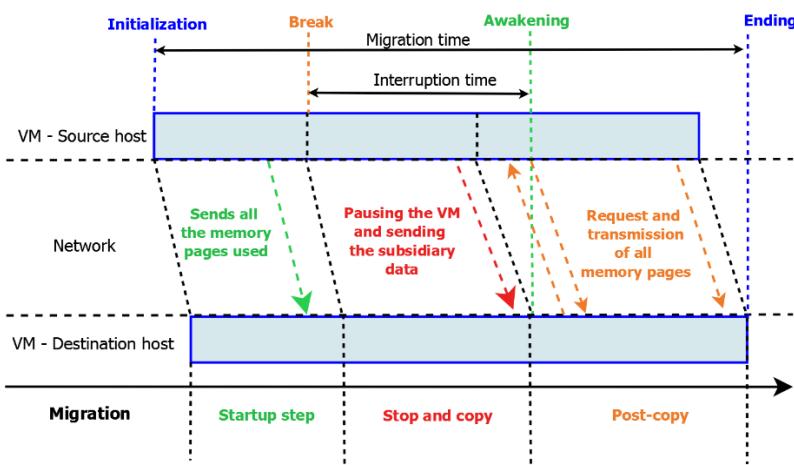


Fig. 10. Hybrid post-copy algorithm.

Hybrid post-copy [10] has been proposed to reduce post-copy performance issues. Fig. 10 shows the hybrid post-copy algorithm. Without interrupting the outstanding service, the progressively modified memory pages are transferred from the source node to the destination (pre-copy). The copy of the modified memory pages ends when a critical point is reached; therefore, the service is suspended on the source machine and its state is restored on the destination machine with consistent data (post-copy). We propose this hybrid migration approach, which substantially reduces both the downtime and the total migration time.

4. Simulations and Discussions

The objective of the simulations that follow is to evaluate the performances of our replacement approach, which is an improvement over the Horiuchi and Tachibana one. We compare our approach with the Horiuchi and Tachibana version. The performances are evaluated through the impact of the number of replacements on the overall QoS as well as the migration time of the services between a source node and a destination node. In this section, we present these performance results from the OMNET ++ simulator. The simulations were conducted on several network samples: Network1 is a small network (20 nodes and 31 links), and Network2 is a large network (60 nodes and 90 links). The aim of the tests on the different sized networks is to evaluate the effectiveness of both approaches according to the scalability of the network. The structure and resource weights of each network were randomly generated to show the adaptation of the tree topology construction strategy to any type of traffic weight within the nodes.

4.1. The Impact of the Number of Virtual Service Resource Replacements on the QoS

In this section, we present the influence of the number of replacements on the overall QoS in a network. Indeed, we have proposed in the improved version of the Horiuchi and Tachibana [6] approach that the number of replacements can be reduced by using a replacement starting delay k . For each network, we performed some tests for different replacement initialization delays: $k = 2$ sec, $k = 6$ sec and $k = 10$ sec. Fig. 11 and Fig. 12 illustrate the results of this study for networks of 20 and 60 nodes. In a small network (see Fig. 11), we found that the replacement rate is relatively low compared to that of large networks (see Fig. 12). This observation is reflected in many similarities in the number of replacements that are made after each variation of the replacement initialization delay k . This can be explained by the fact that a large number of nodes also induces a significant amount of traffic [6]. In Network2, there is almost no such similarity. This means that, as the number of nodes is high, many replacements are possible, and among them, many can be avoided. In all cases, these simulations show that our replacement approach, which imposes a replacement initialization delay, is better than that of Horiuchi and Tachibana, which performs many unnecessary replacements and has high amount of downtime compared to our approach.

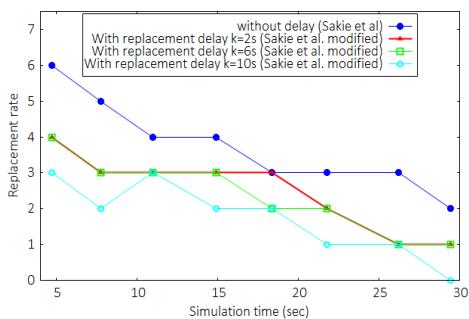


Fig. 11. Replacement rate for a 20-node network.

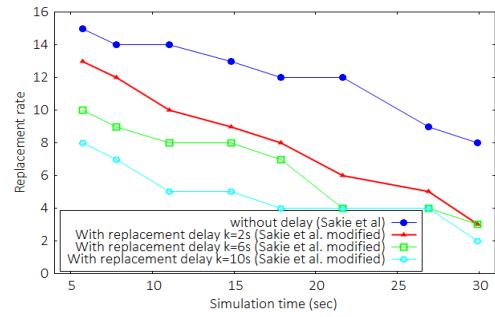


Fig. 12. Replacement rate for a 60-node network.

Concerning the reduction in the amount of traffic, Fig. 13 and Fig. 14 give a comparison of the traffic variation management method of Horiuchi and Tachibana with its modified version (Horiuchi and Tachibana

modified) that we proposed for Network1 (see Fig. 13) and Network2 (see Fig. 14). Both methods have almost the same traffic reduction rate, whether in a small or large network. Nevertheless, there are some differences, like the fact that the Horiuchi and Tachibana approach reduces the amount of traffic more than our approach. This difference is explained by the fact that, because of the replacement initialization delay that we impose, some replacements are not performed. In this case, the traffic weight is not greatly affected

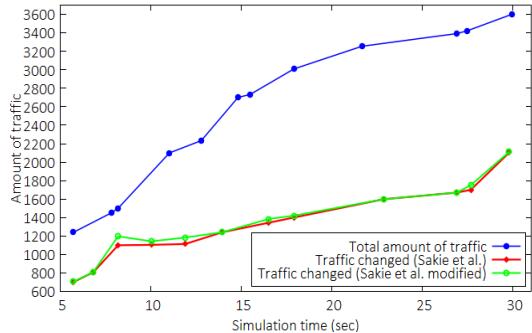


Fig. 13. Total amount of traffic variation for Network1.

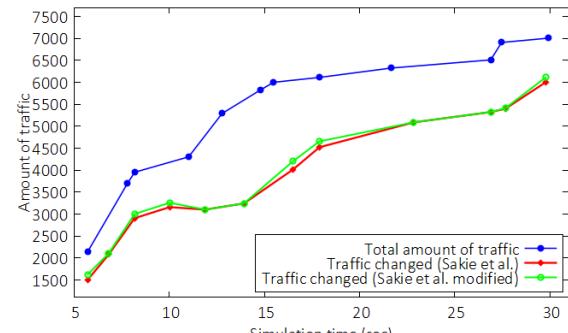


Fig. 14. Total amount of traffic variation for Network2.

We can also see that the traffic reduction rate is not very high in a large network compared to that of small networks. This can be explained by the difficulty to control all of the network traffic when the number of nodes and network users is important.

4.2. The Impact of the Number of Virtual Service Resource Replacements on the QoS

To measure the impact of migration time on QoS, we observed during the simulations the migration rate and the average time of migrations over a simulation time interval of 0 to 30 seconds. We compare our approach, which integrates migration, with that of Horiuchi and Tachibana, which does not consider it. Fig. 15 and Fig. 16 show that the average migration time of Horiuchi and Tachibana is lower than ours. This is explained by the fact that for each migration, the method of Horiuchi and Tachibana transfers all of the data on time, without a stop-and-copy step, which reduces the total migration time. Nevertheless, as we have shown previously, this method is not very realistic. This observation is the same in large networks (see Fig. 16).

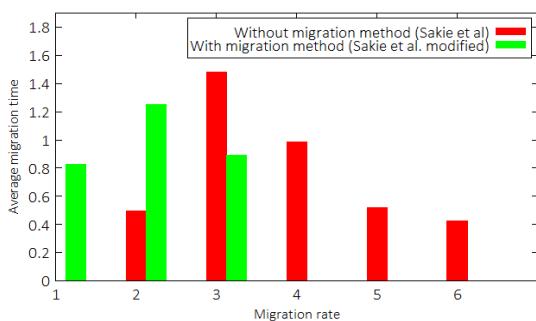


Fig. 15. Total amount of traffic variation for Network 1.

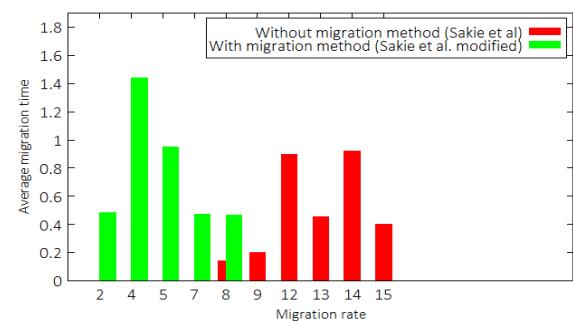


Fig. 16. Total amount of traffic variation for Network 2.

Based on the preceding results, we can conclude that the virtual resource replacement approach that we have proposed certainly has many points of similarity with the Horiuchi and Tachibana one, but it greatly improves upon it by integrating elements to better appreciate the replacement effects on QoS in a real virtual

network.

5. Conclusion

Our aim in this paper was to improve the QoS of virtual networks against traffic changes due to user mobility. We proposed a dynamic virtual resource replacement approach that helps to reduce the amount of traffic load in the nodes. This proposed approach is an improvement of the Horiuchi and Tachibana approach. It has the advantage of avoiding replacements due to temporary traffic changes that do not disturb the QoS for a long time. In addition, our approach preserves the consistent state of the replaced virtual service resource when it is restored to the destination host, using a data migration technique that we proposed. The numerical results of our simulations helped to better demonstrate the impact of these advantages on the QoS in the virtual network. Therefore, it appears that our replacement approach is better than that of Horiuchi and Tachibana and is closer to reality. However, the virtual service resource replacement idea could be just as useful in other research fields with the same QoS issues. For example, in the IoT (Internet of Things), equipment resources (storage space, processing power, energy) are generally limited. Consequently, there is a permanent need for resources from the equipment in this field to ensure QoS. Then, it would be interesting for the future work to deal with this dynamic virtual service resource replacement problem in the field of the IoT.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

JFM suggested this work and gave the main idea. YFY carried out analysis and experimentations. VKT and YFY wrote the first draft of the paper. JFM revised the first draft to get the final submitted manuscript. In addition, all authors read and approved the final version of the work.

References

- [1] Fernandes, N. C., Moreira, M. D., Moraes, I. M., Ferraz, L. H. G., Couto, R. S., Carvalho, H. E., et al. (2011). Virtual networks: Isolation, performance, and trends. *Annales des télécommunications*, 66(5-6), 339-355.
- [2] Chowdhury, N. M. K., & Boutaba, R. (2009). Network virtualization: state of the art and research challenges. *IEEE Communications magazine*, 47(7), 20-26.
- [3] Pham, T. S., Lattmann, J., Lutton, J. L., Valeyre, L., Carlier, J., & Nace, D. (2012). A restoration scheme for virtual networks using switches. *2012 IV International Congress on Ultra Modern Telecommunications and Control System* (pp. 800-805). IEEE.
- [4] Seddiki, M. S. (2015). *Allocation Dynamique des Ressources et Gestion de la Qualité de Service Dans la Virtualisation des Réseaux*. Doctoral dissertation, Université de Lorraine.
- [5] Shahriar, N., Ahmed, R., Chowdhury, S. R., Khan, A., Boutaba, R., & Mitra, J. (2017). Generalized recovery from node failure in virtual network embedding. *IEEE Transactions on Network and Service Management*, 14(2), 261-274.
- [6] Horiuchi, S., & Tachibana, T. (2018). Dynamic Replacement of virtual service resources based on tree topology for mobile users in virtual networks. *Journal of Computers*, 13(12), 1335-1349.
- [7] Koyanagi, Y., & Tachibana, T. (2014). *Dynamic Resource Allocation Based on Amount of Traffic in Virtual Networks* (Report Vol. 113, No. 473). Miyazaki, Japan: IEICE Technical Report.
- [8] Liu, J., Li, Y., & Jin, D. (2015). SDN-based live VM migration across datacenters. *ACM SIGCOMM Computer Communication Review*, 44(4), 583-584.
- [9] Kherbache, V. (2016). *Ordonnancement des Migrations à chaud de Machines Virtuelles*. Doctoral

dissertation, Université Côte d'Azur.

- [10] Luo, Y., Zhang, B., Wang, X., Wang, Z., Sun, Y., & Chen, H. (2008). Live and incremental whole-system migration of virtual machines using block-bitmap. *Proceedings of the 2008 IEEE International Conference on Cluster Computing* (pp. 99-106). IEEE.
- [11] Keshavamurthy, U., & Guruprasad, H. S. (2015). VM migration: A survey. *Global Journal of Engineering Science and Researches*, 2, 219-224.
- [12] Venkatesha, S., Sadhu, S., & Kintali, S. (2009). Survey of virtual machine migration techniques. *Memory*.
- [13] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., & Warfield, A. (2005). Live migration of virtual machines. *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2* (pp. 273-286). USENIX Association.

Copyright © 2020 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited ([CC BY 4.0](#)).



Jean Frédéric Myoupo is a professor of computer science in the University of Picardie-Jules Verne (UPJV), Amiens, France, where he leads the parallel and mobile computing research group in the Computer Science Lab-MIS. Professor Myoupo is the former dean of the Faculty of Mathematics and Computer Science of UPJV from 1999 to 2002. He received his Ph.D in applied mathematics from the Paul Sabatier University of Toulouse, France, in 1983 and his habilitation in computer science from the University of Paris 11, Orsay, France, in 1994. He held many positions in mathematics and computer science departments such as lecturer or associate professor in different universities, such as the University of Sherbrooke, Québec, Canada, the University of Yaounde, Cameroon, the University of Paris 11, Orsay, France and the University of Rouen, France. Professor Myoupo has served as member of program committee of international conferences, and he has been associate editor or member of editorial board of many international journals in computer science. His current research interests include parallel algorithms and architectures, and mobile computing and network management.



Yannick Florian Yankam is a Ph.D student at the University of Dschang, Dschang, Cameroon. He received his master degree in networks and distributed services from the Faculty of Science, University of Dschang, in 2016. His current research interests include computer networks, network virtualization, quality of service in virtual networks and cloud computing.



Vianney Kengne Tchendji is a senior lecturer of computer science at the University of Dschang, Dschang, Cameroon. He received his Ph.D in computer science from the University of Picardie-Jules Verne, Amiens, France, in June 2014. His current research interests include network virtualization, parallel algorithms and architectures, scheduling, wireless communication and ad hoc networking.

A Centralized and Conflict-free Routing Table Update Method Through Triplets' Lists Vector in SDN Architectures

Jean Frédéric Myoupo
Computer Lab-MIS
University of Picardie Jules Verne
Amiens, France
jean-frederic.myoupo@u-picardie.fr

Yannick Florian Yankam , Vianney Kengne
Tchendji
dept. of Mathematics and Computer Science
University of Dschang
Dschang, Cameroon
yyankam@yahoo.fr, vianneykengne@yahoo.fr

Abstract—For a long time, the distributed management protocols have proven their efficiency in managing computers' networks; however, the recent success of centralized management architectures such as software-defined networking (SDN) to supervise large-scale network infrastructures such as the Cloud, is attracting increasing interest from major computer network companies. In this architecture type, the network switches and routers just simply forward packets by following the flow table rules set by a controller, which owns the control plane of the network. Thus, faced with a failure, for example, the concerned nodes follow the rules provided by the same controller. With the IPFRR (IP Fast ReRoute) routing strategy known for its fast rerouting capabilities, these rules are computed and set up in the devices in advance. However, when a link or node failure becomes persistent over time, the controller must recompute the routing paths and update the nodes in order to keep an acceptable quality of service (QoS). In this paper, we propose a conflict-free mechanism for the routing and rerouting tables' updates of the nodes by the controller without disrupting the current traffic. We describe an efficient strategy of choosing the nodes to update, and we define an update scheme for these nodes. We demonstrate through the simulation of our update's scheme on different networks that our strategy improves QoS by reducing packet routing delays and the data loss rate in case of a persistent link failure in the network.

Keywords-Centralized architecture; software-defined networking (SDN); network virtualization; QoS; network recovery.

I. INTRODUCTION

The adoption of new network paradigms, such as network virtualization, has made it possible to cope with new types of users' needs (on demand videos, VOIP, etc.) and find solutions to the limits encountered in the Internet [1]; it has also led to the implementation of a new network paradigm such as cloud computing to provide resources to the users' requests dynamically [1, 2]. However, given the stakes and the network infrastructure, which become increasingly large and complex to supervise, new network management strategies were considered: these are centralized architectures, such as software-defined networking (SDN) [3, 4]. The idea is to separate the control plane from the data plane of each equipment. The control plane [4] is centralized

within a main equipment called the controller, while the data plane [4] is kept on the network devices (e.g., switches and routers). SDN provides flexibility and programmability, which meet the cloud automation needs and make it easier to add new services in communication networks [4, 5]. In such a context, when a failure occurs in the network, the devices forward the packets according to the rules defined in their flow tables [6, 7]. These rules are computed and integrated inside the switches/routers in advance.

However, when a failure is determined to be persistent, these rules become obsolete and no longer allow for a good QoS in the network. A failure is determined as persistent if its duration is more than ten minutes [7]. Then, it is necessary to recalculate the routing paths and update the routing tables of the devices. The most important challenges to overcome by the controller in this case of failure are the following:

- to construct news routing and rerouting rules based on the older rules in order to handle the failure;
- to select the nodes to update;
- to choose the efficient update order that will be applied to the nodes. Shall all the involved nodes be updated simultaneously or sequentially?
- shall all the computed rules be sent at the same time? This challenge is about the packet weight and the maximum transmission unit (MTU) value of the networks' interfaces. In large-scale networks, the size of the updates' packet can increase easily with the nodes.

A. Related Works

Looking at the routing tables' update problem in the literature, to the best of our knowledge, many of the centralized [8, 9], distributed [10, 11] and even hybrid [6, 8] rerouting schemes provided in computers' networks simply define the rerouting paths calculation and do not state specially the routing tables update challenge. Maybe this is because few ideas about that have already been stated in basic routing protocols such as OSPF (Open Shortest Path First) [10], RIP (Routing Information Protocol) [11, 12], and EIGRP (Enhanced Interior Gateway Routing Protocol) [13]. Nevertheless, those distributed protocols are not individually

efficient in centralized architectures such as SDN with Openflow [14, 15].

For rerouting schemes based on a tree topology computation in wireless mesh networks (WMN) [8], a lot of messages are exchanged between the root node and its child nodes when a routing table update is needed in case of a link failure; moreover, each node that needs to update its routing table must send a Route Request (RREQ) message to the root before, and the root replies with an update message packet called Route Set (RSET). In a large-scale network, this could be very harmful.

In SDN centralized architecture, when a persistent link failure is determined, [7] suggests an update strategy based on an IPFRR (IP Fast ReRoute [16]) mechanism that avoids looping. In that work, the authors propose to update the nodes from the destination of a routing tree till to the node that detects the failure. But, this scheme has the following limitations:

- overloading of the rerouting path, which may affect the handling of other link failures and load balancing in the network;
- it involves nodes for which an update is not necessarily needed. This increases the conflict risks to manage in the rerouting paths;
- it only states on the order that the routing tables will be updated from one node to another one but does not state how the topology recomputation will be done and how the update's information will be sent by the controller.

B. Our Contribution

The main motivation of this paper is to improve the computer networks QoS by providing some solutions to the drawbacks cited above concerning the works [7, 8]. So, our contribution in this paper solves the limits cited above by proposing a mechanism for update the routing and rerouting tables of the nodes from a controller. This mechanism is related to link failures, and we describe it through three points:

- the determination of the nodes to involve in the update process: we show that by updating some particular nodes, we can obtain better packet routing delays in the case of a persistent link failure compared to the transient failure;
- the construction of an update packet structure: we present a packet structure in the form of a vector of lists, inspired by the LSU (Link State Update) and LSA (Link State Advertisement) packets' structure of the Open Shortest Path First (OSPF) protocol [10]. The structure of the updates' packet must be thought carefully in order to provide a lighter updates' packet structure, which can help to forward easily the updated routing information over the network;

- the definition of a nodes' update scheme: We opt for both a simultaneous and sequential update of the involved nodes. Simultaneous update refers to particular nodes called critical nodes, which are not directly linked by links; sequential update refers to the nodes on the old rerouting path used by the IPFRR strategy used in [7] to face transient failure. This helps to decrease the rerouting paths' overloading.

The remainder of this paper is organized as follows. Section 2 presents our update strategy, and section 3 reveals the numerical results of simulations. Finally, section 4 concludes the paper.

II. OUR ROUTING TABLE'S UPDATE MECHANISM IN CENTRALIZED ENVIRONMENT

In order to keep the network consistency after the routing table's update process, our detection mechanism presented in this section satisfies the following constraints contrary to [7]:

- the height of the nominal tree must be minimal;
- the tree obtained after the update of the tables must remain nominal as far as possible;
- a minimum flow weight must be transferred to the rerouting path for each case of persistent link failure.

A. Some Definitions

Before they are used later, let us consider the following definitions:

- Gr : the red part of the network that hosts a failure;
- Gb : the blue coloured part of the network that hosts the destination node of a nominal routing tree;
- N_f : the node that detects a link failure when it occurs;
- *border node*: node of the red part connected directly to a node of the blue part;
- *critical node*: border node nearest to the node that detects a transient failure.

B. Our Critical Nodes Detection's Mechanism

Let us consider a network represented by the graph $G(N, L)$, where N indicates the set of nodes and L the set of links. We consider a persistent link failure (p, q) (see Fig. 1). Let $|N| = \text{Cardinal}(N)$ be the number of nodes and $|L| = \text{Cardinal}(L)$ the number of links. We also assume a node D and destination of the flows of $G(N, L)$. The link failure (p, q) generates the sub-graph Gr , which is a routing tree with the node p as the root (see Fig. 1). For each node of Gr , there is a path linking it to p because of the tree topology. The dotted links indicate the possible presence of a topology. According to [7], there is at least one sub-graph rooted in a node r_i which contains a set of nodes r_i and connects an extremity i of Gr to a node s of Gb : this is the bridge (i, s) . Similarly, according to the configurations, there are sub-trees

N_c of G rooted in k_i nodes ($i=1,2,\dots$) and composed of nodes $k_j \neq i$ that directly connect Gr to G_b .

Our detection strategy uses only one bridge from each branch of the nominal tree Gr , instead of several, in order to keep much as possible the network structure. The search process starts from the node $p = N_f$. We look for the first node of each routing tree's branch, which offers a bridge to reach G_b . This node i is a critical node. Only these critical nodes among the border nodes will be updated in order to reduce the looping's risks in the routing. The exploration of a tree branch stops when a critical node is found.

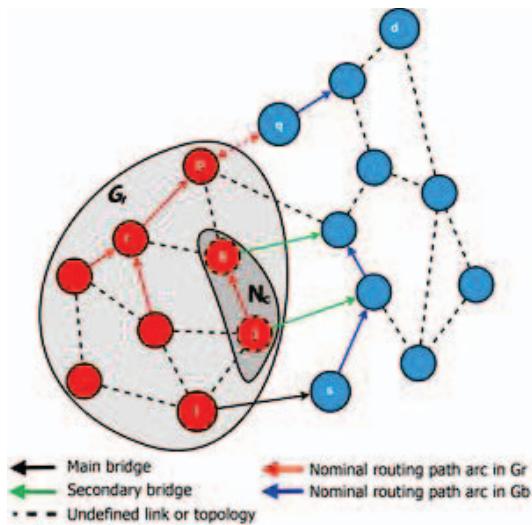


Fig. 1. Our detection mechanism for the nodes to update.

For example, in the case of Fig. 2, node 8 is the extremity of the secondary bridge (8-9) nearest to the node 2, which detects the link failure (2-1); thus, it is a critical node. Only the secondary bridge (8-9) will be exploited instead of (8-9) and (14-13) in the same branch. Node 18 will also be selected as a critical node, and the search process of the critical nodes into the sub-tree rooted in this node will be stopped directly.

This method has the advantage of reducing the flow's weight assigned to the main bridge, which means an availability of more resources for some future link failures' handling. In addition, the search time for the critical nodes is lower.

C. General Update Principle from a Node to Another One

The detection mechanism presented above helps to locate the critical nodes whose routing tables will be updated in addition to the nodes of the rerouting path, which also need to be updated because of rerouting. The critical nodes are all independent of each other because they are located in different tree branches; then, the controller can update those

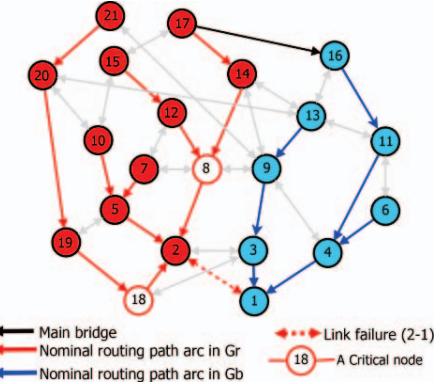


Fig. 2. Application of our detection mechanism on an example.

addition to the nodes of the rerouting path, which also need to be updated because of rerouting. The critical nodes are all independent of each other because they are located in different tree branches; then, the controller can update those critical nodes simultaneously through packet-out messages, without looping risks. On the other hand, with regard to the nodes of the rerouting path, the update will be carried out gradually starting from the extremity node i of the bridge located in Gr towards the node p , which detected the link failure.

Since the routing tables of the nodes in the rerouting path are used for transferring the rerouted flows, the update for these nodes will consist of applying these rerouting configurations in their routing table as the new routing rules. The critical nodes routing tables will be updated in order to allow for these critical nodes to use a secondary bridge; the rerouting paths of these nodes must also be rebuilt by taking into consideration these new routing tables' configurations. Consequently, the rerouting strategies previously developed by the controller will be recomputed by this one and reinstated in the various nodes of Gr . Otherwise, the routing tree would not be nominal any more nor optimal, which would increase the packets routing delays.

By considering Fig. 3, the routing tables updates of the nodes located in the rerouting path will be made according to order 17-15-10-5-2.

On node 17, the flows intended for the arc 17-14 will be deviated towards the bridge 17-16. The arcs such as 15-12 will be updated to 15-17. Finally, we obtain the new nominal tree of Fig. 4.

D. Internal Updates of the Nodes in the Rerouting Path

The structure of the update message: The update message sent by the controller is an updates' lists vector that

contains all the updated information of each node concerned. There is also a value field k , which represents the position of the whole updates' entries associated with a given node. Each entry in the updates' list is a list of triplets

(a, b, c) similar to the structure of LSU messages of the OSPF protocol (see Fig. 5).

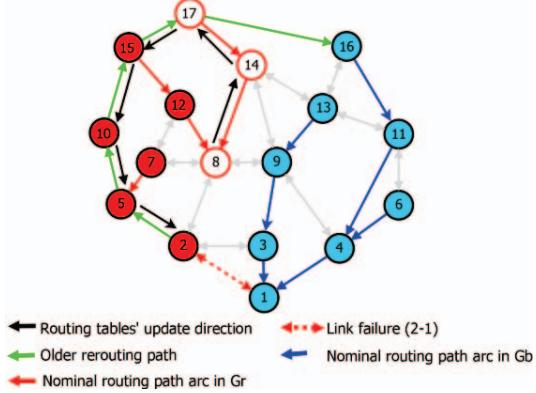


Fig. 3. Routing table's update direction.

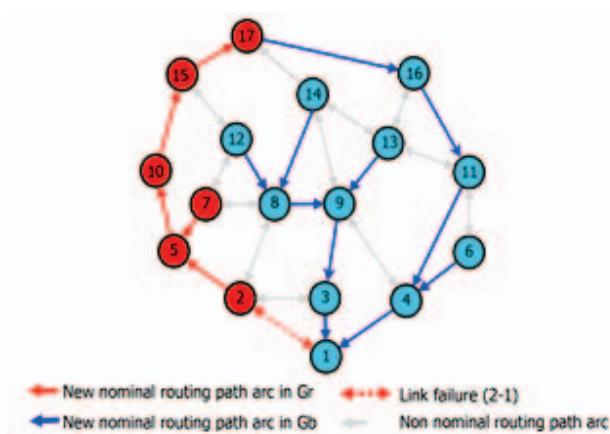


Fig. 4. New routing tree updated with node 1 as destination node.

This figure focuses on the part of the updates' packets that contains the updates' information. For each triplet, a represents the entry gate of the flow to be modified, b is the new exit gate of this flow and c is the exit gate that will be used for forwarding the remainder of the message to the next node in the vector. In case of a switch, c will be a port's MAC address, and in the case of a router, it will be the next hop IP address. For the nodes that need to update many entries in their routing table, only the last triplet in its update list will have the value $c \neq 0$. Each node of the rerouting path that receive an update message do not use it until it receives a permission message from the controller or from the preceding node in the rerouting path; to achieve this latency, the SDN-switches or routers are made of agents as it is the case in [7] with filters. These agents manage

the flow, analyzes the ports status and set off the appropriate behavior provided by the controller. When a node wants to use the update message, it reads the field k 's value, recovers its triplets' list, carries out its updates, then increments the value of k and returns the update message to the next node in the vector. The last triplet list has the value of c equal to 0 for the last triplet in the list; this means the end of update message transmission between the nodes of the rerouting path. Each node, when it receives its update triplets' list, checks if $c = 0$ is in the last triplet; if it is true, then it updates its routing table and destroys the update message, or else it forwards this update message to the next node.

The update process into the nodes from the triplets' list is represented by Fig. 6, which presents the successive updates of nodes 17, 15, 10 and 5 of Fig. 3.

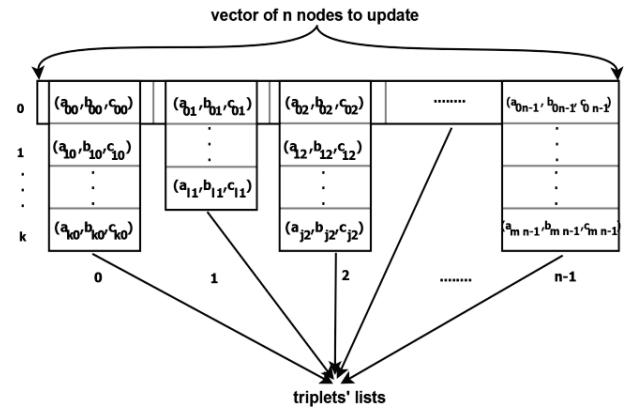


Fig. 5. The updates' triplets list vector structure.

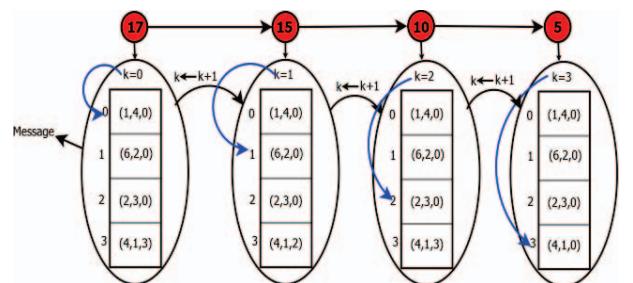


Fig. 6. Update of a node to another one with the triplets' list vector.

1) The Maximum Transmission Unit (MTU) constraint: The MTU is the maximum packet size that can be fully transmitted (without fragmentation) on a network interface. This value helps to determine the packets' data size to

transmit on the networks' interfaces in order to ensure the routing.

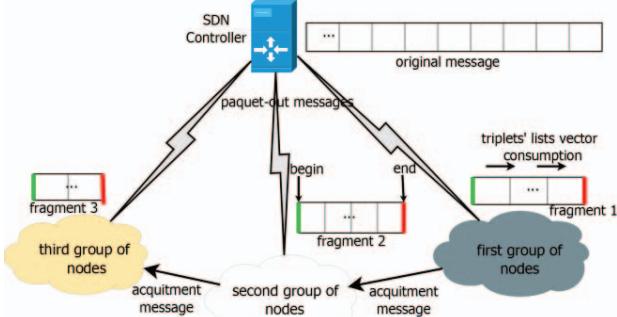


Fig. 7. New routing tree updated with node 1 as destination node.

The update message structure (vector of lists) set up by the controller can quickly grow with a significant number of nodes to be managed in this process, with the risk to quickly pass the MTU value. For example, the MTU values are as follows:

- 68 bytes for IPv4 networks [17]
- 1500 bytes for IPv6 networks [18]

Taking into consideration this MTU value, the update packet of the nodes will be fragmented in smaller packets by the controller before being transmitted. Each fragment will be assigned to its appropriate group of nodes as shown in Fig. 7.

E. Reconstruction of the rerouting tables rules using the new routing configurations

In addition of their routing table's update, the rerouting rules of the nodes must also be updated. This is because these rerouting rules are set up from the routing table's configurations, and when those configurations are updated, the old rerouting rules become obsolete. Thus, the controller builds a virtual nominal routing tree and uses [7] to compute new rerouting rules based on this virtual nominal routing tree. During the computation of the new rerouting rules, the network uses old rerouting rules as in Cisco routers, resulting in poor QoS. Because of the packet routing delays and data loss rate, which are important for the QoS [19] in computer networks, the latencies during the computations must be minimal. That is why the operations for determining the critical nodes and computations of the new routing and rerouting rules must all be done as a preliminary in the controller before initiating any tables update process.

Concerning the routing tables' update and the rerouting rules, they can be done at the same time in each implied node; this is because the controller has already transmitted the information to the nodes, and no additional computations are done by these nodes.

III. IMPLEMENTATION AND SIMULATION RESULTS

The implementation and evaluation of our routing table's update scheme have been done in the OMNet++ network

simulator environment. This network simulator environment has been chosen because it has an extensive graphical user interface (GUI) and intelligence support, provides good computation times and carries out large-scale networks [20]. In addition, the flexibility of the NED language offered by this simulator fully allows for network topology customization even when the simulation is running, which meets the needs of this study. The networks considered for these simulations are network1 (5 nodes and 7 links), network2 (10 nodes and 18 links), and network3 (20 nodes and 31 links). These three networks satisfy the assumptions cited above; they are randomly generated to better appreciate the ability of our update's strategy to work in any network that satisfies our hypothesis. The simulations have been done in a computer with the following configuration: Core i5 2.40 GHz, 4.00 GB RAM, and 12 MB cache.

The objective of our simulations is to make a QoS comparison of the network in the absence of persistent link failure and in the presence of those failures in order to show the efficiency of our routing table's update strategy for the QoS improvement in the presence of failures. With this intention, we carried out simulations on high data rate networks (1.2 Mbits/sec to 512 Mbits/sec) with various scales, and we focus on the packet routing delays.

Consequently, for network1 (5 nodes and 7 links), the analysis of packets' routing delays in various nodes and various network instability situations allowed for us to appreciate the QoS variability. The data were obtained by carrying out several tests on the networks in the following order:

- in the absence of failure;
- in the presence of transient link failure;
- the previous link failure is determined as persistent.

By fixing node 3 as the packets' destination node in network1, the simulation obtains the nominal tree of Fig. 8, where the shortest paths are built using the datarate of the links. The link failure (node[2]-node[3]) yields to the network configuration of Fig. 9, from which the network exploits node 4 as a critical node

By studying the packet delivery time's variation for node 4 (see Fig. 10) and node 1 (see Fig. 11) without fixing common destination for all the packets (as it was the case previously with the node 3), which is closer to real computers' networks, we can note that the difference between the packets' routing delays is lower for the critical nodes

. This is certainly explained by the fact that a critical node is an exit gate for different flows of the local rerouting implemented in the IPFRR; thus, this critical node allows for

the interconnection of the two network parts Gr and Gb generated by the link failure.

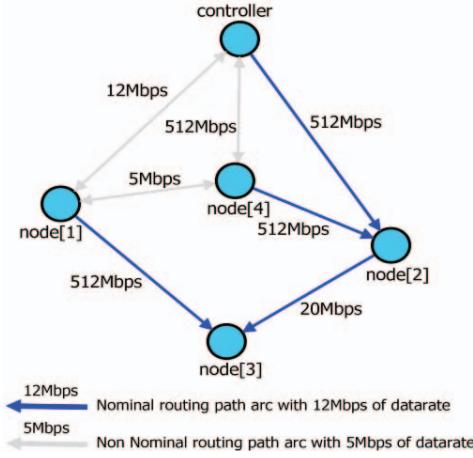


Fig. 8. Routing tree with node 3 as destination node used during the simulations of network1.

Around time $T = 1.16985$, there is an abrupt rise in the packets rerouting delays, which is explained by the fact that a failure occurs at this moment; this enables us to measure the negative impact of the links' failures on QoS in the network.

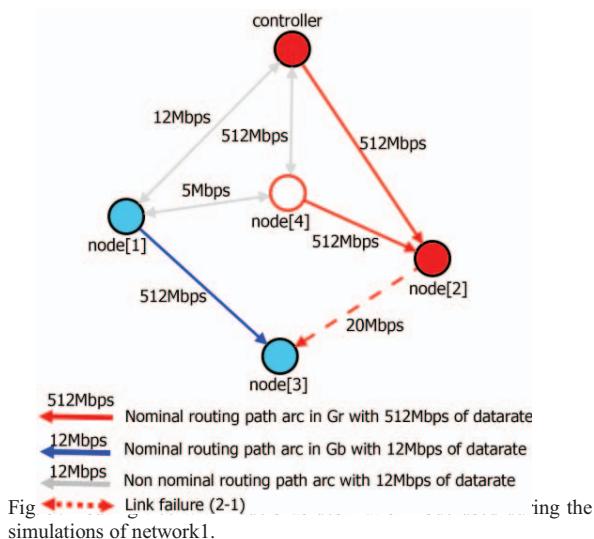


Fig. 9. Routing tree with node 3 as destination node during the simulations of network1 after link failure (2-1).

On the other hand, for the node 1, which is a Gb node, this variation is very significant between the state of the network in the absence of link failure and its state in the presence of link failure. This is because of the link failure that disturbs the traffic.

By analysing the same packets routing delays for network2 (10 nodes) running with a persistent link failure as

above, we obtain similar results with network1 (see Fig. 12), particularly for the case of a fixed destination, which is node 4.

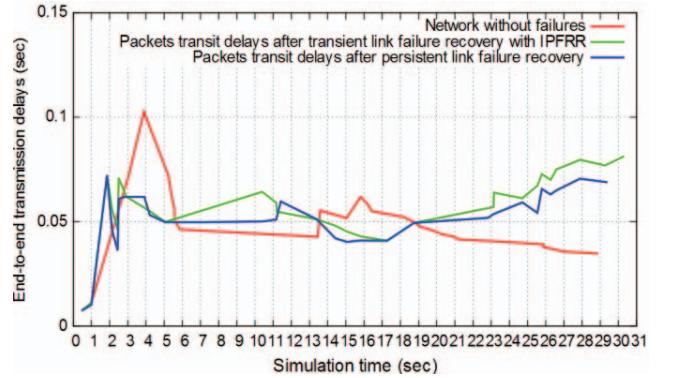


Fig. 10. Packets routing delays' variations for the node 4.

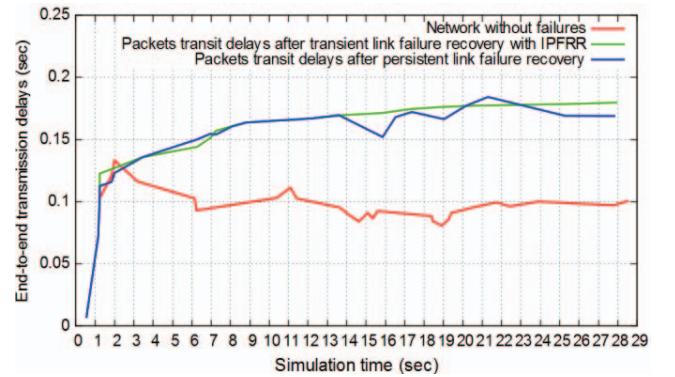


Fig. 11. Packets routing delays' variations for the node 1.

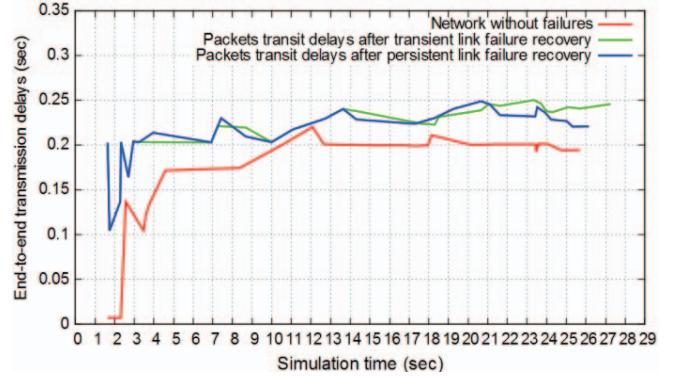


Fig. 12. Packets routing delays' variations in network2 with the node 4 as destination node.

The previous observations done about node 3 of network1 are also verified for node 4 of network2 as shown in Fig. 12; this result suggests that our routing tables' update

strategy improves the packet routing delays in the presence of persistent link failure, independent of the size of the network.

About the data loss rate, Fig. 13 illustrates it for nodes 2 and 3 in network1. Node 2 is the node that detects a link failure, and node 3 is the destination node. A comparison between this data loss rate using the method presented in [7] and the other one using our update strategy reveals that our method provides better results. This means that the data loss rate in our update strategy is lower.

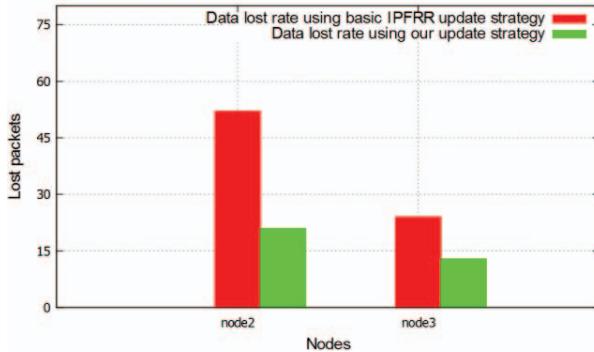


Fig. 13. Comparison of the data lost rate of our update strategy with [7] for network1.

III. CONCLUSION AND FUTURE WORKS

The aim of this paper was to propose a routing and rerouting tables' update mechanism to face nodes' reconfiguration challenges in case of persistent link failure in a computer network. These reconfigurations must be done from a controller, which defines the management policies of the network. Then, we proposed an update mechanism that efficiently targets the nodes to imply in the update process and sets up an update scheme for these nodes. This update scheme uses a triplets' lists vector inspired to OSPF's LSU packet to modify the routing tables of the nodes. The immediate consequence of this strategy as shown by the simulations carried out is a smaller packets routing delay and a lower data loss rate in case of persistent link failure than in the case of transient failures while preserving as much as possible the network structure.

Further works will address the persistent nodes failure problem. It would also be interesting to think about the cascading updates in multiple virtual planes built over a substrate network after a persistent physical link failure.

ACKNOWLEDGMENT

We thank the anonymous reviewers whose valuable comments and suggestions have significantly improved the presentation and the readability of this work.

REFERENCES

- [1] Chowdhury, NM Mosharaf Kabir, and Raouf Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, 2010, pp. 862-876.
- [2] Haider, Aun, Richard Potter, and Akihiro Nakao, "Challenges in resource allocation in network virtualization," *In 20th ITC Specialist Seminar*, vol. 18, no. 2009, ITC, Hoi An Vietnam, 2009.
- [3] Farhad, Hamid, HyunYong Lee, and Akihiro Nakao, "Software-defined networking: A survey," *Computer Networks*, vol. 81, 2015, pp. 79-95.
- [4] Hu, Fei, Qi Hao, and Ke Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, 2014, pp. 2181-2206.
- [5] Azodolmolky, Siamak, Philipp Wieder, and Ramin Yahyapour, "SDN-based cloud computing networking," *In Transparent Optical Networks (ICTON)*, 2013 15th International Conference on, pp. 1-4, IEEE, 2013.
- [6] Rothenberg, Christian Esteve, Marcelo Ribeiro Nascimento, Marcos Rogerio Salvador, Carlos Nilton Araujo Corrêa, Sidney Cunha de Lucena, and Robert Raszuk, "Revisiting routing control platforms with the eyes and muscles of software-defined networking," *In Proceedings of the first workshop on Hot topics in software defined networks*, pp. 13-18. ACM, 2012.
- [7] Pham, Thanh Son, "Autonomous management of quality of service in virtual networks," PhD diss., Université de Technologie de Compiègne, 2014.
- [8] Lim, Azman Osman, Xudong Wang, Youiti Kado, and Bing Zhang, "A hybrid centralized routing protocol for 802.11s WMNs," *Mobile Networks and Applications*, vol. 13, no. 1-2, 2008, pp. 117-131.
- [9] Muruganathan, Siva D., Daniel CF Ma, Rolly I. Bhasin, and Abraham O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks," *IEEE Communications Magazine* 43, no. 3, 2005, S8-13.
- [10] John T Moy, "OSPF : anatomy of an Internet routing protocol," AddisonWesley Professional, 1998.
- [11] Charles Perkins, Elizabeth Belding-Royer, and Samir Das, "Ad hoc on-demand distance vector (aodv) routing," Technical report, 2003.
- [12] Hedrick, Charles L, "Routing information protocol," 1988.
- [13] Albrightson, R., J. J. Garcia-Luna-Aceves, and Joanne Boyle, "EIGRP--A fast routing protocol based on distance vectors," 1994.
- [14] Hu, Fei, Qi Hao, and Ke Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials* 16, no. 4, 2014, pp. 2181-2206..
- [15] Kotronis, Vasileios, Xenofontas Dimitropoulos, and Bernhard Ager, "Outsourcing the routing control logic: better internet routing based on SDN principles," *In Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 55-60, ACM, 2012.
- [16] Papán, Jozef, "IP Fast Reroute," *Information Sciences and Technologies*, vol. 8, no. 2, 2016, p. 1.
- [17] Moy, John T, "OSPF: anatomy of an Internet routing protocol," Addison-Wesley Professional, 1998.
- [18] Crawford, Matt, "Transmission of IPv6 packets over ethernet networks," 1998.
- [19] Seddiki, Mohamed Said, "Allocation dynamique des ressources et gestion de la qualité de service dans la virtualisation des réseaux," PhD diss., Université de Lorraine, 2015.
- [20] Bilalb, Sardar M., and Mazliza Othmana, "A Performance Comparison of Network Simulators for Wireless Networks," arXiv preprint arXiv:1307.4129, 2013.