AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Élimination de l'impact de la résilience réseau dans un transport de flux vidéo par implémentation dans une architecture SDN contrainte par l'existant

THÈSE

présentée et soutenue publiquement le 19 Décembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention Automatique, Traitement du signal et des images, Génie informatique)

par

Constant COLOMBO

Composition du jury

Rapporteurs :	Abdelhamid MELLOUK	Professeur - Université Paris-Est Créteil (UPEC)
	Philippe OWEZARSKI	Directeur de recherche - LAAS-CNRS
Examinateurs :	Marion GILSON	Professeur - Université de Lorraine
	Jean-Jacques LESAGE	Professeur - ENS Paris Saclay
	Janan ZAYTOON	Professeur - Université de Reims Champagne-Ardenne
	Éric GNAEDINGER	Maître de conférences - Université de Lorraine
Directeur :	Francis LEPAGE	Professeur - Université de Lorraine
Co-directeur :	René KOPP	Ingénieur - TDF

CRAN - UMR 7039

TDF - SAS

Mis en page avec la classe thesul.

Remerciements

Je souhaiterais avant toute chose remercier l'ensemble des personnes qui m'ont amené à commencer ou à finir cette thèse.

Tout d'abord je voudrais remercier Francis Lepage, Professeur à l'Université de Lorraine, d'avoir accepté de diriger ces travaux. Puisse sa retraite être aussi riche que sa carrière ! Je remercie également René Kopp, Architecte à TDF, de m'avoir encadré tout au long de ces trois ans. Sa capacité à me ramener aux fondamentaux, tout en considérant un environnement industriel complexe, m'ont permis d'avancer toujours plus efficacement. Ces travaux n'auraient pas été possible sans leur confiance sans faille en moi pour mener cette thèse à bien.

Je souhaiterais également remercier Eric Gnaedinger, Maître de conférence à l'Université de Lorraine, qui m'a introduit au milieu de la recherche et sans qui cette thèse n'aurait pas eu lieu.

Mes remerciements vont également à Abdelhamid Mellouk, Professeur à l'UPEC, et Philippe Owezarski, Directeur de recherche au LAAS, pour avoir accepté d'être rapporteurs de ce travail, ainsi que Jean-Jacques Lesage, Professeur à l'ENS Paris-Saclay, et Janan Zaytoon, Professeur à l'Université de Reims Champagne-Ardenne, pour leur participation au jury en tant qu'examinateurs.

Je suis reconnaissant à l'entreprise TDF d'avoir accepté cette collaboration avec le CRAN, et de m'avoir accueilli au sein de ses locaux au cours de ces trois ans. L'accès aux plateformes et aux données a permis une meilleure compréhension de l'environnement dans lequel ces travaux devaient s'inscrire.

J'aimerai particulièrement remercier toute l'équipe de Metz, pour tous nos échanges professionnels, scientifiques et personnels : Jean, Antoine, Christian, Gérard, Christophe L., Philippe, Christophe K. et Jean-Pierre. Je remercie chaleureusement Sylvain Rondeau, Ingénieur conception à TDF, pour son aide et ses idées face à mes problèmes de transport botanique. Je tiens à remercier Stéphen Pirlot, docteur de l'Université de Lorraine et Ingénieur conception à TDF, qui m'a amené à faire cette thèse, pour sa présence, ses conseils, et son amitié pendant ces trois ans. Ces travaux n'auraient pu aboutir sans leur aide à tous. Merci les Schtroumpfs de Metz.

Je tiens à remercier également le CRAN et tous ses membres, de m'avoir accueilli pour cette thèse. Si ma présence était intermittente, cela n'aura pas empêché des collaborations riches. Je remercie notamment Mohaimenul Hossain, Eric Rondeau, Jean-Philippe Georges et Van Phuc Do, pour nos échanges sur le SDN et les algorithmes génétiques. Mes remerciements vont aussi à Benoit Iung, pour avoir accepté d'être membre de mon comité de suivi au cours de ces trois années.

Mes remerciements vont également à toute l'équipe de Polytech Nancy. Je tiens à remercier les doctorants sur place pour leur accueil et leur sympathie, et en particulier Dawid Machala pour nos échanges scientifiques et artistiques. Do widzenia i powodzenia ! Je remercie également Laetitia Bau, docteur de l'Université de Lorraine, pour son soutien et ses conseils précieux.

Je tiens à remercier chaleureusement Marion Gilson, Professeur à l'Université de Lorraine, pour son soutien sans faille, son aide précieuse et son amitié. Non contente de m'avoir formé pendant mes études d'ingénieur, elle m'a soutenu pendant ces trois ans et m'a offert l'opportunité d'enseigner. C'est un honneur pour moi qu'elle ait accepté d'examiner mon travail en tant que membre du jury.

Ces remerciements ne seraient pas complets sans un mot à l'intention de ma famille et de mes amis. J'aimerais commencer par remercier Quentin Grandemange, docteur de l'Université de Lorraine, pour m'avoir incité à tenter l'aventure de la thèse, ainsi que sa compagne, Elise Vouriot, qui en verra également bientôt le bout. Je remercie également tous mes amis, qui m'ont amené aussi loin sans admettre leur participation : Alexandre, Benjamin, Maxime et Pierre.

J'aimerai ensuite remercier ma grand-mère, notamment pour sa patience lors de mes exposés scientifiques du dimanche midi, et pour savoir me faire redescendre sur terre face à mes problèmes. Parfois, tout peut être aussi simple que d'aller manger avec ceux qu'on aime. Je remercie également mon grand-père pour m'avoir encouragé sans même le savoir. J'espère qu'il me trouverait toujours aussi "fortiche" aujourd'hui.

Mes remerciements vont également à mes parents, sans qui je n'aurai ni l'esprit scientifique ni la force qui m'ont permis de mener ces travaux à bien. Je remercie également mon frère et ma sœur, même de loin. Merci de m'avoir fait grandir, et de m'avoir emmené jusqu'ici.

Je remercie enfin celle qui partage ma vie, et qui a partagé mes réussites et mes échecs au cours de ces trois années. Un soutien émotionnel, un appui scientifique, et une complicité à toute épreuve. Pour avoir partagé tout cela, cette thèse, c'est aussi un peu la sienne.

A ma famille :
ceux dont je viens,
et ceux qui m'ont trouvé.

Sommaire

Table des figures **ix**

Table des algorithmes **xiii**

Préface **1**

Liste des publications **3**

Introduction **5**

Chapitre 1 Les réseaux de transport audiovisuel **9**

1.1 Les flux audiovisuels 10

1.2 Le transport audiovisuel 12

1.3 La multidiffusion 14

1.4 TDF et le réseau TMS 15

1.5 L'ingénierie protocolaire du réseau TMS 17

1.6 Conclusion 18

Chapitre 2 Seamless Multicast **19**

2.1 Approche d'architecture 20

2.1.1 Techniques Seamless 20

2.1.2 Arbres redondants et résilients 20

2.1.3 Contraintes 21

2.1.4 Concept proposé 23
2.2 Redondance : définition formelle d'une paire d'arbre 24
2.2.1 Vocabulaire de théorie des graphes 24
2.2.2 Le problème DMDT 25
2.2.3 $\mathcal{NP}$-difficulté 29
2.3 Résilience : mécanisme de déploiement dynamique 30
2.3.1 État de l'art 30
2.3.2 Description des états du système 31
2.3.3 Mécanismes de résilience envisagés 32
2.3.4 Mécanisme dynamique proposé 34
2.3.5 Temps de réaction et phénomènes de bagotage 35
2.4 Conclusion 36

Chapitre 3 Algorithmique associée au problème DMDT **37**

3.1 Etat de l'art des approches par algorithmes d'exploration 39
3.2 Prérequis algorithmiques 40
3.2.1 Plus court chemin 40
3.2.2 Plus court chemin contraint en délai 40
3.2.3 Chemins totalement indépendants 41
3.2.4 SHERPA : Plus courts chemins contraints en Délai à Indépendance Maximale 43
3.3 Heuristique 1 : Iterative SHERPA (IS) 46
3.3.1 Procédé 46
3.3.2 Complexité temporelle 49
3.3.3 Variante à indépendance de nœud 49
3.3.4 Variante multi-source 49
3.4 Heuristique 2 : Red Tree First (RTF) 50
3.4.1 Procédé 50

3.4.2 Complexité temporelle . . . 53
3.4.3 Variante à indépendance de nœud . . . 53
3.4.4 Variante multi-source . . . 53
3.5 Raffinage des pseudo-arbres . . . 54
3.5.1 Procédé . . . 54
3.5.2 Complexité . . . 58
3.6 Algorithmes génétiques . . . 60
3.6.1 Introduction . . . 60
3.6.2 Etat de l'art des approches par algorithme génétique . . . 61
3.6.3 Proposition d'un algorithme génétique . . . 62
3.6.4 Réglage des paramètres . . . 67
3.7 Expérimentations . . . 71
3.7.1 Temps de calcul des heuristiques . . . 71
3.7.2 Comparaison à des pseudo-optimums . . . 73
3.7.3 Algorithme génétique appliqué . . . 75
3.8 Conclusion . . . 78

Chapitre 4 Implémentation Software-Defined Networking 79

4.1 Le Software-Defined Networking . . . 80
4.1.1 Présentation . . . 80
4.1.2 Le protocole OpenFlow . . . 82
4.1.3 Etat de l'art relatif au transport audiovisuel en SDN . . . 84
4.2 Architecture réseau proposée . . . 85
4.3 Description technique de l'implémentation . . . 86
4.4 Protocole expérimental . . . 89
4.4.1 Réseau d'expérimentation . . . 89
4.4.2 Paramètres du contrôleur . . . 89
4.4.3 Règles de commutation . . . 90

4.4.4 Émulation de récepteurs Seamless 91
4.4.5 Déroulement du protocole 91
4.5 Résultats 92
4.5.1 Panne impactant un seul arbre, indépendance stricte 93
4.5.2 Panne impactant un seul arbre, indépendance maximale 96
4.5.3 Panne impactant les deux arbres 98
4.5.4 Création d'une nouvelle liaison 100
4.6 Conclusion 102

Conclusion **103**

Annexes **105**

Annexe A Modélisation du Seamless Multicast à l'aide d'un réseau d'automates finis **105**

A.1 Automates finis 105
A.2 Uppaal 105
A.3 Modélisation 106
A.4 Vérification 110

Annexe B Interface du contrôleur SDN **111**

Glossaire **113**

Bibliographie **117**

Table des figures

1.1 Principe d'acheminement d'un flux audiovisuel 10
1.2 Processus complet d'acheminement d'un flux audiovisuel 11
1.3 Intérêt de la multidiffusion 12
1.4 Illustrations des notions de chemin, arbre et arbre invalide 14
1.5 Hiérarchie du réseau TMS 16
1.6 Principe d'architecture MVPN exploitant PIM et IGMP 17

2.1 Illustration du principe de réception "Seamless" 20
2.2 Illustration du concept de transport redondant et résilient 21
2.3 Illustration du concept d'indépendance maximale 22
2.4 Exemples de calcul de disponibilité d'un chemin 23
2.5 Illustrations des notions de graphe non-orienté et orienté 24
2.6 États possibles pour une destination 31
2.7 États possibles des arbres composants du système 31
2.8 États possibles du système 32
2.9 Exemple de cas de pannes impactant les deux arbres 33
2.10 Mécanisme de dynamisation de l'architecture 34
2.11 Exemple d'application du mécanisme de résilience 35

3.1 Illustration de l'algorithme SHERPA 43
3.2 Illustration algorithme IS 47
3.3 Illustration algorithme RTF 51
3.4 Illustration algorithme de raffinage 55
3.5 Principe de fonctionnement d'un algorithme génétique 61
3.6 Encodage "Path-oriented" 62
3.7 Encodages des deux arbres 62

3.8 Exemple d'encodage concaténé 63
3.9 Génération d'un individu aléatoire 64
3.10 Croisement par paires 65
3.11 Mutation aléatoire d'un individu 65
3.12 Encodage des paramètres pour le meta-AG 69
3.13 Influence du nombre de destinations sur le temps de calcul des heuristiques 72
3.14 Différence de qualité de solution fournie par les heuristiques suivant le nombre de destinations 73
3.15 Qualité de solution fournie par les heuristiques par rapport à des tracés existants 74
3.16 Moyenne des fonctions d'évaluations des meilleures solutions de l'AG pour $S = 0$ suivant les jeux de paramètres 76
3.17 Moyenne des fonctions d'évaluations des meilleures solutions de l'AG pour $S = 1$ suivant les jeux de paramètres 77

4.1 Comparaison des architectures traditionnelles et SDN 81
4.2 Exemples d'applications de règles OpenFlow 82
4.3 Modes de remplissage de la table des Flows dans le protocole OpenFlow . . 83
4.4 Architecture réseau 85
4.5 Architecture de l'implémentation du contrôleur SDN 87
4.6 Principe de fonctionnement du module de découverte de topologie 88
4.7 Principe de fonctionnement du module de mesure de latence 88
4.8 Exemple de déploiement de règles OpenFlow définies dans le cadre expérimental 90
4.9 Émulation de réception Seamless 91
4.10 Principe expérimental vérification du comportement de l'implémentation . 92
4.11 Scénario A : Panne impactant un seul arbre, indépendance stricte possible 94
4.12 Résultats du scénario A 95
4.13 Scénario B : Panne impactant un seul arbre, indépendance maximale . . . 96
4.14 Résultats du scénario B 97
4.15 Scénario C : Panne impactant les deux arbres 98
4.16 Résultats du scénario C 99
4.17 Scénario D : Nouveau lien permettant une meilleure situation 100
4.18 Résultats du scénario D 101

A.1 Patron du plan de contrôle sur UPPAAL 108
A.2 Patron du plan de données sur UPPAAL 109

B.1 Illustration de l'interface du contrôleur développé 111

Table des algorithmes

3.1 Algorithme de Suurballe-Tarjan 42
3.2 SHERPA - Sharing-Edges Restrained Paths 45
3.3 IS - Iterative SHERPA 48
3.4 RTF - Red Tree First 52
3.5 Algorithme de Raffinage IS et RTF 56
3.6 Fonction de Chemin unique 57

Préface

Les travaux présentés dans cette thèse se sont déroulés dans le cadre d'une collaboration CIFRE entre le laboratoire CRAN et l'entreprise TDF.

CRAN

Le CRAN est une unité mixte de recherche (UMR 7039) créée en 1980, commune à l'Université de Lorraine (UL) et au CNRS (Sections 7, 26 et 28). Il accueille également des chercheurs de L'Institut de Cancérologie de Lorraine (ICL, Centre de lutte contre le cancer) et du CHU.

Les recherches menées au CRAN concernent l'Automatique, définie comme la science de la modélisation, de l'analyse, de la commande et de la supervision des systèmes dynamiques, mais aussi le traitement du signal et le génie informatique. Les applications et retombées de ces recherches concernent également des domaines transverses, notamment la santé ou la sûreté de fonctionnement des systèmes.

TDF

TDF est un opérateur d'infrastructures opérant un réseau de plus de 18000 sites de diffusion audiovisuelle répartis sur la Métropole et en Outre-Mer. Ses activités sont essentiellement liées à la TNT, la diffusion de stations radios, la VoD, les télécoms, les réseaux fibre et les datacenters.

Le groupe TDF est issu de TéléDiffusion de France, une entité résultant de la division de l'Office de Radiodiffusion-Télévision Française (ORTF) en 1975, et devient une société anonyme financée par des fonds privés en 1987. TéléDiffusion de France appartient alors au groupe France Télécom jusqu'en 2004 où le groupe cède ses parts, et l'entreprise est alors renommée simplement "TDF". Cet historique explique en partie l'importance de l'entreprise dans le marché des opérateurs français.

Liste des publications

Brevet

1. C. Colombo et R. Kopp, *"Procédé de configuration d'une architecture réseau et architecture réseau associée,"* INPI FR1855411, June 19, 2018.

   Le contenu de ce brevet correspond aux contributions des chapitres 2 et 3.

Conférences

1. C. Colombo, F. Lepage, R. Kopp and E. Gnaedinger, *"SHERPA : a SDN Multipath Approach to Eliminate Resilience Impact on Video Streams,"* 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, China, 2018.

   Cette publication décrit l'algorithme SHERPA proposé dans ces travaux de thèse en section 3.2.4.

2. C. Colombo, F. Lepage, R. Kopp and E. Gnaedinger, *"Two SDN Multi-tree Approaches for Constrained Seamless Multicast,"* 2018 IEEE 21st International Conference on Computational Science and Engineering (CSE), Bucharest, Romania, 2018.

   Cette publication décrit les heuristiques IS et RTF proposés dans ces travaux de thèse respectivement en sections 3.3 et 3.4.

Posters

1. C. Colombo, F. Lepage, R. Kopp and E. Gnaedinger, *"Elimination of network resilience impact in a video stream transport by implementing an SDN architecture constraint by existing network,"* RESCOM 2017 Summer School on Virtualization and Dehardwarization, GDR CNRS RSD, Le Croisic, France, 2017 (Poster).

2. C. Colombo, F. Lepage, R. Kopp and E. Gnaedinger, *"Elimination of network resilience impact in a video stream transport by implementing an SDN architecture constraint by existing network,"* APIL 2018 - Annual PhD students conference IAEM Lorraine Nancy, France, 2018 (Poster).

3. C. Colombo, F. Lepage, R. Kopp and E. Gnaedinger, *"Elimination of network resilience impact in a video stream transport by implementing an SDN architecture constraint by existing network,"* SDN DAY 2018 "IDNs (Intelligence Defined Networks)" - TincNET, Paris, France, 2018 (Poster).

Introduction

Apparue dans les années 2000 suivant différents standards, la Télévision Numérique Terrestre (TNT) est une technologie de diffusion de signaux de télévision basée sur un réseau d'émetteurs hertziens. Le réseau TNT français déployé en 2005 s'est développé, remplaçant progressivement la diffusion analogique jusqu'en 2011. Depuis 2016, le réseau permet la diffusion en Haute Définition pour les chaînes gratuites.

La société TDF a été un acteur majeur du développement des réseaux TNT en France, et opère aujourd'hui un réseau s'étendant sur le territoire métropolitain et en Outre-mer. Ce réseau assure non seulement la diffusion de chaînes audiovisuelles du service public et des émissions radio, mais aussi des chaînes privées et des diffusions événementielles. Les opérateurs de réseaux de diffusion audiovisuelle comme TDF doivent répondre à un ensemble de contraintes contractuelles vis-à-vis de leurs clients. Ces contraintes sont définies dans les engagements de qualité et de performance de service (Service Level Agreement - SLA). Les réseaux de transport audiovisuel ont notamment pour but le transport de contenu en temps réel d'un site source de type studio vers de multiples sites de diffusion, desservant des milliers voire des millions de téléspectateurs ; d'où une très forte exigence notamment sur le taux de disponibilité. L'existence d'un réseau dédié à la TNT permet plus facilement le respect des contraintes définies par le SLA. En effet les infrastructures y sont optimisées pour le transport des flux audiovisuels en temps réel.

La Qualité d'Éxperience (Quality of Experience - QoE) est définie par l'ITU-T dans [1] comme le degré de satisfaction ou de contrariété d'un utilisateur d'une application ou d'un service. Dans le cas de la diffusion audiovisuelle, c'est l'expérience de l'utilisateur final, c'est-à-dire sa qualité de perception du contenu. Ce sont les indicateurs de QoE qui sont à l'origine du SLA que doit respecter le réseau. Toujours d'après l'ITU-T dans [2], l'ensemble des caractéristiques d'un service permettant de répondre aux besoins et contraintes forment la Qualité de Service (Quality of Service - QoS). Il existe différentes méthodes de mesure de QoE, et différentes techniques de QoS [3]. Plus spécifiquement, dans le contexte du transport de flux audiovisuel, c'est la robustesse d'un réseau et de ses composants qui permet d'améliorer le taux de disponibilité des services. En effet, les chaînes techniques audiovisuelles peuvent multiplier l'effet d'un impact sur le réseau. Il est alors aussi important d'éviter ces impacts que de réduire leur effet au minimum.

À cette fin, différentes architectures réseau et ingénieries protocolaires existent, exploitant des approches de reroutage, pour minimiser les conséquences d'une panne réseau, ou de redondance, pour en diminuer le risque. Dans le cas particulier de la multidiffu-

sion, l'ingénierie Multicast VPN (MVPN) [4] est un standard dans l'industrie. Il existe différentes implémentations de cette ingénierie. Les travaux présentés dans [5] décrivent le cas spécifique du réseau exploité par TDF. Sont également décrits l'ingénierie MVPN en place ainsi qu'une version améliorant la disponibilité d'un service de multidiffusion sur le réseau.

L'objectif des travaux présentés dans cette thèse est d'améliorer la QoE des services de multidiffusion de contenu audiovisuel sur un réseau à portée nationale. En particulier, on cherchera à inhiber les effets que peut avoir un évènement réseau sur le transport d'un signal audiovisuel d'une source vers de multiples destinations. En effet, les mécanismes existants sur les réseaux traditionnels comportent un temps de retour à un état stable appelé "temps de convergence". Durant ce temps non-nul, le contenu du flux audiovisuel est perdu. Le flux transporté étant un flux en temps réel, les pertes ne peuvent pas être traitées par les mécanismes de retransmission et les mémoires tampon trop importantes. Ainsi, ces travaux proposent une nouvelle approche d'architecture protocolaire pour résoudre les limitations des mécanismes existants dans les réseaux traditionnels.

Dans un système, la redondance se définit comme l'utilisation d'éléments en plusieurs exemplaires pour assurer le fonctionnement en cas d'évènement impactant ou non le système. L'architecture proposée dans ces travaux de thèse consiste en une redondance du flux audiovisuel transporté. En effet, il existe des mécanismes dits "Seamless" capable de recomposer un contenu à partir de deux flux ou plus, même si l'un de ces flux est manquant ou dégradé. La proposition d'architecture assure le transport redondant du flux audiovisuel vers un ensemble d'équipements de réception Seamless. Cela implique le calcul d'un double tracé sur la topologie réseau. Dans le cas d'une multidiffusion, on parle alors d'une paire d'arbres de multidiffusion. Ces arbres devront être les plus indépendants possibles pour assurer une plus grande redondance, mais également optimisés vis-à-vis de l'utilisation des ressources du réseau. Les travaux présentés dans cette thèse proposent différents algorithmes pour calculer les tracés nécessaires au transport redondant et optimisé de flux audiovisuels.

Néanmoins, en cas de pannes multiples sur le réseau, la redondance peut ne pas être suffisante pour assurer la continuité de service. La résilience d'un système se définit comme la capacité du système à retourner à un état fonctionnel après un évènement impactant le système. Ainsi, un mécanisme de dynamisation du déploiement des arbres est proposé en association à la redondance pour assurer en cas d'évènement réseau que non seulement un flux arrive à destination, mais également qu'il soit rapidement sécurisé. L'architecture proposée, combinant redondance et résilience, est nommée "Seamless Multicast".

Pour implémenter ce principe de Seamless Multicast, l'utilisation d'une technologie réseau émergente est proposée : le Software-Defined Networking (SDN). Cette architecture réseau permet la collecte d'informations sur l'état du réseau ainsi que l'activation d'algorithmes et protocoles de manière centralisée. Le contrôleur réseau proposé, collectant les informations, exploite les algorithmes ainsi que le mécanisme de dynamisation développés dans les travaux de cette thèse pour déployer sur un réseau un service de transport résilient et redondant.

Le Chapitre 1 de cette thèse présente plus en détail les réseaux de transport audio-visuels et les solutions existantes de la multidiffusion pour aborder leurs caractéristiques particulières. De plus les missions de TDF et le réseau associé à cette étude y seront présentées.

Afin de répondre à ces besoins et contraintes spécifiques, une nouvelle approche est nécessaire. La proposition principale appelée Seamless Multicast est l'objet de ces travaux. Le principe de cette architecture réseau fait l'objet du Chapitre 2. Il s'agit d'exploiter une paire d'arbres de diffusion redondants de manière résiliente pour assurer la continuité de service. Le mécanisme de résilience proposé est présenté dans ce chapitre

Les éléments algorithmiques composant le Seamless Multicast sont quant à eux détaillés dans le Chapitre 3. En particulier trois algorithmes visant le même objectif de routage point-à-multipoint sont présentés, ainsi que leur évaluation théorique et expérimentale.

Le Chapitre 4 présente un état de l'art du SDN, et décrit comment cette technologie permet à l'approche Seamless Multicast d'être implémentée de manière centralisée. Une implémentation détaillée des concepts décrits est présentée et utilisée pour évaluer expérimentalement le comportement de l'architecture réseau proposée.

Chapitre 1

Les réseaux de transport audiovisuel

Sommaire

1.1 Les flux audiovisuels 10
1.2 Le transport audiovisuel 12
1.3 La multidiffusion 14
1.4 TDF et le réseau TMS 15
1.5 L'ingénierie protocolaire du réseau TMS 17
1.6 Conclusion 18

Ce chapitre présente le contexte global de ces travaux. D'une part les flux audiovisuels y sont définis au sein du processus complet d'acheminement de contenu audiovisuel, et les contraintes liées à ce processus sont évoquées. D'autre part, le réseau étudié et son fonctionnement sont présentés dans ce chapitre afin de faire apparaître la problématique abordée par ces travaux de thèse.

1.1 Les flux audiovisuels

On appelle flux audiovisuel une suite de données dans le temps qui encode une information audio ou vidéo. Si à l'origine ces flux étaient analogiques, c'est-à-dire des signaux continus, ils sont aujourd'hui en grande majorité numériques, autrement dit des valeurs discrètes. La figure 1.1 illustre le principe d'acheminement d'un flux audiovisuel. L'origine du flux est généralement un appareil de captation audio ou vidéo (caméra, microphone...). Le flux est ensuite préparé pour sa prise en charge sur un réseau de transport, pour atteindre des points de diffusion qui vont émettre jusqu'aux équipement de restitution audio ou vidéo (téléviseur, poste radio...).

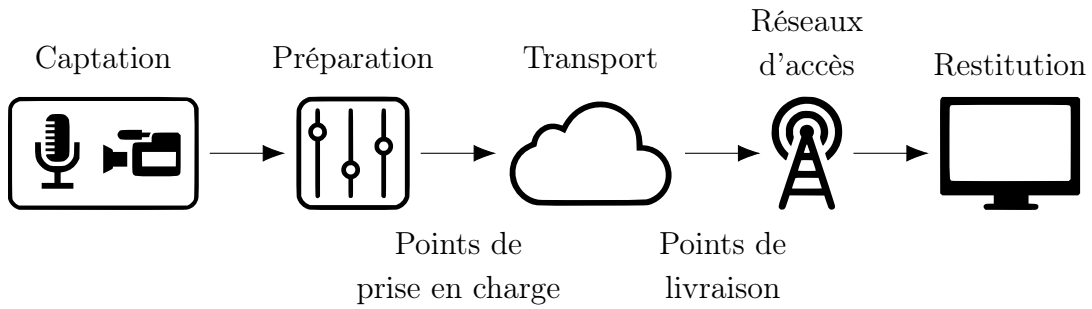


FIGURE 1.1 – Principe d'acheminement d'un flux audiovisuel

Ainsi, un flux audiovisuel est une suite ordonnée de valeurs discrètes. On appelle bande de base un tel flux s'il n'a pas encore été compressé et s'il ne comprend pas de signalisation. Par exemple, à la sortie d'un équipement de captation type caméra ou microphone on trouve un flux audiovisuel bande de base.

Ces flux peuvent être compressés, on parlera alors de codage. Les normes ISO/IEC MPEG définissent notamment des formats de compression et de signalisation pour les flux bande de base.

Les flux audiovisuels codés ne correspondent chacun qu'à un seul contenu. Dans le cas particulier de la diffusion, différentes chaînes peuvent êtres regroupées dans un seul flux qu'on appelle alors multiplexe, en y ajoutant des informations de contenu et de signalisation (sous-titre, filtrage de contenu, tables DVB, etc.). Par exemple, la norme DVB est utilisée pour exploiter des flux codés avec les normes ISO/IEC MPEG afin de créer les multiplexes des chaînes de la TNT en France ; SPTS (Single Program Transport Stream) contenant une seule chaîne, MPTS (Multi Program Transport Stream) comprenant plusieurs chaînes multiplexées. Ces flux audiovisuels sont appelés flux de transport, car ils

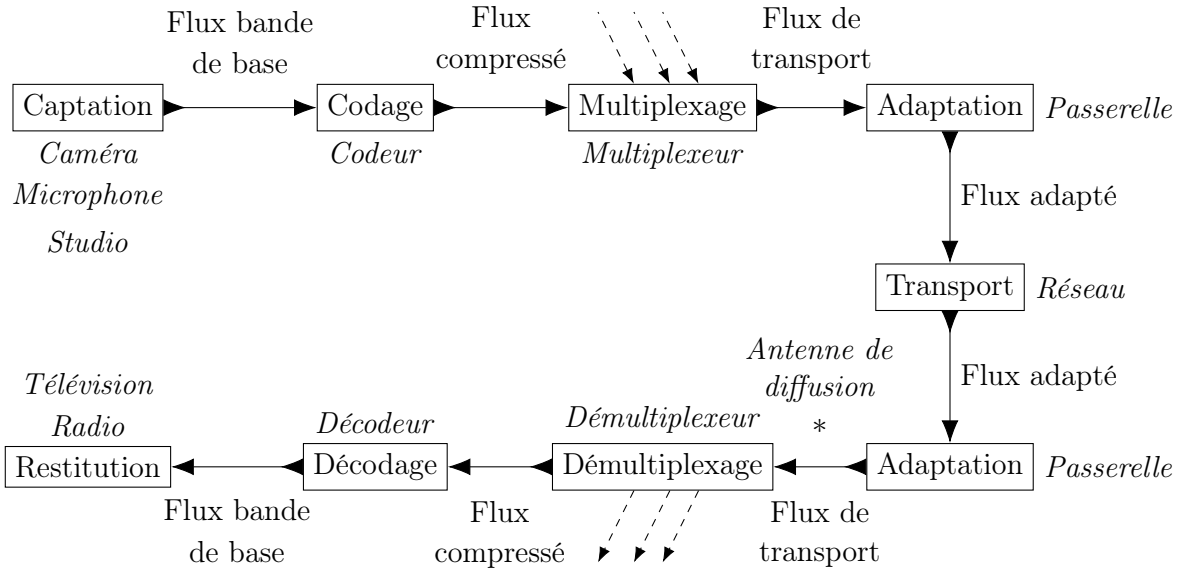


FIGURE 1.2 – Processus complet d'acheminement d'un flux audiovisuel

peuvent être transportés tels quels vers les points de diffusion, une fois adaptés à la technologie du réseau de transport sous-jacent ; dans le cas du réseau TDF, une encapsulation IP/MPLS.

Les flux audiovisuels de transport peuvent être acheminés à travers un réseau via une fonction passerelle. La passerelle est le point de démarcation et d'adaptation reliant deux réseaux de nature différente. Elle permet la conversion du flux vers un format correspondant au réseau de transport. Elle peut également embarquer des fonctions de correction d'erreur, de métrologie, ainsi que de démarcation réseau.

On peut ainsi suivre une illustration du processus complet d'acheminement d'un flux dans la figure 1.2. Ce schéma résume le principe général, dont les composants peuvent être regroupés ou divisés. Par exemple pour la TNT, le démultiplexage, le décodage et la restitution sont aujourd'hui effectuées par le téléviseur.

Dans cette thèse, on s'intéresse uniquement à la partie transport du processus de diffusion. Pour en simplifier la lecture, le terme de flux sera employé pour désigner un flux audiovisuel de transport.

1.2 Le transport audiovisuel

Les réseaux de transport pour la diffusion audiovisuelle hertzienne présentent plusieurs spécificités liées aux objectifs de couverture d'une part et d'autre part par la nature du flux transporté.

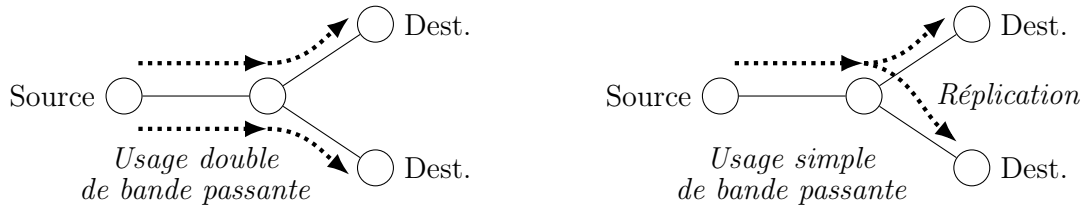
En particulier en France, la loi no. 2007-309 [6] précise dans l'article 96-2 :

> "Les éditeurs de services nationaux de télévision [...] assurent la diffusion de leurs services par voie hertzienne terrestre en mode numérique auprès de 95 % de la population française [...]"

Cette obligation légale des éditeurs se reporte sur leurs besoins en transport audiovisuel. En d'autres termes, le réseau de transport doit couvrir largement le territoire pour leur permettre d'atteindre cet objectif de couverture en population.

Ainsi pour le réseau de transport, cela signifie également un déploiement géographique cohérent avec la nature du terrain. Des supports de transport de type FH (Faisceau Hertzien) sont notamment employés pour joindre des points de diffusion isolés et non-raccordables à la fibre optique. Ces points hauts peuvent être situés en zone montagneuse mais aussi dans des zones éloignées des artères de communication. À noter que les supports de transmission de type FH présentent une bande passante limitée, typiquement de 40 à 150 Mbps. Il existe donc une problématique de gestion de la bande passante. Le débit d'un seul flux multiplexé est généralement compris entre 7 et 80 Mbps. Une valeur usuelle sur le réseau étudié est 30 Mbps.

Le réseau transporte un même flux vers de multiples points géographiques depuis la source. On parle de transport point-à-multipoint, ou multidiffusion. Cela implique l'utilisation de techniques de mutualisation de liens pour une meilleure exploitation de la bande passante sur le réseau. Les protocoles de type MVPN [4] sont des standards industriels sur les réseaux IP traditionnels. La figure 1.3 illustre l'intérêt des techniques de multidiffusion, en particulier de la réplication de données. Grâce à la réplication, la bande-passante des liens du réseau peut être mutualisée pour joindre des destitions différentes.



(a) Diffusion simple (Unicast)

(b) Multidiffusion (Multicast)

FIGURE 1.3 – Intérêt de la multidiffusion

Dans le cadre de la diffusion audiovisuelle, les flux sont continus et doivent être acheminés en temps réel. Le réseau de transport doit alors respecter des contraintes de performance importantes. En effet, si une donnée est perdue durant le transport, elle ne peut être ré-émise à cause de la contrainte temps réel, et la continuité du flux sera interrompue. Dans un transport classique, des mécanismes de retransmission comme on peut en trouver dans le protocole TCP sur les réseau IP existent. Ils ne sont pas compatibles avec un flux de type temps réel. Suivant la quantité et l'importance des données perdues, les autres équipements du processus de diffusion peuvent compenser la perte -par exemple avec des codes correcteurs- ou en multiplier l'impact à cause d'une trop grande perte de continuité.

Il faut que le réseau de transport assure un taux de pertes de données minimal, ainsi qu'une latence et une gigue faibles. Le taux de perte est la quantité de données perdues relativement à la quantité de données acheminées par le réseau. La latence est le délai d'acheminement, qui doit être suffisamment faible pour correspondre aux prérequis d'une transmission temps réel. La gigue est la variation de latence des différents paquets de données. Elle doit être faible pour minimiser le besoin de mémoires tampons dans les équipements et réduire la latence globale de transport.

Les engagements sur la disponibilité d'un service sont généralement de 99,999% sur l'année, mais peuvent aller jusqu'à 100% sur certains créneaux horaires. Ces taux de disponibilité sont associés à des garanties de temps rétablissement (GTR) qui peuvent descendre jusqu'à 1 heure. Des exigences de QoE existent également et sont mesurées par les impacts à la restitution.

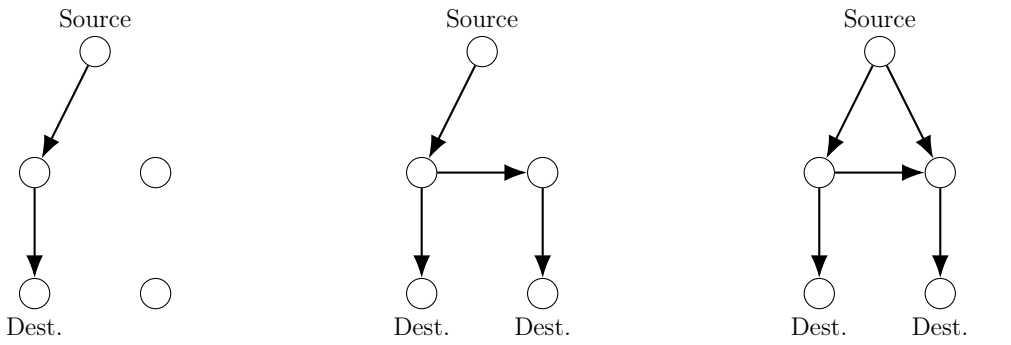
Ainsi, pour répondre aux besoins et contraintes de la diffusion audiovisuelle, il faut déployer sur un réseau des protocoles et des outils de QoS appropriés.

1.3 La multidiffusion

Comme mentionné précédemment en 1.2, la multidiffusion est une technique de communication point-à-multipoint exploitant la réplication de données pour mutualiser les ressources réseau.

La notion de "groupe" définit un ensemble de destinataires qui cherchent à obtenir un flux. Un même protocole de multidiffusion peut alimenter plusieurs groupes à partir de différentes sources.

Un groupe est associé à un "arbre" de diffusion, c'est-à-dire un ensemble de liens joignant la source à toutes les destinations du groupe, comme illustré dans la figure 1.4. Les liens de l'arbre transportent le flux jusqu'aux destinations à travers des chemins uniques à chaque destination. Autrement dit, un arbre est un ensemble de chemins uniques vers chacune de ces destinations, et il ne doit donc présenter aucune boucle. Il est à noter que par définition, un chemin est un arbre qui ne joint qu'une seule destination.



(a) Chemin (b) Arbre (c) Arbre invalide à chemins multiples

FIGURE 1.4 – Illustrations des notions de chemin, arbre et arbre invalide

La multidiffusion est généralement abordée au niveau transport. Autrement dit, le terme désigne les techniques réseau permettant la multidiffusion. [7] définit des caractéristiques permettant de les classifier, notamment la propagation des données, la capacité de retransmission, ou encore les mécanismes de fiabilité.

Les protocoles MVPN [4] sont des standards industriels sur les réseaux IP traditionnels. On peut citer la combinaison des protocoles PIM [8] et IGMP [9], ou encore MPBGP [10].

Des éléments de QoS apparaissent dans les différentes techniques de multidiffusion [11]. Elles sont notamment liées à des problématiques de résilience et de robustesse des protocoles.

Il existe également des approches de multidiffusion au niveau application [12], qui peuvent se fonder ou non sur des techniques de multidiffusion au niveau transport. Dans cette thèse, l'approche multidiffusion au niveau transport sera exclusivement abordée.

1.4 TDF et le réseau TMS

La société TDF opère un réseau de transport des multiplexes de la TNT appelé TMS. Si d'autres services sont transportés sur ce réseau, les flux audiovisuels des chaînes de la TNT restent le cœur de métier de l'opérateur. Cela explique les spécificités de ce réseau essentiellement dédié à cet usage.

Le réseau TMS est un réseau IP/MPLS traditionnel, optimisé pour les contraintes du transport de flux temps réel. Pour le transport de contenu en multidiffusion, une architecture MVPN [4] est utilisée, fondée sur les protocoles PIM [8] et IGMP [9]. La section 1.5 présente l'architecture utilisée. Une analyse détaillée de cette couche protocolaire sur le réseau TMS est disponible dans [5].

En termes de couche physique, la couverture géographique implique l'usage de différentes technologies physiques sur la topologie.

Pour couvrir de grandes distances, des fibres optiques (FO) sont utilisées. Ces FO peuvent être louées à d'autres opérateurs ou appartenir à TDF. On parle de Fibres Optiques Eclairées (FOE) si l'opérateur souscripteur ne choisit pas les longueurs d'ondes utilisées, et de Fibres Optiques Noires (FON) si l'opérateur souscripteur ou propriétaire doit émettre une longueur d'onde de son choix sur la fibre. Des transmissions par Faisceau Hertzien sont parfois imposées par l'absence d'autres moyens de transmission, ou préférées pour des raisons économiques ou de diversité de technologie et de chemin. Les performances de ces transmissions peuvent dépendre du terrain, des conditions climatiques, des obstacles... Enfin, pour des liaisons locales entre des équipements d'un même site, des liaisons cuivre peuvent être employées. De plus, suivant l'importance des nœuds du réseau ainsi que pour des raisons économiques, les équipements utilisés ne sont pas toujours de même catégorie.

Cette hétérogénéité de technologies de transmission et de gammes d'équipements introduit d'une part une hétérogénéité de performances sur le réseau en termes de bande passante, latence, gigue, taux de perte, taux d'erreur. D'autre part, des démarcations d'ingénieries protocolaires existent sur la topologie du réseau.

Si les technologies composant la couche physique sont contraintes pour des raisons technologiques, l'architecture du déploiement du réseau est pensée en termes de couverture et de redondance. L'architecture du réseau TMS est hiérarchique et construite sur cinq boucles. Le cœur de réseau densément maillé couvre la région Parisienne, où les studios de captation des chaînes nationales sont généralement situés. Les boucles couvrent le territoire national, auxquelles sont rattachées des boucles régionales reliées aux différents sites émetteurs. Ces sites émetteurs comportent des antennes diffusant les multiplexes aux postes de télévision sur leur zone de couverture. La figure 1.5 présente cette architecture hiérarchique. L'utilisation de boucles assure une redondance de chemins vers les sites de diffusion.

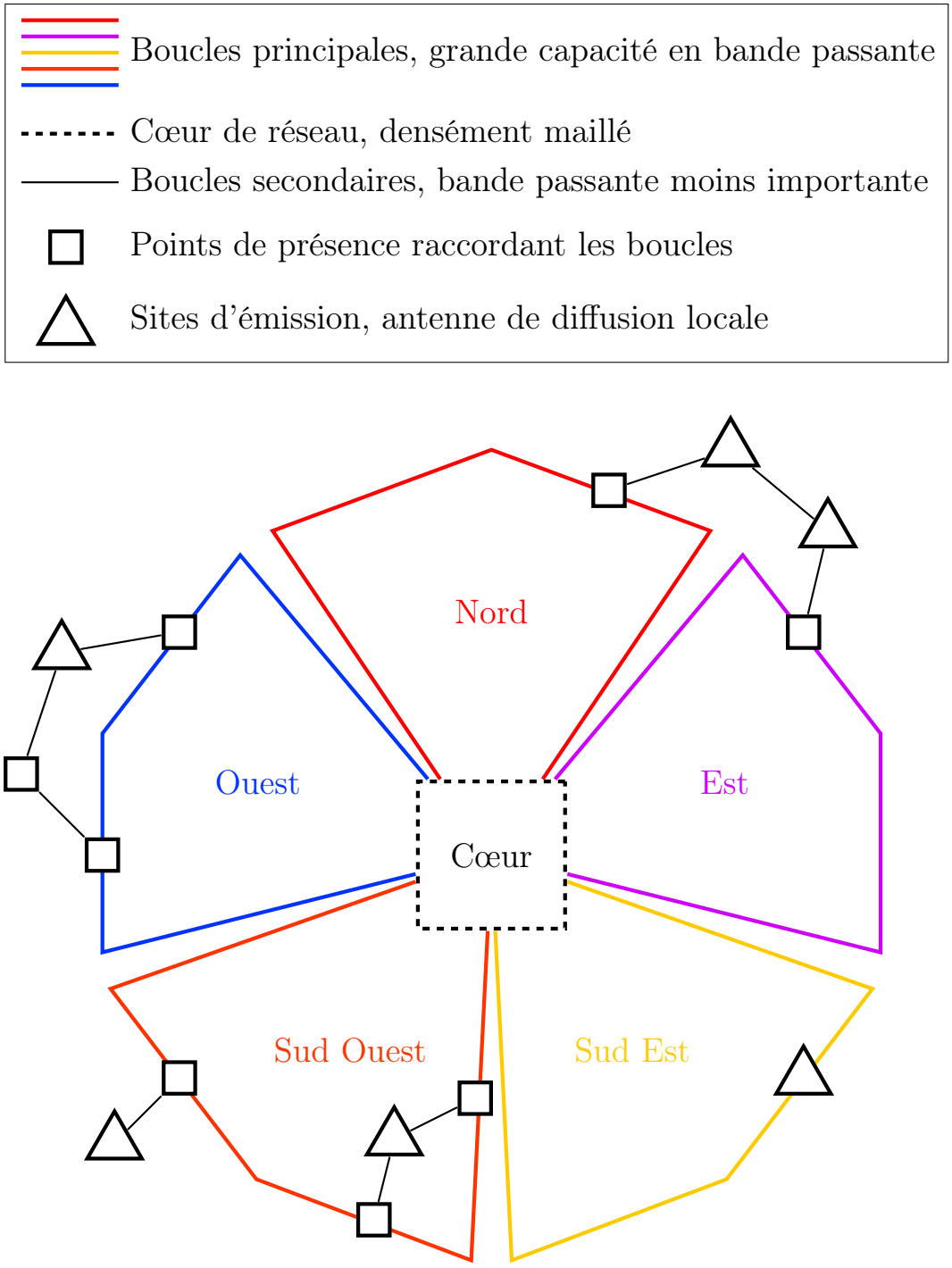


FIGURE 1.5 – Hiérarchie du réseau TMS

1.5 L'ingénierie protocolaire du réseau TMS

L'ingénierie déployée sur le réseau TMS est une variante du standard MVPN [4] et est fondée sur une combinaison des protocoles PIM [8] et IGMP [9].

Le protocole PIM est déployé sur le réseau de transport et se base sur les informations de routage existantes pour définir des chemins de transport du flux d'une source vers de multiples destinations. L'ensemble de ces chemins forment l'arbre de multidiffusion.

Le protocole IGMP est déployé sur le réseau d'accès, c'est-à-dire sur la partie du réseau opérateur où sont connectés les équipements clients. Ce protocole permet la gestion des abonnements à un flux de multidiffusion.

La figure 1.6 illustre le déploiement de ces protocoles : un équipement client émet une demande d'abonnement à travers un commutateur via le protocole IGMP. Le routeur de rattachement au réseau interprète cette demande, et déclenche le protocole PIM. Le protocole PIM établit alors une route vers cette nouvelle destination à l'aide des informations disponibles sur les routeurs. Ainsi, au fur et à mesure des requêtes d'abonnement, un arbre de multidiffusion est établi. En cas de panne réseau, le protocole PIM assure la redirection des flux suivant le nouveau plan de routage établi par les routeurs.

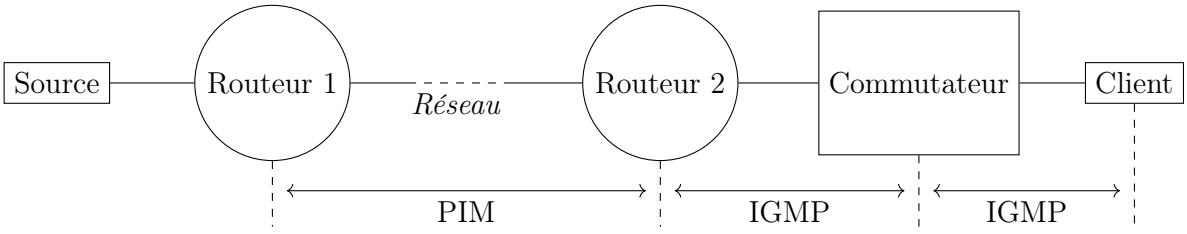


FIGURE 1.6 – Principe d'architecture MVPN exploitant PIM et IGMP

Ce principe d'architecture présente néanmoins des limitations :
- l'absence de vision globale du protocole PIM implique un déploiement non-optimal de l'arbre de multidiffusion en termes d'utilisation de bande passante ;
- la construction de l'arbre de multidiffusion par le protocole PIM est effectuée indépendamment pour chaque destination, ce qui appuie la non-optimalité du tracé en terme de bande passante ;
- il n'existe pas de mécanisme de gestion de bande passante dans le protocole PIM utilisé sur le réseau TMS, ce qui pourrait causer des problèmes de surprovisionnement. Néanmmoins, sur le réseau TMS, le protocole est limité, à travers un service virtuel, à l'exploitation de liens pour lesquels la capacité nécessaire est assurée ;
- en cas de panne sur le réseau, des pertes de contenu peuvent apparaître à cause du temps de convergence de PIM, lui-même dépendant du temps de convergence des protocoles de routage.

Ainsi, il existe encore une marge d'amélioration concernant la gestion de bande passante, mais aussi en continuité de service.

1.6 Conclusion

Dans ce chapitre, les flux audiovisuels et leur nature ont été définis, ainsi que le vocabulaire nécessaire à la compréhension du transport de flux en multidiffusion. Dans ces travaux de thèse, seuls les flux de transport et le réseau associé sont considérés. Les contraintes de transport d'un flux temps réel en multidiffusion impliquent une forte disponibilité du réseau. L'architecture de réseau étudiée dans cette thèse a été présentée :

- une couche physique hétérogène en performances et en technologies, mais cohérente avec le besoin de couverture géographique ;
- un déploiement hiérarchique depuis un cœur de réseau vers des boucles régionales ;
- une ingénierie protocolaire de multidiffusion exploitant les protocoles PIM et IGMP.

Le transport de flux audiovisuels sur un réseau existant est soumis à de fortes contraintes de QoE. L'ingénierie en place sur le réseau étudié permet de respecter ces contraintes, mais des limitations existent, et une meilleure utilisation du réseau est possible. Il est donc nécessaire de définir une nouvelle approche. Les contraintes spécifiques des flux transportés et de la multidiffusion sont présentées formellement dans le chapitre 2. Une approche dite "Seamless Multicast" est proposée pour répondre encore mieux aux attendus en QoS.

Chapitre 2

Seamless Multicast

Sommaire

2.1 Approche d'architecture 20
2.1.1 Techniques Seamless 20
2.1.2 Arbres redondants et résilients 20
2.1.3 Contraintes 21
2.1.4 Concept proposé 23
2.2 Redondance : définition formelle d'une paire d'arbre 24
2.2.1 Vocabulaire de théorie des graphes 24
2.2.2 Le problème DMDT 25
2.2.3 $\mathcal{NP}$-difficulté 29
2.3 Résilience : mécanisme de déploiement dynamique 30
2.3.1 État de l'art 30
2.3.2 Description des états du système 31
2.3.3 Mécanismes de résilience envisagés 32
2.3.4 Mécanisme dynamique proposé 34
2.3.5 Temps de réaction et phénomènes de bagotage 35
2.4 Conclusion 36

Dans ces travaux de thèse, l'objectif est d'améliorer la disponibilité d'un service de transport en multidiffusion d'un flux audiovisuel temps réel, notamment en supprimant les impacts possibles lors des phases de convergence du réseau. Des mécanismes de redondance et de résilience sont abordés dans la littérature. Ce chapitre présente une approche combinant redondance et résilience qui respecte les contraintes spécifiques du réseau.

2.1 Approche d'architecture

2.1.1 Techniques Seamless

Afin d'assurer la continuité de service du transport d'un flux, il est possible d'acheminer celui-ci simultanément par deux chemins redondants dans un réseau. Cela signifie que le flux est transporté en double, et que l'équipement récepteur est capable : de sélectionner l'un des flux, de basculer en cas de coupure de ce flux et de recombiner les deux flux en cas de pertes de paquets sur l'un des flux ou sur les deux flux. On parlera de techniques de réception "Seamless" d'un flux. Dans la littérature, le terme de transport "Hitless" existe également. La figure 2.1 présente une vue globale du principe de réception Seamless.

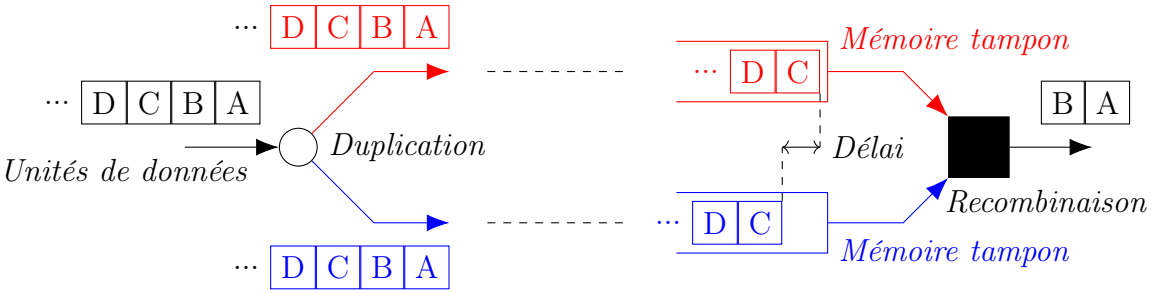


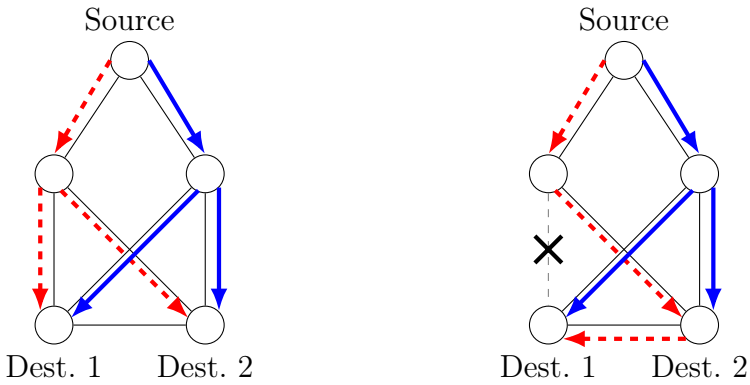
FIGURE 2.1 – Illustration du principe de réception "Seamless"

Des technologies permettent ces fonctions, telles que [13] ou encore [14] pour les réseaux ATM, mais ne seront pas détaillées dans le cadre de ces travaux. En effet, les problématiques et techniques associées sont hors du cadre de ces travaux, et on fera abstraction de l'implémentation de ces techniques : si un équipement reçoit chaque paquet de données sur au moins l'un des deux flux, on considèrera qu'il est capable de restituer un flux complet.

2.1.2 Arbres redondants et résilients

Exploitant ce principe de transport redondant, l'architecture suivante est proposée. Deux arbres de multidiffusion redondants sont établis, transportant le même flux simultanément d'une même source au même ensemble de destinations. Ces arbres doivent être indépendants par destination. En d'autres termes, pour chaque destination, les chemins contenus dans chaque arbre ne partagent aucun nœud ou lien commun. L'objectif est

d'éviter les points uniques de panne, ou SPOF - "Single Point Of Failure". Ainsi, en cas de panne sur le réseau, si l'un des chemins composant un arbre est hors-service, l'autre arbre contient nécessairement un autre chemin vers la même destination. Cette destination ne subit alors aucun impact grâce aux techniques Seamless évoquées plus haut. À titre d'exemple, la figure 2.2a illustre une paire d'arbres indépendants.



(a) Arbres indépendants (b) Sécurisation dest. 1 après panne

FIGURE 2.2 – Illustration du concept de transport redondant et résilient

L'architecture proposée comporte également un mécanisme de résilience. En effet, suite à une panne, il est possible qu'une destination ne soit plus alimentée que par un seul chemin via un seul arbre. On parle alors de destination "non-sécurisée". Si une seconde panne se produit et que le seul chemin existant au sein d'un arbre est hors-service, alors la destination subira un impact lors de la restitution des images ou du son. Grâce à un mécanisme de résilience on cherche à rétablir la sécurisation. Après une première panne ayant causé la mise hors-service d'un chemin vers une destination, une nouvelle paire de chemin est mise en place, toujours au sein d'un arbre de multidiffusion. Ainsi, le transport pourra supporter une seconde panne, sans que la destination ne subisse un impact. Un exemple de sécurisation après une panne est présenté en figure 2.2b : le lien entre la destination 2 et la destination 1 est utilisé après une panne pour permettre l'acheminement du flux.

2.1.3 Contraintes

À cette approche combinant redondance et résilience s'ajoutent, trois contraintes.

Indépendance maximale

La première contrainte est liée à la notion d'indépendance par destination des arbres. En effet, suivant la topologie, une indépendance totale n'est pas toujours possible, par exemple suite à une panne ou dans le cas d'une partie pendulaire du réseau. On appelle pendulaire une partie du réseau qui n'est reliée que par un seul lien au reste du réseau.

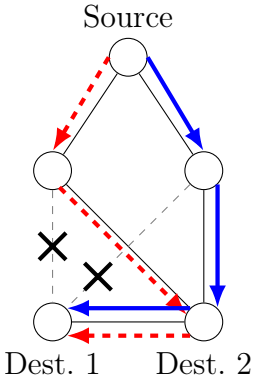


FIGURE 2.3 – Illustration du concept d'indépendance maximale

Il est néanmoins possible d'établir des chemins partageant un minimum d'éléments du réseau. On parle alors d'indépendance maximale. Une indépendance totale impose une stricte différence entre les arbres, alors que l'indépendance maximale correspond à un partage minimal d'éléments. La figure 2.3 présente un cas de panne pour lequel les deux arbres doivent partager un lien pour sécuriser les destinations, puisque les deux autres liens vers la destination 2 sont en panne.

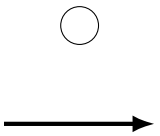
Contrainte de distance et de délai

La deuxième contrainte est due à la fois à la sûreté de fonctionnement et à la problématique de diffusion temps réel. Il s'agit d'une contrainte sur la longueur des chemins empruntés pour le transport du flux. D'une part, chaque nœud et lien traversé dégrade la disponibilité d'un chemin. En effet, chaque élément possède un taux de disponibilité, duquel peut définir un temps théorique cumulé durant lequel il est fonctionnel sur une période donnée. La disponibilité d'un chemin diminue si le nombre de composants augmente. En particulier, la disponibilité d'un système dont les composants sont en série est le produit de leur disponibilité :

$$D_{Système} = \prod_{Composants} D_{Composant} \tag{2.1}$$

La figure 2.4 illustre deux exemples simples de calcul de disponibilité. On peut noter que la différence de disponibilité annuelle d'un chemin à trois sauts par rapport à un chemin à deux sauts est de plus d'une heure. Étant donné les contraintes associées à la diffusion audiovisuelle, l'objectif est de réduire au maximum l'indisponibilité.

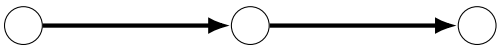
D'autre part, si la latence totale du transport est trop élevée, l'objectif de diffusion temps réel n'est pas rempli. Chaque élément du réseau ajoute un délai à la latence globale. Pour un nœud du réseau, il s'agit du temps de traitement d'un paquet de données (commutation, routage, analyse...) et pour une liaison, du temps de transmission. Ce temps peut dépendre du protocole, de la technologie physique ou encore de la qualité de support de transmission.



$D_{noeud} = 0.99999$ soit une indisponibilité de :
$60 * 24 * 365 * 0.99999 = 5.256$ min/an

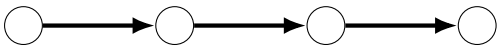
$D_{lien} = 0.9999$ soit une indisponibilité de :
$60 * 24 * 365 * 0.9999 = 52.56$ min/an

(a) Valeurs de disponibilité utilisées pour les exemples



$D = 0.99999^3 * 0.9999^2 = 0.99977$
Soit une indisponibilité de 120.888 min/an

(b) Chemin à deux sauts



$D = 0.99999^4 * 0.9999^3 = 0.99966$
Soit une indisponibilité de 178.704 min/an

(c) Chemin à trois sauts

FIGURE 2.4 – Exemples de calcul de disponibilité d'un chemin

Ainsi, la contrainte de latence et de sûreté de fonctionnement sont corrélées. Dans les deux cas il s'agit de réduire le nombre d'éléments traversés, ou de préférer traverser un ensemble d'éléments pour lequel la latence totale est faible. Algorithmiquement, il s'agit uniquement de modifier la pondération d'un graphe suivant l'usage. Dans la suite de ces travaux, on utilisera le terme "contrainte en délai" indistinctement pour simplifier la lecture.

Usage de bande passante optimisé

La troisième contrainte concerne l'utilisation des ressources du réseau. Augmenter le nombre de chemins disjoints améliore la robustesse du transport, mais implique également d'augmenter la consommation en bande passante. C'est notamment pour cela que ces travaux se limitent à deux chemins disjoints par destination, c'est-à-dire à deux arbres de multidiffusion.

Ainsi, la bande passante totale utilisée par les arbres doit être minimisée. De plus, il est possible que certaines liaisons du réseau soient utilisées par des services différents. En effet, d'autres transports de flux audiovisuels existent sur la même infrastructure, certains nationaux, d'autres régionaux. Ainsi, l'utilisation des liens peut varier au sein de la topologie. Il faut donc éliminer dès la phase de calcul les liens de la topologie de transport dont la bande passante disponible ne peut pas supporter deux fois le débit du flux transporté.

2.1.4 Concept proposé

L'architecture Seamless Multicast consiste à exploiter une paire résiliente d'arbres redondants pour transporter un flux audiovisuel temps réel sur un réseau. Les arbres doivent être maximalement indépendants et contraints en délai, et la paire la moins coûteuse possible.

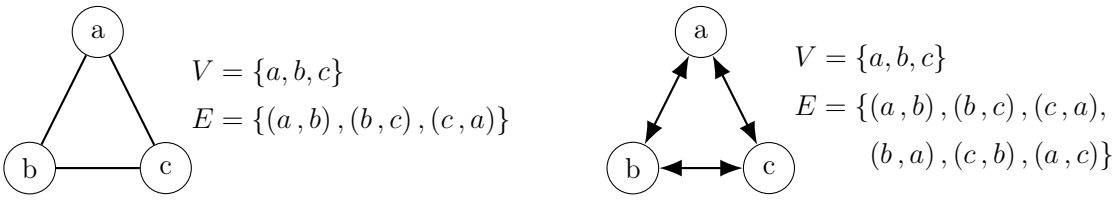
2.2 Redondance : définition formelle d'une paire d'arbre

Afin de calculer des chemins de transport, la topologie réseau est modélisée comme un graphe, sur lequel on peut appliquer des méthodes de calcul bien connues dans la théorie des graphes. On peut noter qu'il existe des mécanismes de création de chemins par ingénierie protocolaire, ne nécessitant pas une vision globale du problème, comme par exemple le standard MVPN. Dans cette formulation, chaque lien du réseau est transposé dans le graphe comme deux liens unidirectionnels. En particulier dans le contexte industriel de ces travaux, les flux sont en général unidirectionnels, et la bande passante consommée dans un seul sens. Cette modélisation est donc nécessaire et est compatible avec des réseaux dont les liens sont exploités de manière symétrique.

2.2.1 Vocabulaire de théorie des graphes

Un graphe est défini comme un ensemble de liens et de nœuds. On utilise usuellement V pour l'ensemble des sommets et E pour les liens. Autrement dit, un graphe est un couple d'ensemble $G = (E, V)$ avec E constitué de paires d'éléments de V, c'est-à-dire $E \in V^2$.

Si les éléments de E sont des paires ordonnées ou couples, le graphe est dit orienté. On l'appelle également digraphe. Ces notions sont illustrées dans la figure 2.5.



(a) Graphe (b) Digraphe

FIGURE 2.5 – Illustrations des notions de graphe non-orienté et orienté

Un graphe peut être pondéré par un ensemble C contenant les poids associés à chaque lien de E. On a alors $|C| = |E|$. Un graphe peut être associé à de multiples pondérations.

Un chemin P est un sous-ensemble de E ordonné de liens distincts reliant un ensemble de sommets distincts sauf le premier et le dernier (excepté pour un chemin d'un seul lien).

Un cycle C est un sous-ensemble de E ordonné de liens distincts reliant un ensemble de sommets distincts y compris le premier et le dernier. Un graphe est dit acyclique s'il ne contient aucun cycle.

Un graphe est dit connecté si il existe un chemin pour toute paire de sommets donnée. Si au moins une paire de sommet ne peut pas être liée par un chemin, le graphe est dit déconnecté.

Un arbre A est un graphe connecté et acyclique. Le nœud entrant d'un lien est appelé nœud père, tandis que le nœud sortant est qualifié de nœud fils. Si l'un des nœuds de l'arbre est défini comme racine, on parle alors d'arbre enraciné. Il s'agit généralement d'un nœud n'ayant pas de père. Les nœuds d'un arbre n'ayant pas de fils sont appelés des feuilles.

2.2.2 Le problème DMDT

Dans un graphe $G_0 = (V, E)$ pondéré, connecté et orienté, l'objectif est de calculer deux arbres de multidiffusion ayant les mêmes nœuds terminaux, les plus indépendants possibles par destination, les moins coûteux possible. De plus, on souhaite que chaque chemin source-destination soit sous un certain seuil de longueur (en nombre de sauts, en latence...). On nommera ce problème DMDT : Delay-constrained Minimal Destination maximally-disjoint Trees.

Formulation non-linéaire

Dans un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai, on cherche, pour une source s, un ensemble de destinations T et une contrainte en délai Δ, deux arbres A_R et A_B les plus disjoints possible en liens, les plus légers possible sous contrainte de délai Δ connectant s à T.

Soient R et B deux vecteurs binaires représentants les deux arbres recherchés A_R et A_B. Chaque élément d'un vecteur représente l'utilisation d'un lien $(x, y) \in E$. Leur valeur est notée $R(x, y)$ et $B(x, y)$. Elle vaut 1 si le lien appartient à l'arbre correspondant, 0 sinon. Soit Z un vecteur dont les valeurs représentent chacune l'utilisation d'un lien $(x, y) \in E$ par les deux arbres A_R et A_B pour joindre une même destination. Cette valeur est nulle si le lien n'est pas commun aux deux arbres, sinon elle correspond au nombre de destinations jointes par les deux arbres en passant par ce lien. Autrement dit :

$$\forall (x, y) \in E,\ R(x, y) = \begin{cases} 0 & \text{si } (x, y) \in A_R \\ 1 & \text{si } (x, y) \notin A_R \end{cases} \tag{2.2}$$

$$\forall (x, y) \in E,\ B(x, y) = \begin{cases} 0 & \text{si } (x, y) \in A_B \\ 1 & \text{si } (x, y) \notin A_B \end{cases} \tag{2.3}$$

$$\forall (x, y) \in E,\ Z(x, y) = \begin{cases} 0 & \text{si } (x, y) \notin A_R \cap A_B \\ z & \text{sinon, avec } z = \text{ nombre de destinations jointes} \end{cases} \tag{2.4}$$

Une formulation de ce problème d'optimisation non-linéaire est la suivante :
Minimiser :

$$\begin{aligned}\mathcal{F}(R,B) = &\sum_{(x,y)\in E} C_0(x,y)R(x,y) \\ &+ \sum_{(x,y)\in E} C_0(x,y)B(x,y) \\ &+ 2\sum_{(x,y)\in E} C_0(x,y) \sum_{(x,y)\in E} Z(x,y)\end{aligned} \tag{2.5}$$

Sachant que :

$$R(x,y) \in \{0,1\} \qquad \forall (x,y) \in E \tag{2.6}$$

$$B(x,y) \in \{0,1\} \qquad \forall (x,y) \in E \tag{2.7}$$

$$Z(x,y) = \sum_{t\in T} R_t(x,y) * B_t(x,y) \qquad \forall (x,y) \in E \tag{2.8}$$

$$\sum_{E} D(x,y) * R_t(x,y) \leq \Delta \qquad \forall t \in T \tag{2.9}$$

$$\sum_{In(v)} R_t(x,y) - \sum_{Out(v)} R_t(x,y) = 0 \qquad \forall t \in T \text{ et } v \in V \setminus \{s,t\} \tag{2.10}$$

$$\sum_{In(t)} R_t(x,y) = 1 \qquad \forall t \in T \tag{2.11}$$

$$\sum_{Out(s)} R_t(x,y) = 1 \qquad \forall t \in T \tag{2.12}$$

$$0 \leq R_t(x,y) \leq R(x,y) \qquad \forall (x,y) \in E \text{ et } t \in T \tag{2.13}$$

$$\sum_{E} D(x,y) * B_t(x,y) \leq \Delta \qquad \forall t \in T \tag{2.14}$$

$$\sum_{In(v)} B_t(x,y) - \sum_{Out(v)} B_t(x,y) = 0 \qquad \forall t \in T \text{ et } v \in V \setminus \{s,t\} \tag{2.15}$$

$$\sum_{In(t)} B_t(x,y) = 1 \qquad \forall t \in T \tag{2.16}$$

$$\sum_{Out(s)} B_t(x,y) = 1 \qquad \forall t \in T \tag{2.17}$$

$$0 \leq B_t(x,y) \leq B(x,y) \qquad \forall (x,y) \in E \text{ et } t \in T \tag{2.18}$$

Avec :

- R_t vecteur représentant le chemin de s vers t au sein de l'arbre A_R
- B_t vecteur représentant le chemin de s vers t au sein de l'arbre A_B
- $In(v) = \{(a,b) \in E \mid a = v\}$ liens entrant d'un noeud $v \in V$;
- $Out(v) = \{(a,b) \in E \mid b = v\}$ liens sortant d'un noeud $v \in V$.

La fonction de coût se compose de trois termes. Les deux premiers termes correspondent au poids de chacun des arbres, que l'on veut le plus bas possible. Le troisième terme permet d'évaluer l'indépendance des arbres, à l'aide du vecteur Z défini par la contrainte (2.8), pondéré par deux fois le poids total du graphe.

La pondération par deux fois le poids total du graphe permet de donner une plus grande importance à l'indépendance des arbres. Quels que soient les arbres trouvés, le nombre de liens communs par destination pénalisera plus fortement la fonction d'évaluation. La contrainte d'optimisation de bande passante (i.e. du poids de chaque arbre) est secondaire par rapport à l'indépendance des arbres.

$$\beta = \sum_{(x,y)\in E} Z(x,y) \tag{2.19}$$

Pour toute paire d'arbre partageant β (2.19) liens relativement à chaque destination, la somme du poids des arbres et de β multipliée par deux fois le poids du graphe est plus petite que toute autre paire d'arbres qui serait moins indépendante, c'est-à-dire pour laquelle la valeur de β est plus élevée. Autrement dit, la fonction de coût doit être croissante en β indépendamment du poids des arbres :

Theorème 1.

$\forall A_1, A_2 \subset E$ *et la valeur associée* $\beta_{12} \geq 0$,
$\forall A_3, A_4 \subset E$ *et la valeur associée* $\beta_{34} \geq 0$,
Si $\beta_{12} < \beta_{34}$ *alors :*

$$C_0(A_1) + C_0(A_2) + \beta_{12} * 2 * C_0(E) \leq C_0(A_3) + C_0(A_4) + \beta_{34} * 2 * C_0(E) \tag{2.20}$$

Démonstration.

Pour un graphe $G_0 = (V, E)$ pondéré par $C_0 \geq 0$,
$\forall A \subset E, A \neq \emptyset$,

$$0 < C_0(A) \leq C_0(E)$$
$$\Rightarrow 0 < C_0(A_i) + C_0(A_j) \leq 2C_0(E),\ \forall A_i, A_j \subset (E) \tag{2.21}$$

$\forall A_1, A_2 \subset E$ et la valeur associée $\beta_{12} \in \mathbb{N}^+, A_1 \neq \emptyset, A_2 \neq \emptyset$,
$\forall A_3, A_4 \subset E$ et la valeur associée $\beta_{34} \in \mathbb{N}^+, A_3 \neq \emptyset, A_4 \neq \emptyset$,
$0 \leq \beta_{12} < \beta_{34}$
$\Rightarrow \beta_{12} + 1 \leq \beta_{34}$ car $\beta_{12}, \beta_{34} \in \mathbb{N}^+$
$\Rightarrow \beta_{12} * 2C_0(E) + 2 * C_0(E) \leq \beta_{34} * 2C_0(E)$ car $C_0(E) > 0$
$\Rightarrow \beta_{12} * 2C_0(E) + C_0(A_1) + C_0(A_2) \leq \beta_{34} * 2C_0(E)$ d'après (2.21)
$\Rightarrow \beta_{12} * 2C_0(E) + C_0(A_1) + C_0(A_2) < \beta_{34} * 2C_0(E) + C_0(A_3) + C_0(A_4)$
car $C_0(A_3) > 0$ et $C_0(A_4) > 0$

□

Pour chaque destination t, les vecteurs B_t et R_t représentent respectivement les liens appartenant au chemin vers t au sein des arbres A_r et A_b. Ainsi, R et B doivent en suivre les valeurs, comme énoncé par les contraintes g_R et g_B. A noter qu'il ne s'agit pas d'une contrainte d'égalité, car si un lien est emprunté pour joindre une destination, sa valeur dans les vecteurs représentant les arbres est de 1, même s'il existe des destinations pour lesquelles il n'est pas utilisé.

La contrainte en délai Δ est exprimée par les contraintes (2.9) et (2.14). Les contraintes (2.10, 2.11, 2.12) et (2.15, 2.16, 2.17) définissent la continuité de l'arbre, de la source s jusqu'aux destinations T.

Cette formulation ne garantit pas l'absence de cycles dans les arbres A_R et A_B, ni de multiples chemins vers une destination au sein d'un même arbre. Néanmoins, la minimisation de la fonction de coût élimine ce problème. En effet, si un cycle existe dans un arbre, alors il existe nécessairement une solution similaire sans ce cycle, et elle sera nécessairement moins coûteuse. Ainsi la solution optimale du problème ne contient pas de cycle. Le raisonnement est similaire pour le cas des chemins multiples.

Variante à indépendance de nœud

On peut représenter l'indépendance de nœud en exploitant les mêmes vecteurs de représentation que dans la première formulation. Un nœud y appartient à un arbre si et seulement si il existe un lien dont le nœud sortant est y, ou si y est la source s des arbres :

$$y \in A_R \Leftrightarrow \exists\, x \in V \text{ tel que } R(x,y) = 1 \text{ ou } y = s \tag{2.22}$$

$$y \in A_B \Leftrightarrow \exists\, x \in V \text{ tel que } B(x,y) = 1 \text{ ou } y = s \tag{2.23}$$

De même, un nœud y appartient au chemin de s vers une destination t au sein d'un arbre A_R si et seulement si :

$$\exists\, x \in V \text{ tel que } R_t(x,y) = 1 \text{ ou } y = s \tag{2.24}$$

Un nœud y appartient aux deux chemins de s vers une destination t au sein des deux arbres A_R et A_B si et seulement si :

$$\exists\, x_R, x_B \in V^2 \text{ tel que } R_t(x_R,y) * B_t(x_B,y) = 1 \text{ ou } y = s \tag{2.25}$$

Afin d'adapter la formulation du problème à l'indépendance de nœud, il suffit de remplacer la contrainte (2.8) par la contrainte (2.26) suivante :

$$Z(x,y) = \sum_{t \in T} \left(\sum_{x_R, x_B \in V^2} R_t(x_R,y) * B_t(x_B,y) \right) \qquad \forall y \in V \tag{2.26}$$

Cette nouvelle contrainte (2.26) ignore le cas ou $y = s$: les arbres ont nécessairement la même source d'après la définition du problème. Aussi, il est trivial qu'ils partagent ce nœud pour tous les chemins. Il en va de même pour les nœuds destinations.

2.2.3 $\mathcal{NP}$-difficulté

On appelle problème de décision un problème dont la réponse est "Oui" ou "Non". Un problème de décision est dit de classe $\mathcal{P}$ s'il existe au moins un algorithme permettant de résoudre ce problème en un temps polynomial. Il est dit de classe $\mathcal{NP}$ si le problème est soluble en un temps polynomial par une machine de Turing non-déterministe, c'est-à-dire qui peut fonctionner en parallèle sans communication entre les files de calculs. Par définition, tous les problèmes de classe $\mathcal{P}$ sont également de classe $\mathcal{NP}$. La relation inverse est indéterminée, et se rapporte au célèbre problème $\mathcal{P} = \mathcal{NP}$. Un problème de décision est dit $\mathcal{NP}$-complet s'il existe une réduction polynomiale de tout problème de classe $\mathcal{NP}$ qui amène à ce problème. Autrement dit, si l'on peut transformer en un temps polynomial tout problème de classe $\mathcal{NP}$ en un problème donné, alors celui-ci est $\mathcal{NP}$-complet.

Concernant les problèmes d'optimisation, la notion de $\mathcal{NP}$-difficulté permet en pratique de définir si un problème est "difficile", c'est à dire s'il existe un algorithme en temps polynomial permettant de le résoudre.

En pratique, cela signifie qu'il n'est pas possible de garantir à la fois de trouver une solution optimale et de la trouver rapidement. Autrement dit, soit une solution est obtenue rapidement avec l'espoir qu'elle soit satisfaisante, soit la solution optimale est obtenue en un temps plus long, qu'on espère suffisamment court pour l'application souhaitée.

Theorème 2.
Le problème DMDT est $\mathcal{NP}$-difficile.

Démonstration.
Le problème du calcul de Chemins Contraints et Maximalement Indépendants (CCMI) est $\mathcal{NP}$-difficile comme prouvé par [15].

Le problème CCMI est un cas particulier du problème DMDT. Ce dernier se définit de la manière suivante : trouver une paire d'arbre d'une source vers $|T|$ destinations dont les chemins par destinations sont les plus disjoints possibles, les moins coûteux possible et contraints en délai. Pour $|T| = 1$, le problème DMDT peut alors être traité comme une instance du problème CCMI.

Ainsi, par généralisation du problème CCMI qui est $\mathcal{NP}$-difficile, on prouve que le problème DMDT est lui-même $\mathcal{NP}$-difficile.

□

La $\mathcal{NP}$-difficulté du problème DMDT impose une orientation des travaux vers une solution par approximation. En effet, en pratique, garantir rapidement l'obtention de la solution optimale d'un problème $\mathcal{NP}$-difficile est impossible. Néanmoins, l'existence d'une telle solution est garantie. Ainsi l'approche des travaux de cette thèse se concentre sur deux heuristiques ainsi que sur un algorithme génétique. L'objectif est l'obtention rapide d'une solution satisfaisante au problème. L'algorithmique associée à la résolution du problème DMDT fait l'objet du chapitre 3 de cette thèse.

2.3 Résilience : mécanisme de déploiement dynamique

L'architecture Seamless Multicast propose d'utiliser une paire d'arbres de diffusion redondants pour assurer la continuité de service en cas de panne simple. Un mécanisme de résilience est associé à cette paire d'arbres afin d'assurer la continuité de service en cas de panne. On cherche à définir un mécanisme tel que, en cas de panne sur un chemin, le mécanisme assure que ce chemin est rétabli en un temps raisonnable, sans impact sur le reste de la diffusion, afin que le système retourne dans un état sécurisé par redondance de transport du flux.

On appellera "destination impactée" toute destination pour laquelle au moins l'un des chemins est hors-service en cas de panne. On parlera de "non-sécurisée" pour désigner toute destination pour laquelle l'un des chemin est hors-service, et de "non-alimentée" en cas de pannes des deux voies.

Dans cette partie, on travaillera sous l'hypothèse que l'architecture est capable de calculer et déployer des arbres aux propriétés énoncées en 2.2.

2.3.1 État de l'art

Les technologies de diffusion Multicast IP exploitent divers mécanismes de résilience. Les protocoles PIM [8] ou MBGP [10] comprennent notamment des mécanismes de reroutage. Certaines architectures exploitent également des résiliences pro-actives, par exemple en déployant des routes de secours à l'avance [16]. Pour aller plus loin, [17] se base sur le protocole PIM, en exploitant une redondance de source et d'arbres dynamiques comme solution de secours.

L'un des défauts de la résilience en multidiffusion sur IP est qu'en cas de panne, le rétablissement du service ne peut se faire qu'après un temps plus long qu'un simple calcul de chemin. En effet, l'information qu'une destination cherche à se reconnecter à une source doit remonter aux éléments de contrôle du réseau. Par exemple, dans le protocole PIM [8], les routeurs intermédiaires doivent attendre de nouvelles requêtes d'abonnement au groupe de multidiffusion des destinations impactées avant d'établir à nouveau la connexion.

Pour pallier ce temps supplémentaire, certaines architectures exploitent des résiliences pro-actives, par exemple en déployant des routes de secours à l'avance. On peut citer les technologie Fast Reroute sur des réseaux IP/MPLS [18], ou des procédés indépendants du protocole déployé dans [19].

D'autres approches visent à réduire le temps de rétablissement en changeant l'échelle du calcul, par exemple dans [20] où l'arbre de multidiffusion est divisé en sous-arbres qui disposent chacun d'un mécanisme de résilience.

Les architectures de diffusion existantes fondées sur du Software-Defined Networking et l'état de l'art associé seront abordées au chapitre 4 de cette thèse.

Ces travaux partent du principe qu'il est possible de déployer un arbre de multidiffusion après l'avoir calculé (par exemple par des tunnels MPLS, OpenFlow...). De même,

des routes alternatives peuvent-être appliquées. Afin de travailler sur un mécanisme de résilience approprié, il est nécessaire de définir les états du système étudié.

2.3.2 Description des états du système

Le système étudié vise à établir une diffusion sans perte vers de multiples destinations. Afin de décrire les états possibles du système, il faut d'abord définir les états d_i de fonctionnement de chacune des destinations. La figure 2.6 représente un diagramme d'état pour une destination donnée.

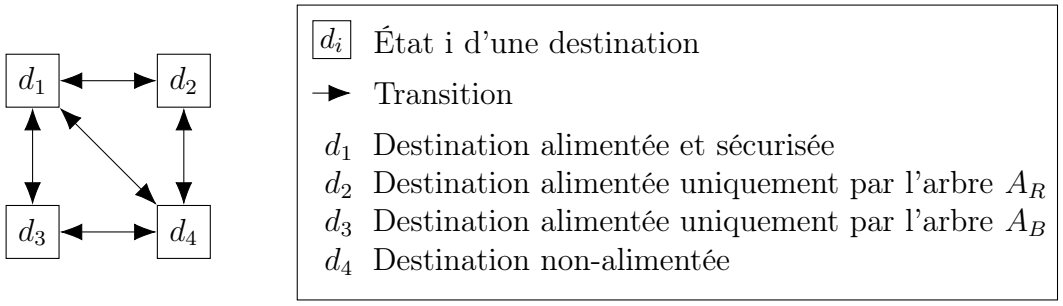


FIGURE 2.6 – États possibles pour une destination

Il est pertinent que le système soit considéré "par destination", car c'est la réception d'un flux par les destinations qui définit la QoE et la QoS du système. Cependant, l'état des arbres donne une information synthétique sur le système. La figure 2.7 présente les états de fonctionnement des deux arbres séparément.

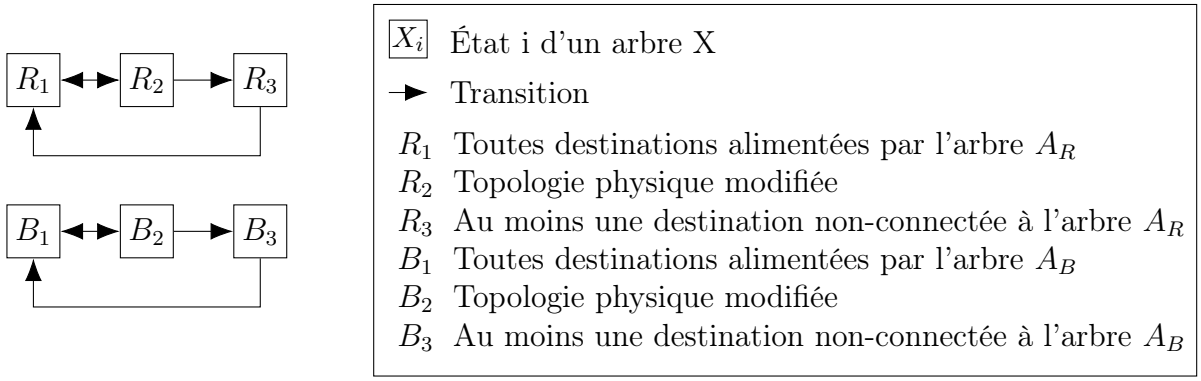


FIGURE 2.7 – États possibles des arbres composants du système

On peut enfin définir des états génériques pour le système, qui serviront à définir les mécanismes de résilience. La figure 2.8 présente les états D_i de fonctionnement du système.

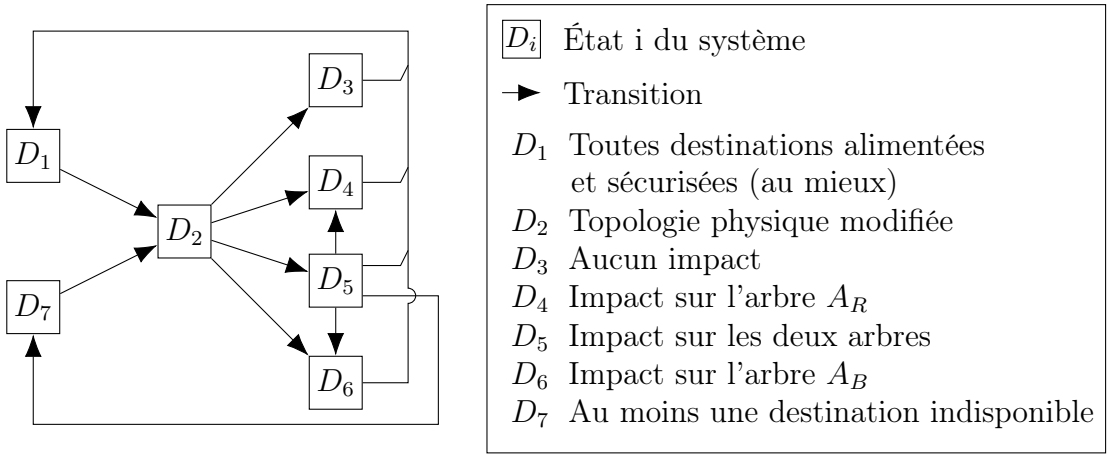


FIGURE 2.8 – États possibles du système

Il est important de noter que les états D_i sont conditionnés par les états R_i et B_i, eux-même conditionnés par les états d_i de chaque destination. Par exemple, $D_1 = R_1 \, et \, B_1$ ou encore $R_1 = \prod(d_1 \, ou \, d_2)$.

L'état D_2 est l'état dans lequel la topologie physique a été modifiée, quelle que soit l'évolution de la topologie. Il peut s'agir d'une panne, ce qui amène aux états D_4, D_5 ou D_6, ou d'un ajout de nouveaux éléments au réseau. Dans ce cas, les arbres ne sont pas impactés par le changement, et le système rentre alors dans l'état D_3.

2.3.3 Mécanismes de résilience envisagés

L'état de l'art exploite deux arbres redondants, ou assure la résilience d'un arbre. Notre problématique est d'assurer la résilience d'une paire d'arbre, en conservant l'indépendance maximale. Les approches de l'état de l'art sont donc insuffisantes, mais donnent des bases sur lesquelles s'appuyer.

Après rétablissement, l'objectif est de revenir à la situation initiale. Sans quoi, au fur et à mesure des pannes du réseau, la topologie logique va se dégrader. Il faut donc considérer la problématique de rétablir les arbres précédemment établis sur la topologie physique initiale, et ce sans impact sur la diffusion. La même problématique se pose dans le cas d'évolution de la topologie physique, par ajout de nœud, de lien, rebouclage ou maillage entre nœuds existants. Si une nouvelle paire d'arbre plus optimale est possible, il faut l'atteindre sans impacter la diffusion lors du déploiement. On cherche à définir un mécanisme de résilience déterministe.

De manière traditionnelle, on parle de routage pour une technique établissant un chemin entre deux points du réseau. On peut séparer les calculs de routage en deux

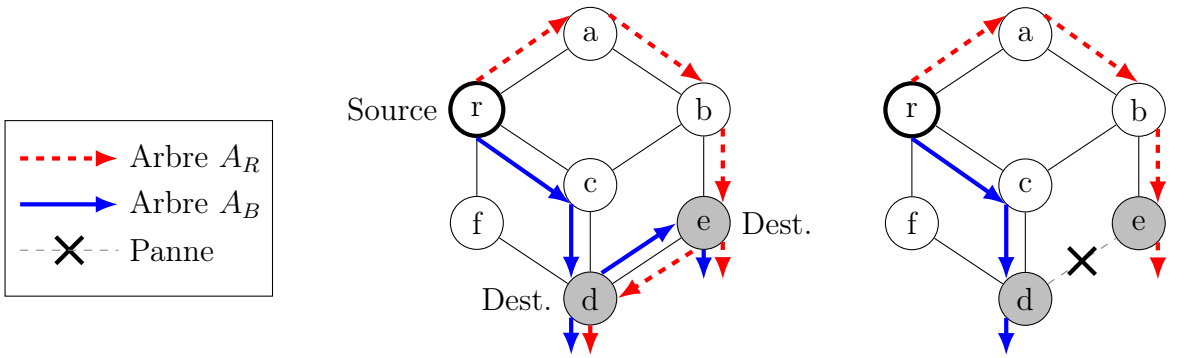
catégories : le routage de chemin et le routage de segment.

L'approche par routage de chemin consiste à calculer un chemin de la source vers la destination. Afin d'assurer la résilience d'un arbre, on trouve des solutions exploitant du routage de chemin pour chaque destination impactée. Dans le cadre de la diffusion par paire d'arbre, on pourrait envisager pour chaque destination impactée de recalculer un chemin, en pondérant l'arbre défaillant à une valeur nulle et le fonctionnel à une valeur maximale. Les liens communs sont également pondérés à une valeur nulle. Les liens défaillants et branches inutiles sont au préalable retirés de la topologie de calcul. Cette méthode ne permet pas en tant que telle d'assurer la résilience d'une paire d'arbre, notamment car elle ne considère pas le cas dans lequel les deux arbres sont impactés.

De la même manière que pour le routage de chemin, le routage de segment vise à rétablir la connexion entre deux points d'un chemin dont le ou les liens originaux sont en panne. Il faut noter que les deux solutions peuvent aboutir au même résultat. Des solutions basées sur le routage de segment existent dans la littérature, et il est possible d'envisager un mécanisme de résilience pour la diffusion par paire d'arbre, de manière similaire au routage de chemin. Mais un tel mécanisme possède le même défaut que précédemment, à savoir qu'il ne considère pas les cas où les deux arbres sont impactés.

La figure 2.9 présente une telle situation. Certains liens sont exploités dans des sens différents par les arbres, et une panne peut impacter les deux arbres pour des destinations différentes. Ainsi, si l'arbre A_R assure la continuité de service d'une destination en cas de panne, l'arbre A_B est peut-être en train de l'assurer pour une autre destination. Dans l'exemple de la figure 2.9, si le lien entre les deux destinations d et e tombe en panne, alors c'est l'arbre A_R qui diffuse pour e et l'arbre A_B qui diffuse pour d. Il n'est pas possible de redéployer un seul des deux arbres sans qu'une destination subisse un impact de déploiement. Dans ce genre de cas, il est important de rétablir la sécurisation, tout en maintenant au moins une diffusion en place pour assurer la continuité de service.

En conclusion, assurer la résilience de chacun des chemins ou de chacun des arbres est insuffisant. Par conséquent, les travaux présentés dans la suite de ce chapitre proposent un mécanisme de résilience qui concerne la paire comme une seule entité.



(a) Légende (b) Topologie (c) Cas de panne d-e

FIGURE 2.9 – Exemple de cas de pannes impactant les deux arbres

2.3.4 Mécanisme dynamique proposé

Une approche est de conserver un chemin et recalculer le second pour augmenter l'indépendance et donc la robustesse. Ainsi, la continuité de service est assurée. Mais rien ne garantit le retour à l'état initial, supposé optimal. Si on force un retour à l'état initial, on prend le risque d'interrompre le service dans certains cas.

On considère alors une nouvelle approche : au lieu d'étudier les cas de panne et de rétablissement, le système doit considérer les changements de topologie d'une manière générale (panne, ajout de liens...). La figure 2.10 propose un mécanisme dit de "dynamisation" de l'architecture, plutôt que de résilience. L'annexe A présente une définition formelle de ce mécanisme sous la forme d'un automate fini, ainsi qu'une vérification de ses propriétés.

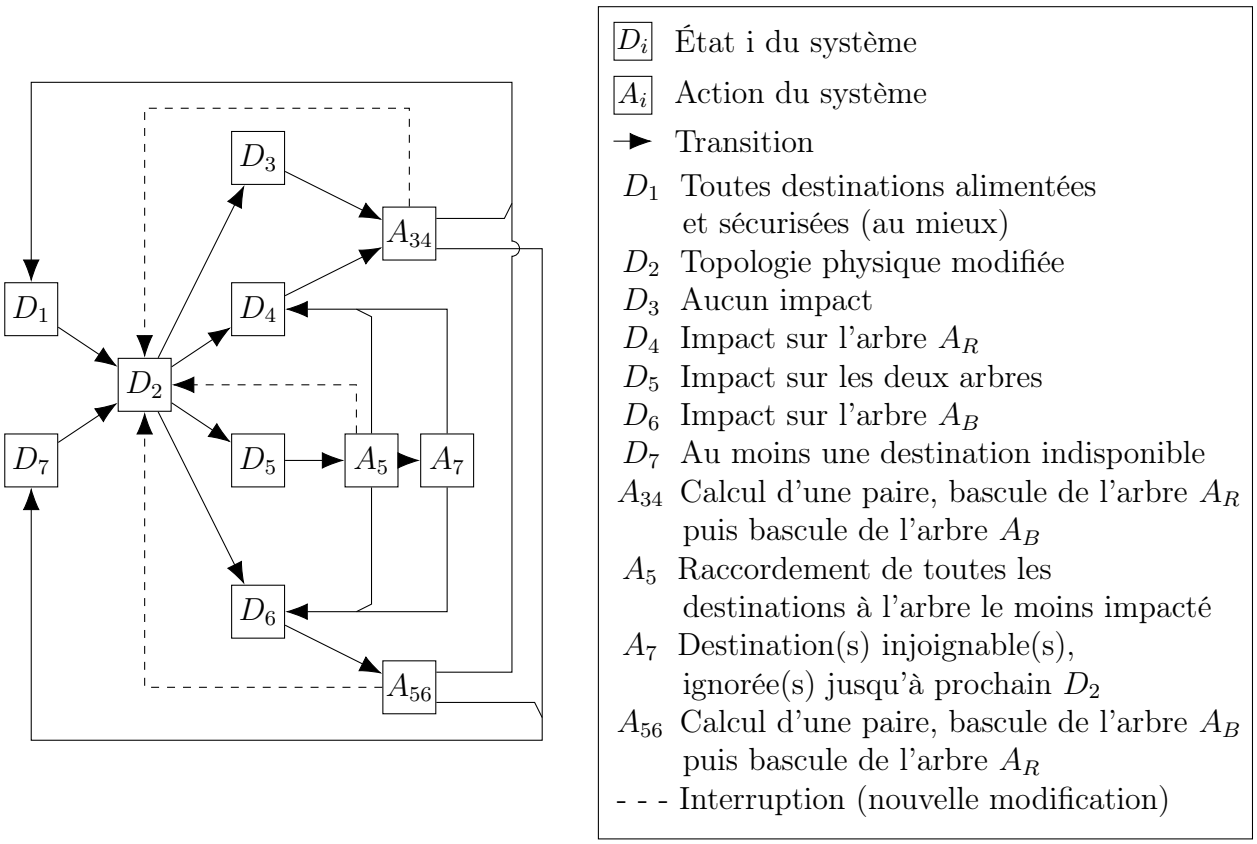


FIGURE 2.10 – Mécanisme de dynamisation de l'architecture

L'objectif est, si possible, de toujours revenir à l'état D_1, avec la meilleure solution possible. C'est pourquoi, si l'on atteint l'état D_3 qui n'est pas défaillant, on calcule à nouveau une paire d'arbres sur la nouvelle topologie.

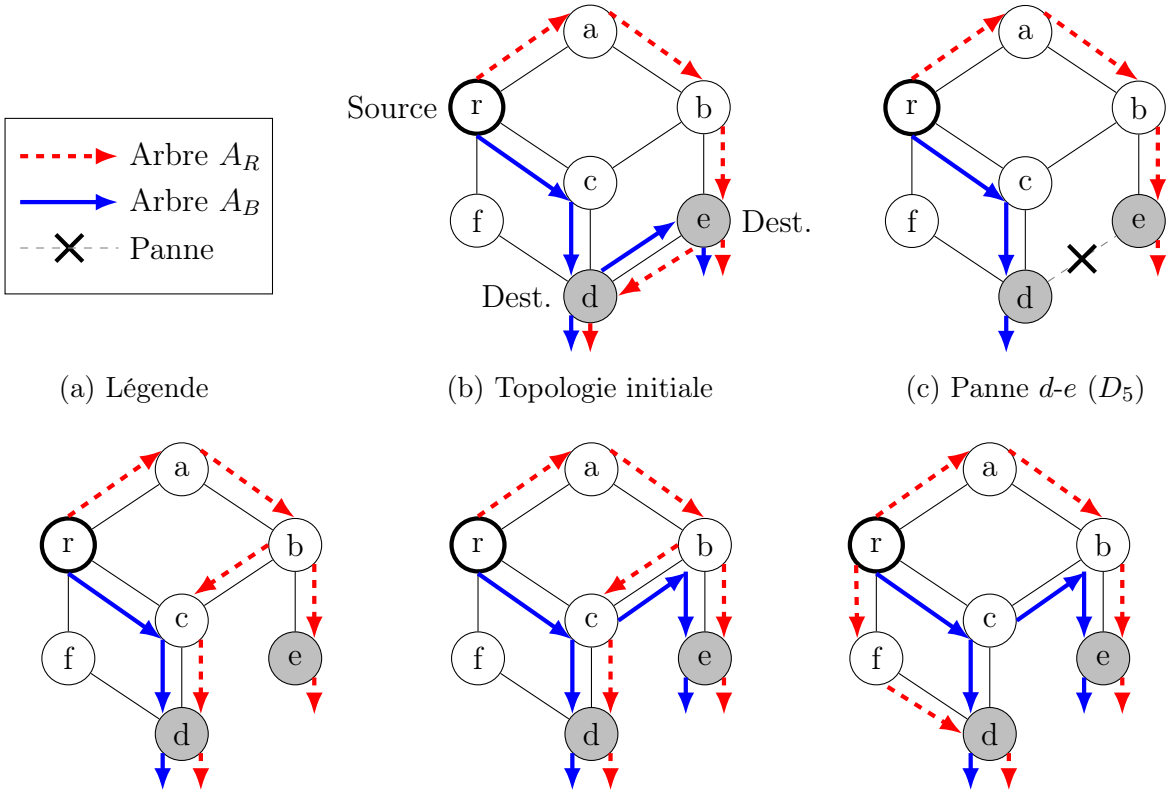
L'étape A_5 nécessite de rétablir la connexité complète d'un arbre. Cela peut-être effectué avec différentes méthodes, par routage de chemin ou de segment, avec différentes pondérations et ordre des destinations à traiter. Il s'agit de préparer l'un des arbres pour pouvoir effectuer ensuite une bascule sans impact.

Les actions de calcul et de déploiement peuvent être interrompues en cas de nouveau changement de topologie. L'idée est de ne pas déployer des topologies logiques erronées si la topologie physique a été modifiée en cours de calcul.

Contrairement au diagramme d'état présenté en figure 2.8, les états D_3, D_4, D_5 et D_6 peuvent mener à l'état D_7, car les algorithmes employés ont une faible probabilité d'échouer alors qu'une solution existe.

Il faut noter que des variantes de cette proposition sont possibles : en particulier, l'état D_3 peut mener à l'état A_{34} aussi bien qu'à l'état A_{56}, le mécanisme serait identique.

Un exemple d'application du mécanisme est présenté en figure 2.11.



(a) Légende (b) Topologie initiale (c) Panne *d*-*e* (D_5)

(d) Raccordement arbre A_R (A_1) (e) Bascule arbre A_B (A_{56}) (f) Bascule arbre A_R (A_{56})

FIGURE 2.11 – Exemple d'application du mécanisme de résilience

2.3.5 Temps de réaction et phénomènes de bagotage

Dans tous les schémas précédents, il faut prendre en compte les différents temps de réaction. Par exemple, dans le cas d'une panne courte, il n'est peut-être pas nécessaire d'effectuer un calcul complet, pour changer à nouveau de topologie quelques instants plus tard. Cela est d'autant plus vrai lors des phénomènes de fonctionnement intermittent d'une liaison. Par exemple, dans le cadre d'un Faisceau Hertzien, une mauvaise météo peut dégrader la liaison physique de manière intermittente. On parle alors de "bagotage".

Une manière de minimiser ces effets est d'ajouter une temporisation avant de déployer une nouvelle version de la topologie logique, si toutes les destinations sont alimentées par au moins un arbre. La topologie sera dans une situation non-sécurisée dans ce laps de temps, mais sera toujours fonctionnelle.

Il est également possible d'envisager de détecter les phénomènes de bagotage, et de choisir d'exclure la liaison en cause le temps que la situation redevienne stable.

2.4 Conclusion

Dans ce chapitre, le principe de l'architecture proposée appelée Seamless Multicast a été établi : pour transporter sur un réseau un flux audiovisuel temps réel, une paire résiliente d'arbres la moins coûteuse possible est déployée. Les arbres y transportent le flux de manière redondante, et sont maximalement indépendants et contraints en délai pour assurer un maximum de disponibilité.

Une formulation mathématique du problème DMDT associé a été proposée dans ces travaux de thèse. Le problème DMDT consiste au calcul d'un paire d'arbres redondants, optimisés en coût, contraints en longueur, et les plus indépendants possible.

Enfin, un mécanisme de dynamisation du déploiement d'une paire d'arbres a été proposé pour répondre au besoin de résilience. Ce mécanisme présuppose un procédé de calcul d'une solution au problème DMDT.

Afin d'implémenter l'architecture Seamless Multicast, étant donnée la $\mathcal{NP}$-difficulté du problème DMDT, la suite de ces travaux propose dans le chapitre 3 des heuristiques de calculs. Le chapitre 4 décrit l'architecture réseau proposée, associée à l'usage d'algorithmes centralisés et fondée sur le Software-Defined Networking.

Chapitre 3

Algorithmique associée au problème DMDT

Sommaire

3.1 Etat de l'art des approches par algorithmes d'exploration . . 39
3.2 Prérequis algorithmiques 40
3.2.1 Plus court chemin 40
3.2.2 Plus court chemin contraint en délai 40
3.2.3 Chemins totalement indépendants 41
3.2.4 SHERPA : Plus courts chemins contraints en Délai à Indépendance Maximale 43
3.3 Heuristique 1 : Iterative SHERPA (IS) 46
3.3.1 Procédé 46
3.3.2 Complexité temporelle 49
3.3.3 Variante à indépendance de nœud 49
3.3.4 Variante multi-source 49
3.4 Heuristique 2 : Red Tree First (RTF) 50
3.4.1 Procédé 50
3.4.2 Complexité temporelle 53
3.4.3 Variante à indépendance de nœud 53
3.4.4 Variante multi-source 53
3.5 Raffinage des pseudo-arbres 54
3.5.1 Procédé 54
3.5.2 Complexité 58
3.6 Algorithmes génétiques 60
3.6.1 Introduction 60
3.6.2 Etat de l'art des approches par algorithme génétique 61
3.6.3 Proposition d'un algorithme génétique 62
3.6.4 Réglage des paramètres 67
3.7 Expérimentations 71

3.7.1 Temps de calcul des heuristiques 71
3.7.2 Comparaison à des pseudo-optimums 73
3.7.3 Algorithme génétique appliqué 75
3.8 Conclusion 78

Dans ce chapitre, ces travaux de thèse proposent deux heuristiques pour résoudre le problème DMDT défini en section 2.2.2, ainsi qu'une approche par algorithme génétique. L'une des heuristiques nécessite notamment de définir un nouvel algorithme de calcul de chemins indépendants contraints en délai. Les algorithmes proposés sont évalués théoriquement et expérimentalement dans ce chapitre.

3.1 Etat de l'art des approches par algorithmes d'exploration

Dans l'état actuel de nos connaissances, aucune méthode ne permet de résoudre le problème DMDT tel que formulé en section 2.2. Il existe néanmoins des heuristiques permettant de calculer un arbre de Steiner contraint en délai [21] et même un algorithme fournissant l'optimum [22]. Cet état de l'art présentera les solutions les plus proches du problème défini.

Dans [23], est présentée une solution basée sur un algorithme génétique, permettant de générer une paire d'arbres recouvrants (Spanning Trees) indépendants. Ces arbres reliant tous les nœuds du graphe contiennent les arbres connectant uniquement les destinations souhaitées. La fonction objectif de cet algorithme représente la probabilité de panne, ainsi l'algorithme ne vise pas à optimiser l'usage de la bande passante, et n'est pas contraint. Cela s'explique par le contexte différent des réseaux de contrôles de systèmes, qui ne transportent pas le même type de données.

Dans [24], un algorithme est proposé permettant de calculer des arbres indépendants. Il peut prendre en compte le coût des arbres, ainsi qu'une contrainte en délai. Néanmoins, il exige qu'il existe au moins deux chemins indépendants entre toute paire de nœuds du graphe. De plus, l'algorithme se fonde sur un graphe non-dirigé, et ne permet pas de prendre en compte les différences de caractéristiques dans chaque direction d'un lien.

Un autre algorithme a été développé pour parer à ces défauts [25, 26, 27]. Plus général, il s'applique à des graphes dirigés, et pose moins de pré-requis sur la nature du graphe. Pour chaque destination il doit exister deux chemins strictement disjoints. Une partie de nos travaux a été d'adapter cet algorithme au problème DMDT mais sans succès.

Si l'état de l'art propose des algorithmes pour des problèmes proches du problème DMDT, ces travaux de thèse proposent deux nouveaux algorithmes pour le résoudre.

3.2 Prérequis algorithmiques

3.2.1 Plus court chemin

L'algorithme de Dijkstra [28] est un algorithme classique permettant de trouver le plus court chemin entre deux points d'un graphe avec des poids positifs. Sa complexité est de $T_{Dijkstra} = O(|E| + |V| * log(|V|))$.

À noter que dans le cas de graphes dont les poids des liens peuvent être négatifs, l'algorithme de Bellman-Ford [29] est à privilégier. En effet, l'algorithme de Dijkstra est incompatible avec des poids négatifs. Sa complexité est de $T_{BF} = O(|E| * |V|)$.

3.2.2 Plus court chemin contraint en délai

Fondés sur les algorithmes de plus court chemin cités en 3.2.1, deux algorithmes permettent d'imposer une contrainte en délai :

— LARAC [30] se base sur des utilisations successives de l'algorithme de Dijkstra. A chaque itération, la pondération du graphe est modifiée suivant le chemin trouvé et les délais obtenus. La complexité temporelle de cet algorithme est $T_{LARAC} = O(|E|^2 * log^4|E|)$ [30].

— CBF [31] est un algorithme d'exploration du graphe, qui prend en compte la notion de délai dans le choix des prochains sauts à explorer. Le cas le plus complexe est le cas où la contrainte est plus grande que n'importe quel délai au sein du graphe, c'est-à-dire lorsque l'algorithme CBF doit effectuer une exploration en largeur intégrale. Autrement dit $T_{CBF} = T_{BFS} = O(|V| + |E|)$.

Dans la suite de ce chapitre, pour plus de lisibilité, les descriptions utiliseront essentiellement l'algorithme CBF dans les notations. De plus, pour les expérimentations, CBF sera préféré pour sa plus faible complexité.

3.2.3 Chemins totalement indépendants

L'algorithme de Suurballe-Trajan [32] permet d'obtenir une paire de chemins totalement indépendants entre deux nœuds d'un graphe. Dans sa version originale, il exploite l'algorithme de Dijkstra [28], et permet d'assurer une indépendance totale de nœud ou de lien. Il nécessite néanmoins que de tels chemins existent, c'est-à-dire que le graphe soit suffisamment connecté.

Dans un graphe $G = (V, E)$, l'algorithme de Suurballe-Tarjan trouve, si possible, pour une paire $(s, t) \in V \times V$ donnée, deux chemins strictement indépendants en liens de s vers t. Les principales étapes plus détaillées de cet algorithme sont présentées ci-après, et détaillées dans l'Algorithme 3.1.

1. Un arbre des chemins les plus courts de la source s vers tous les nœuds de V est construit par l'algorithme de Dijkstra. Par définition il contient P_a le chemin vers la destination t ;
2. Un graphe résiduel est créé à partir du graphe initial : les liens de P_a sont supprimés, et le graphe repondéré (voir détails) ;
3. Un chemin P_b vers la destination t est calculé dans le graphe résiduel par l'algorithme de Dijkstra ;
4. Soit S l'ensemble des liens des chemins P_a et P_b ;
5. Les liens existant dans des directions opposées dans P_a et P_b sont supprimés de l'ensemble S ;
6. S contient P_A et P_B deux chemins totalement disjoints en liens les plus courts possible. En recherchant itérativement dans S les liens parents à partir de t, P_A et P_B sont extraits.

La variante de cet algorithme pour une indépendance de nœud nécessite une transformation préalable du graphe sur lequel est appliqué le calcul. Chaque nœud du graphe est remplacé par deux nœuds, l'un portant toutes les liaisons entrantes du nœud, l'autre toutes les liaisons sortantes, ainsi que leur poids respectifs. Les deux nœuds sont reliés par un lien unidirectionnel du nœud "entrant" vers le nœud "sortant", dont le poids est nul. Une fois le calcul effectué sur ce nouveau graphe par l'algorithme de Suurballe-Tarjan, les chemins trouvés doivent être reconvertis dans le graphe d'origine.

Algorithme 3.1 Algorithme de Suurballe-Tarjan

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0, une source s, une destination t.
Output : Deux chemins P_A et P_B totalement disjoints en liens les plus courts possible de s vers t.

1: **for** $v \in V$ **do**
2: $P \leftarrow Dijkstra(G_0, s, v)$
3: $d_0(v) \leftarrow C(P)$
4: **end for**
5: $P_a \leftarrow Dijkstra(G_0, s, t)$
6: $G_1 \leftarrow G_0$ graphe résiduel pondéré par C_1
7: $C_1 \leftarrow C_0$
8: **for** $(x, y) \in E$ **do**
9: $C_1(x, y) = C_0(x, y) - d_0(y) + d_0(x)$
10: **end for**
11: **for** $(x, y) \in P_a$ **do**
12: $C_1(y, x) \leftarrow C_1(x, y)$
13: $G_1 \leftarrow G_1 \setminus \{(x, y)\}$
14: **end for**
15: $P_b \leftarrow Dijkstra(G_1, s, t)$
16: $S \leftarrow P_a \cup P_b$
17: **for** $(x, y) \in P_a$ **do**
18: **if** $(y, x) \in P_b$ **then**
19: $S \leftarrow S \setminus \{(x, y), (y, x)\}$
20: **end if**
21: **end for**
22: $P_A \leftarrow \emptyset$
23: $v \leftarrow t$
24: **while** $v \neq s$ **do**
25: **for** $(x, y) \in S$ **do**
26: **if** $y = v$ **then**
27: $S \leftarrow S \setminus \{(x, y)\}$
28: $P_A \leftarrow P_A \cup \{(x, y)\}$
29: $v \leftarrow x$
30: **end if**
31: **end for**
32: **end while**
33: $P_B \leftarrow S$
34: **return** P_A, P_B

3.2.4 SHERPA : Plus courts chemins contraints en Délai à Indépendance Maximale

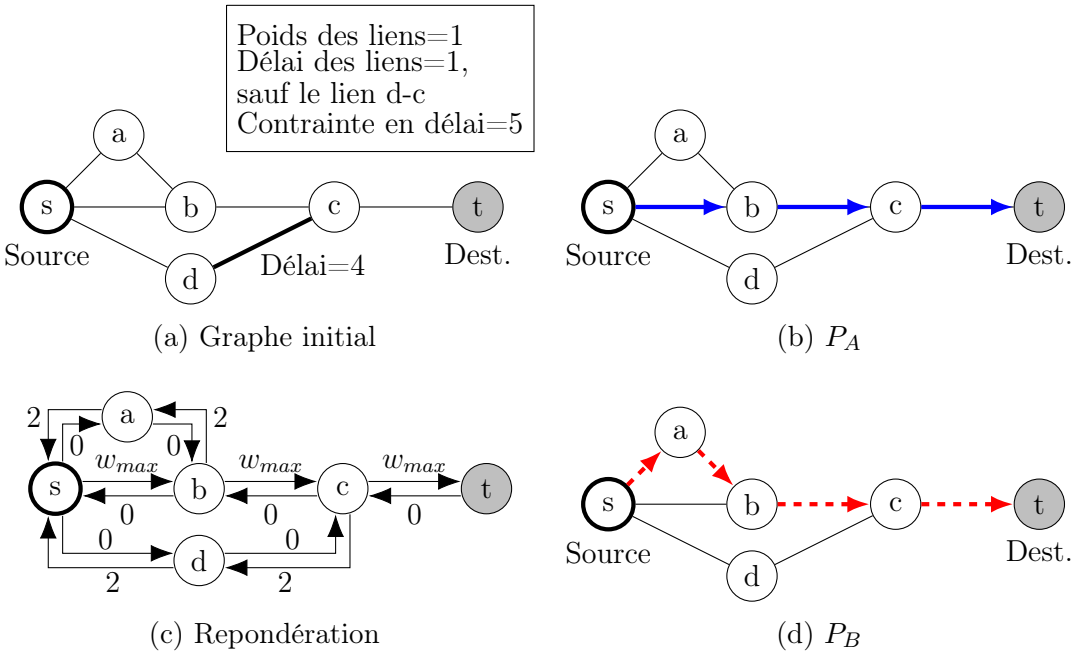
Les travaux présentés dans cette section proposent une variante de l'algorithme de Suurballe-Trajan, afin de résoudre le problème de chemins indépendants, en y ajoutant la contrainte de délai et en relâchant la contrainte d'indépendance totale. Nous noterons cet algorithme SHERPA (SHaring-Edges Restrained PAths). L'Algorithme 3.2 présente le procédé complet.

En d'autres termes, afin de relâcher la contrainte d'indépendance totale, l'étape 11 de l'algorithme de Suurballe-Tarjan original (Algorithme 3.1) est modifiée. Au lieu de modifier le graphe et de supprimer les liens à éviter, cet algorithme propose de les pondérer avec le poids total du graphe. Il est ainsi garanti qu'ils ne seront utilisés qu'en dernier recours. De plus, on crée un ensemble S_{AB} des liens communs entre les chemins P_a et P_b afin qu'ils soient intégrés au chemin P_B après l'extraction de P_A de S.

La contrainte de délai est prise en compte par l'utilisation d'un algorithme contraint en délai (comme LARAC ou CBF) au lieu de l'algorithme de Dijkstra.

De la même manière que l'algorithme de Suurballe-Tarjan, l'algorithme SHERPA peut-être appliqué sur un graphe modifié pour fournir des chemins à indépendance maximale de nœud. Dans la suite, on appellera cet algorithme NSHERPA.

Un exemple d'application de l'algorithme SHERPA est présenté en Figure 3.1.



(a) Graphe initial (b) P_A

(c) Repondération (d) P_B

FIGURE 3.1 – Illustration de l'algorithme SHERPA

La figure 3.1a présente le graphe sur lequel l'algorithme SHERPA est appliqué et ses caractéristiques. Le noeud s est défini comme source, et t comme destination. Le premier chemin le plus court calculé est illustré dans la figure 3.1b.

À partir de ce chemin et des distances respectives des noeuds à la source, le graphe est repondéré comme illustré dans la figure 3.1c.

Un second chemin est alors calculé suivant cette pondération. Le chemin le moins coûteux est celui passant par le noeud d, mais il ne respecterait pas la contrainte en délai fixée. Ainsi, la figure 3.1d illustre le second chemin calculé.

Dans cet exemple, l'étape de suppression de liens utilisés par les deux chemins dans des sens opposés, décrits par les lignes 18 à 23 de l'Algorithme 3.2, n'est pas illustrée.

Theorème 3.
La complexité temporelle de l'algorithme SHERPA est :

$$T_{SHERPA} = O(|V| * T_{CBF} + |E|) \tag{3.1}$$

Démonstration.
Complexité des différentes étapes de l'Algorithme 3.2 :

1-4 $O((|V|+1) * T_{CBF}) \sim O(|V| * T_{CBF})$, calcul de chemin vers chaque nœud.

8-10 $O(|E|)$, pondération de chaque lien.

11 $O(|E|)$, somme des poids de tous les liens.

12-15 $O(2 * |P_a|) \sim O(|E|)$ car $P_a \subset E$.

16 $O(T_{CBF})$.

19-23 $O(|P_a|) \sim O(|E|)$ car $P_a \subset E$.

26-34 Lors de chaque itération de la boucle "While", au moins une destination est retirée S après un parcours. Dans le pire des cas, le lien recherché est le dernier trouvé dans le parcours de S. On a ainsi $|S|$ itérations de la boucle "While", et un parcours de S avec $|S|$ diminuant à chaque itération. On a alors une étape en $O(\sum_{i=0}^{|S|} |S| - i) = O(|S|^2 - \sum_{i=1}^{|S|} i) \sim O(|S|^2) \sim O(|E|^2)$ car $S \subset E$.

La complexité totale est alors :

$$T_{SHERPA} = O((|V|+2) * T_{CBF} + 5 * |E|^2) \sim O(|V| * T_{CBF} + |E|^2)$$

□

Algorithme 3.2 SHERPA - Sharing-Edges Restrained Paths

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai, une source s, une destination t
et une contrainte en délai Δ.

Output : Deux chemins P_A et P_B les plus disjoints possible en liens les plus courts possible contraints en délai de s vers t.

1: **for** $v \in V$ **do**
2: $P \leftarrow CBF(G_0, s, v, \Delta)$
3: $d_0(v) \leftarrow C(P)$
4: **end for**
5: $P_a \leftarrow CBF(G_0, s, a, \Delta)$
6: $G_1 \leftarrow G_0$ graphe résiduel pondéré par C_1
7: $C_1 \leftarrow C_0$
8: **for** $(x, y) \in E$ **do**
9: $C_1(x, y) = C_0(x, y) - d_0(y) + d_0(x)$
10: **end for**
11: $C_{total} \leftarrow \sum_{i \in E} C_0(i)$, poids total de G_0
12: **for** $(x, y) \in P_a$ **do**
13: $C_1(x, y) \leftarrow C_{total}$
14: $C_1(y, x) \leftarrow 0$
15: **end for**
16: $P_b \leftarrow CBF(G_1, s, t, \Delta)$
17: $S \leftarrow P_a \cup P_b$
18: $S_{AB} \leftarrow P_a \cap P_b$
19: **for** $(x, y) \in P_a$ **do**
20: **if** $(y, x) \in P_b$ **then**
21: $S \leftarrow S \setminus \{(x, y), (y, x)\}$
22: **end if**
23: **end for**
24: $P_A \leftarrow \emptyset$
25: $v \leftarrow t$
26: **while** $v \neq s$ **do**
27: **for** $(x, y) \in S$ **do**
28: **if** $y = v$ **then**
29: $S \leftarrow S \setminus \{(x, y)\}$
30: $P_A \leftarrow P_A \cup \{(x, y)\}$
31: $v \leftarrow x$
32: **end if**
33: **end for**
34: **end while**
35: $P_B \leftarrow S \cup S_{AB}$
36: **return** P_A, P_B

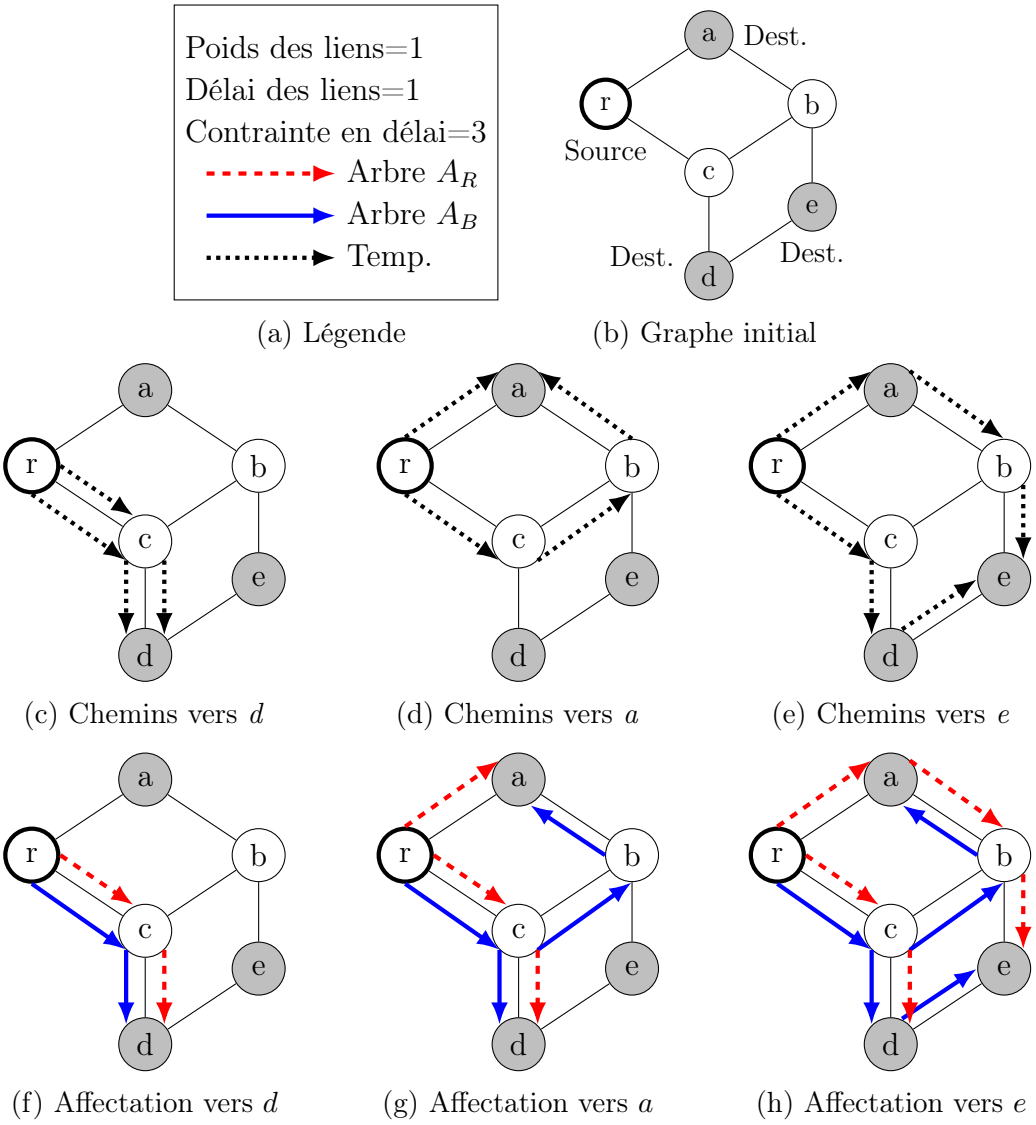
3.3 Heuristique 1 : Iterative SHERPA (IS)

3.3.1 Procédé

Le premier procédé de calcul proposé par ces travaux de thèse pour résoudre le problème DMDT est fondé sur l'algorithme SHERPA décrit en 3.2.4. Ce procédé est appelé IS ("Iterative SHERPA") pour plus de lisibilité. Le détail de cette heuristique est présenté en Algorithme 3.3 et se résume comme suit :

1. Pour chaque destination une paire de chemins est calculée par l'algorithme SHERPA ;
2. La liste des paires de chemin peut-être ordonnée ou non. Il existe un ordre optimal difficile à définir. On préférera ordonner les paires par distance croissante à la source, afin d'assurer le déterminisme de l'heuristique, bien qu'un ordre aléatoire puisse donner un meilleur résultat ;
3. Pour chaque paire, chacun des chemins appartenant à la paire est alors affecté à l'un des deux arbres :
   (a) de sorte à respecter la plus grande indépendance possible ;
   (b) en cas d'égalité, la destination n'est pas traitée.
4. Si au moins une destination n'est pas connectée à l'arbre, affecter aléatoirement la paire et reprendre à l'étape 3 pour toutes les paires non affectées.

Ce procédé tend à maximiser l'indépendance des arbres, en optimisant l'utilisation de lien communs au sein d'un arbre, le tout sous contrainte de délai. La Figure 3.2 illustre un exemple d'application de cette heuristique.



(a) Légende (b) Graphe initial

(c) Chemins vers d (d) Chemins vers a (e) Chemins vers e

(f) Affectation vers d (g) Affectation vers a (h) Affectation vers e

FIGURE 3.2 – Illustration algorithme IS

Les figures 3.2a et 3.2b présentent le graphe sur lequel l'algorithme va être appliqué, pour obtenir une paire d'arbres dont la source est le nœud r et les destinations sont a, d et e. Les figures 3.2c, 3.2d et 3.2e montrent les paires de chemins calculées suivant l'étape 1 de l'heuristique, respectivement vers les trois destinations d, a et e. Les paires sont ordonnées conformément à l'étape 2, suivant leur longueur cumulée et la longueur maximale au sein de la paire. Elles seront ainsi traitées dans l'ordre d, a et e.

En l'absence d'arbres existants, la paire vers d est affectée par défaut, comme visible sur la figure 3.2f. La paire vers a est ensuite traitée, mais suivant l'étape 3.b n'est pas affectée. De même, la paire vers e est ignorée. La paire vers a est alors affectée par défaut. Le tracé est alors illustré en figure 3.2g. Enfin, la paire vers e est affectée pour maximiser l'indépendance, et le tracé complet en figure 3.2h est obtenu. L'affectation inverse aurait impliqué le partage du lien r-a.

Algorithme 3.3 IS - Iterative SHERPA

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai, une source s, un ensemble de destinations T, une contrainte en délai Δ.

Output : Deux arbres A_R et A_B les plus disjoints possible en liens les plus léger possible sous contrainte de délai Δ de s vers T.

1: $A_r \leftarrow \emptyset$
2: $A_b \leftarrow \emptyset$
3: $p \leftarrow |T|$
4: $PP \leftarrow \emptyset$
5: **for** $t \in T$ **do**
6: $\quad P_{t,A}, P_{t,B} \leftarrow SHERPA(s, t, \Delta)$
7: $\quad PP \leftarrow PP \cup \{(P_{t,A}, P_{t,B})\}$
8: **end for**
9: Option : ordonner PP
10: **while** $PP \neq \emptyset$ **do**
11: $\quad (P_A, P_B) \leftarrow First(PP)$
12: $\quad$ **if** $|PP| = p$ **then**
13: $\quad\quad A_r \leftarrow A_r \cup P_A$
14: $\quad\quad A_b \leftarrow A_b \cup P_B$
15: $\quad\quad PP \leftarrow PP \setminus \{(P_A, P_B)\}$
16: $\quad\quad p = 0$
17: $\quad$ **else**
18: $\quad\quad$ **for** $(P_A, P_B) \in PP$ **do**
19: $\quad\quad\quad$ **if** $|(A_r \cup P_A) \cap (A_b \cup P_B)| > |(A_r \cup P_B) \cap (A_b \cup P_A)|$ **then**
20: $\quad\quad\quad\quad A_r \leftarrow A_r \cup P_A$
21: $\quad\quad\quad\quad A_b \leftarrow A_b \cup P_B$
22: $\quad\quad\quad\quad PP \leftarrow PP \setminus \{(P_A, P_B)\}$
23: $\quad\quad\quad$ **else if** $|(A_r \cup P_A) \cap (A_b \cup P_B)| < |(A_r \cup P_B) \cap (A_b \cup P_A)|$ **then**
24: $\quad\quad\quad\quad A_r \leftarrow A_r \cup P_B$
25: $\quad\quad\quad\quad A_b \leftarrow A_b \cup P_A$
26: $\quad\quad\quad\quad PP \leftarrow PP \setminus \{(P_A, P_B)\}$
27: $\quad\quad\quad$ **else**
28: $\quad\quad\quad\quad p = p + 1$
29: $\quad\quad\quad$ **end if**
30: $\quad\quad$ **end for**
31: $\quad$ **end if**
32: **end while**
33: $A_R \leftarrow Raffinage(A_r)$ (Alg. 3.5)
34: $A_B \leftarrow Raffinage(A_b)$ (Alg. 3.5)
35: **return** A_R, A_B

Il peut arriver que ce procédé ne fournisse pas d'arbres valides. C'est-à-dire que les deux arbres obtenus peuvent contenir des cycles, ou de multiples chemins vers un même nœud. Les étapes 33 et 34 de l'Algorithme 3.3 se rapportent à un algorithme de raffinage proposé en section 3.5.

3.3.2 Complexité temporelle

Theorème 4.
La complexité temporelle de l'algorithme IS est :

$$T_{IS} = O(|T| * |V| * T_{CBF} + |T| * |E| + |T|^2 + T_{Raffinage}) \tag{3.2}$$

Démonstration.

Complexité des différentes étapes de l'Algorithme 3.3 :

5-8 $O(|T| * T_{SHERPA})$.

9 Dépend de l'algorithme d'ordonnancement.

10-32 Au pire, à chaque itération de la boucle "While", un seul élément de PP est retiré. On a ainsi $|T|$ itérations, valeur initiale de $|PP|$. Au sein de cette boucle, la boucle "For" créera $|PP|$ opérations à chaque itération. Ainsi, cette partie de l'algorithme a une complexité $O(\sum_{i=1}^{|T|}(|T| - i)) \sim O(|T|^2)$.

33-34 $O(2 * T_{Raffinage})$.

On a alors :

$$T_{IS} = O(|T|*T_{SHERPA}+|T|^2+T_{Raffinage}) \sim O(|T|*(|V|*T_{CBF}+|E|)+|T|^2+T_{Raffinage})$$

□

3.3.3 Variante à indépendance de nœud

Afin d'obtenir une variante de cet algorithme permettant une indépendance de nœud, il suffit d'utiliser l'algorithme NSHERPA en lieu et place de l'algorithme SHERPA dans le processus.

3.3.4 Variante multi-source

Afin d'obtenir une variante de cet algorithme permettant la présence de plusieurs nœuds source, il faut utiliser une source fictive. Un nœud est ajouté au graphe, lié à chacune des sources. Le procédé est alors réalisé comme précédemment en utilisant comme source le nœud créé, en forçant l'affectation des chemins toujours du même côté. Une fois le procédé terminé, le nœud est retiré du graphe.

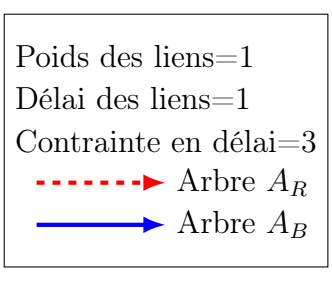
3.4 Heuristique 2 : Red Tree First (RTF)

3.4.1 Procédé

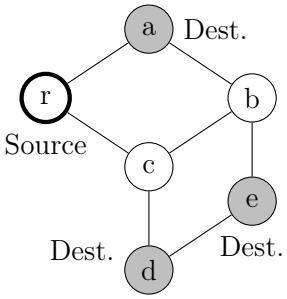
Le second procédé de calcul de deux arbres les plus disjoints possible contraints en délai est fondé sur l'algorithme LARAC. Un premier arbre est construit à l'aide de cet algorithme, puis un second arbre le plus indépendant possible du premier est calculé. Ce procédé est appelé RTF ("Red Tree First") pour plus de lisibilité, et est détaillé en Algorithme 3.4. Il peut se résumer comme suit :

1. Pour chaque destination qui n'est pas encore jointe par l'arbre A_R, l'algorithme CBF est appliqué entre la source et la destination ;
2. Le chemin le plus léger parmi ceux obtenus est ajouté à l'arbre A_R ;
3. La pondération du graphe est temporairement modifiée pour que les liens appartenant à l'arbre A_R aient un poids nul ;
4. S'il reste des destinations à joindre, revenir à l'étape 1 ;
5. La pondération du graphe est temporairement modifiée pour que les liens appartenant à l'arbre A_R aient un poids maximal (poids total du graphe) ;
6. Pour chaque destination qui n'est pas encore jointe par l'arbre A_B, l'algorithme CBF est appliqué entre la source et la destination ;
7. Le chemin le plus léger parmi ceux obtenus est ajouté à l'arbre A_B ;
8. La pondération du graphe est temporairement modifiée pour que les liens appartenant à l'arbre A_B aient un poids nul ;
9. S'il reste des destinations à joindre, revenir à l'étape 6.

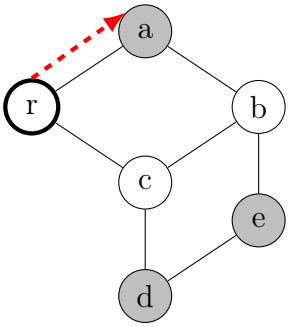
Ce procédé vise à obtenir des arbres les plus légers possible en priorisant la création du premier arbre, et maximise l'indépendance des arbres, le tout sous contrainte de délai. Un exemple d'application est présenté en Figure 3.3.



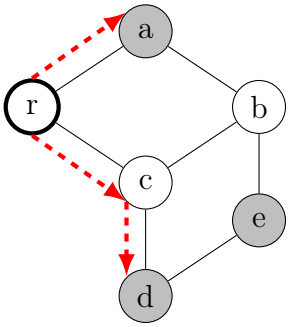
(a) Légende



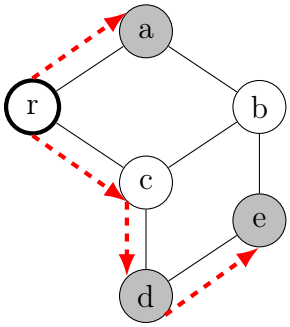
(b) Graphe initial



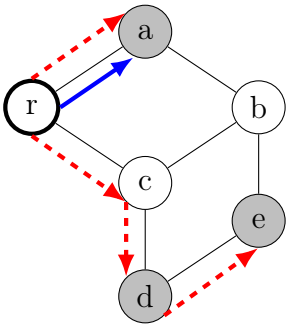
(c) Plus proche : a



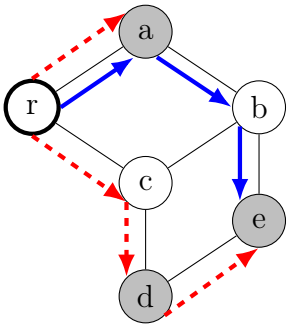
(d) Plus proche : d



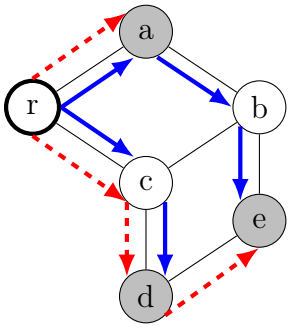
(e) Plus proche : e



(f) Plus proche : a



(g) Plus proche : e



(h) Plus proche : d

FIGURE 3.3 – Illustration algorithme RTF

Les figures 3.3a et 3.3b présentent le graphe sur lequel l'algorithme va être appliqué, pour obtenir une paire d'arbres dont la source est le nœud r et les destinations sont a, d et e. Dans le graphe initial, la destination la plus proche est a. Suivant les étapes 1 et 2 de l'heuristique, la route vers a est alors ajoutée à l'arbre A_R, comme illustré dans la figure 3.3c. Les routes vers d et e sont ensuite ajoutées dans les figures 3.3d et 3.3e.

Le graphe est alors repondéré conformément à l'étape 5. Tous les liens de l'arbre A_R sont pondérés par un poids maximal. Suivant ce nouveau graphe, la destination la plus proche est a. La figure 3.3f montre que la route vers a est ajoutée à l'arbre A_B suivant les étapes 6 et 7 de l'heuristique. La destination suivante la plus proche est e. En effet, d'après l'étape 8, le lien r-a a maintenant un poids nul, e est donc à une distance de $0+1+1=2$ de r alors que d est à deux fois le poids maximal. La route vers e est ajoutée à l'arbre A_B, suivant la figure 3.3g. Enfin, une nouvelle route est calculée vers d et ajoutée pour obtenir le tracé complet comme illustré par la figure 3.3h.

Algorithme 3.4 RTF - Red Tree First

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai, une source s, un ensemble de destinations T, une contrainte en délai Δ.

Output : Deux arbres A_R et A_B les plus disjoints possible en liens les plus léger possible sous contrainte de délai Δ de s vers T.

1: $T_r \leftarrow T$
2: **while** $T_r \neq \emptyset$ **do**
3: $P_{min} \leftarrow E$
4: **for** $t \in T_r$ **do**
5: $P_t \leftarrow CBF(G_0, s, t, \Delta)$
6: **if** $C_0(P_t) \leq C_0(P_{min})$ **then**
7: $P_{min} \leftarrow P_t$
8: $t_{min} \leftarrow t$
9: **end if**
10: **end for**
11: $T_r \leftarrow T_r \setminus \{t_{min}\}$
12: $A_r \leftarrow A_r \cup P_{min}$
13: **for** $(x, y) \in P_{min}$ **do**
14: $C_0(x, y) \leftarrow 0$
15: **end for**
16: **end while**
17: $G_1 \leftarrow G_0$ graphe pondéré par $C_1 \leftarrow C_0$
18: **for** $(x, y) \in A_r$ **do**
19: $C_1(x, y) \leftarrow \sum_{i \in E} C_0(i)$, poids total de G_0
20: **end for**
21: $T_b \leftarrow T$
22: **while** $T_b \neq \emptyset$ **do**
23: $P_{min} \leftarrow E$
24: **for** $t \in T_b$ **do**
25: $P_t \leftarrow CBF(G_1, s, t, \Delta)$
26: **if** $C_1(P_t) \leq C_1(P_{min})$ **then**
27: $P_{min} \leftarrow P_t$
28: $t_{min} \leftarrow t$
29: **end if**
30: **end for**
31: $T_b \leftarrow T_b \setminus \{t_{min}\}$
32: $A_b \leftarrow A_b \cup P_{min}$
33: **for** $(x, y) \in P_{min}$ **do**
34: $C(x, y) \leftarrow 0$
35: **end for**
36: **end while**
37: $A_R \leftarrow Raffinage(A_r)$ (Alg. 3.5)
38: $A_B \leftarrow Raffinage(A_b)$ (Alg. 3.5)
39: **return** A_R, A_B

Comme décrit précédemment pour l'heuristique IS, il est possible que l'heuristique RTF n'aboutisse pas à des arbres valides. Les étapes 37 et 38 de l'Algorithme 3.4 se rapportent au même algorithme de nettoyage présenté plus loin en 3.5.

3.4.2 Complexité temporelle

Theorème 5.
La complexité temporelle de l'algorithme RTF est :

$$T_{RTF} = O(|T|^2 * T_{CBF} + |T| * |E| + T_{Raffinage}) \tag{3.3}$$

Démonstration.

Complexité des différentes étapes de l'Algorithme 3.4 :

2-16 Lors de chaque itération de la boucle "While", une destination est retirée de T_r. On a ainsi $|T|$ itération. Les boucles intérieures (4-10 et 13-15) ont une complexité de $O(|T_r|)$ et $O(|E|)$. On a alors une étape en $O(\sum_{i=0}^{|T|}((|T| - i) * T_{CBF} + |E|)) \sim O(|T|^2 * T_{CBF} + |T| * |E|)$.

18-20 $O(|A_r|) \sim O(|E|)$ car $A_r \subset E$.

22-36 $O(|T|^2 * T_{CBF} + |T| * |E|)$ (voir 2-16).

37-38 $O(2 * T_{Raffinage})$.

On a alors :

$$\begin{aligned} T_{RTF} &= O(2 * (|T|^2 * T_{CBF} + |T| * |E|) + |E| + 2 * T_{Raffinage}) \\ &= O(|T|^2 * T_{CBF} + |T| * |E| + T_{Raffinage}) \end{aligned}$$

□

3.4.3 Variante à indépendance de nœud

Afin d'obtenir une variante de cet algorithme permettant une indépendance de nœud, il faut modifier l'étape 5 du résumé précédent : au lieu de pondérer uniquement les liens appartenant à l'arbre A_R par un poids maximal, il faut pondérer tous les liens entrants des nœuds appartenant à l'arbre A_R par un poids maximal. On garanti ainsi que ces liens seront évités à moins qu'ils appartiennent à l'unique chemin disponible, et donc que les nœuds seront évités.

3.4.4 Variante multi-source

Afin d'obtenir une variante de cet algorithme permettant d'assurer que chaque destination est alimentée par au moins deux sources distinctes parmi celles disponibles, il faut modifier l'étape 1 et 6 du résumé précédent. Lors de l'étape 1, il faut calculer le premier arbre à partir d'une source, en excluant l'autre source de la topologie. De même lors de l'étape 6, il faut calculer le second arbre en excluant la première source.

3.5 Raffinage des pseudo-arbres

Cette section décrit l'algorithme permettant de raffiner les résultats des heuristiques IS et RTF présentées plus haut afin d'éviter l'obtention de solutions non-valides. En effet, utilisés seules, ces heuristiques peuvent aboutir à des pseudo-arbres, respectant les contraintes du problème, mais pouvant contenir des cycles et de multiples chemins vers un même nœud.

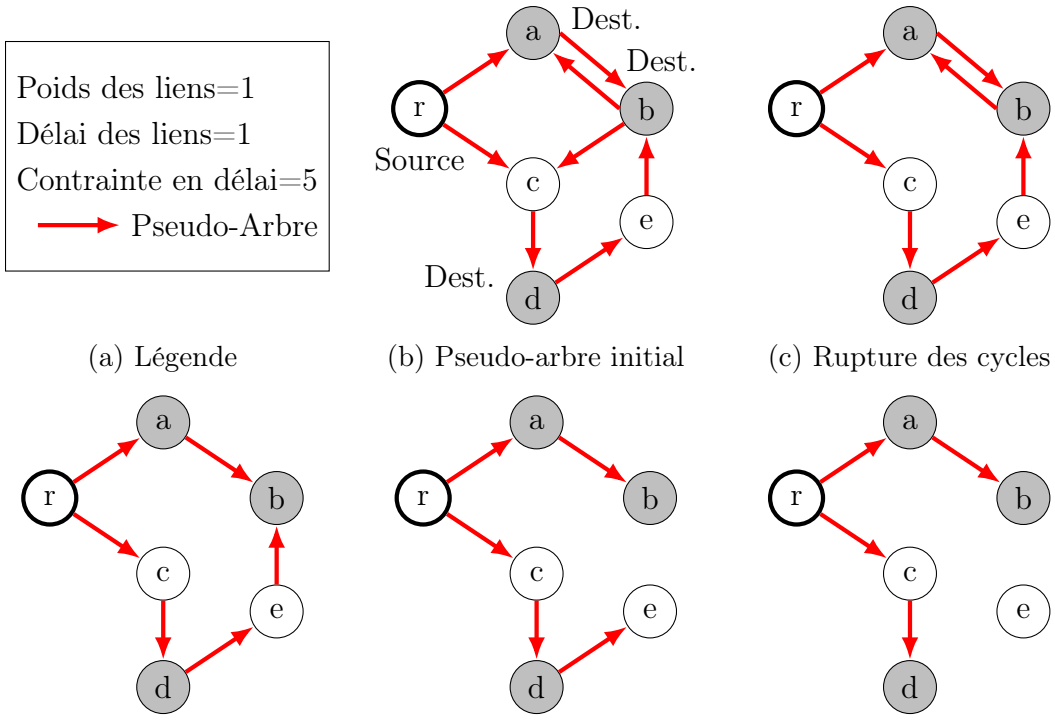
3.5.1 Procédé

Les Algorithmes 3.5 et 3.6 détaillent le processus, qui doit être appliqué à chacun des arbres de la paire calculée. Les principales étapes sont décrites ci-dessous :

1. À partir d'un pseudo-arbre, une exploration en profondeur du graphe correspondant (DFS) permet de détecter les cycles et de les éliminer en supprimant du pseudo-arbre le lien en direction de la racine (source) ;
2. Un traitement particulier est appliqué aux cycles de longueur 2, c'est-à-dire un lien dont les deux directions appartiennent au pseudo-arbre. Le lien conservé est celui dont le délai depuis la source est le plus court. Les cycles sont ainsi éliminés du pseudo-arbre ;
3. Pour chaque nœud possédant plus d'un prédécesseur, une fonction récursive permet de définir quel lien entrant doit être conservé, en se fondant sur le délai le plus court. La problématique des multiples chemins est ainsi résolue ;
4. Après ces suppressions, il peut rester des nœuds terminaux à l'arbre qui ne sont pas des destinations. Ces branches sont alors élaguées.

Le traitement des cycles de longueur 2 ne garantit pas le bon choix lorsque les cycles de longueur 2 sont consécutifs dans le pseudo-arbre, et que les délais des chemins depuis la source sont égaux. Ce cas n'apparaît pas en pratique, car une telle égalité n'existe pas. Si l'algorithme a un défaut théorique, ce défaut n'a aucun impact en pratique.

La Figure 3.4 présente un cas d'application complet de cet algorithme de raffinage. L'exemple est une illustration, en pratique aucune des deux heuristiques ne produit un tel pseudo-arbre sur un cas aussi trivial.



(a) Légende (b) Pseudo-arbre initial (c) Rupture des cycles

(d) Résolution des cycles de longueur 2 (e) Résolution des multiples chemins (f) Élagage des liens inutiles

FIGURE 3.4 – Illustration algorithme de raffinage

La figure 3.4b présente le pseudo-arbre sur lequel est appliqué l'algorithme de raffinage. Ce pseudo-arbre contient de cycles, et en particulier des cycles de longueur 2. Suivant l'étape 1 de l'algorithme, le cycle *c*-*d*-*e*-*b* est détecté, et le lien *b*-*c* est retiré. On obtient alors le pseudo-arbre de la figure 3.4c. L'arbre contient alors uniquement un cycle de longueur 2, *a*-*b*.

Le délai pour joindre le nœud *a* depuis la source à travers le lien appartenant au cycle vaut 5. À l'inverse, le délai pour joindre le nœud *b* en utilisant le lien *a*-*b* vaut 2. C'est donc le lien dans le sens *a*-*b* qui est conservé d'après l'étape 2 de l'algorithme de raffinage. Le tracé obtenu est alors celui présenté en figure 3.4d. Il n'existe alors plus de cycles dans le tracé, mais toujours deux chemins vers le nœud *b*.

Conformément à l'étape 3 de l'algorithme, il faut comparer les délais des deux chemins entrants de *b*. Il n'existe pas de nœud parent de *b* qui soit accessible par plus d'un chemin, l'utilisation d'une récursion n'est alors pas nécessaire. Le lien *e*-*b* est supprimé du tracé, car il correspond au chemin vers *b* le plus long, comme illustré en figure 3.4e.

Enfin, les liens inutiles sont retirés du graphe : *e* est une feuille de l'arbre, mais n'est pas un nœud destination. Le lien *d*-*e* est alors retiré du graphe, et le tracé final de la figure 3.4f est obtenu.

Algorithme 3.5 Algorithme de Raffinage IS et RTF

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai, une source s, un ensemble de destinations T, une contrainte en délai Δ. Un pseudo-arbre $A_0 \subset E$ de s vers T.

Output : Un arbre valide A_N de s vers T.

1: $\Theta \leftarrow$ Cycles identifiés par DFS sur A_0 à partir de s
2: $K_\Theta \leftarrow \{(a,b) \in E \mid (a,b)$ dernier lien d'un cycle $\theta, \forall \theta \in \Theta\}$
3: $\Theta_2 \leftarrow \{\theta \in \Theta \mid |\theta| = 2\}$
4: $A_1 \leftarrow A_0 \setminus K_\Theta$ pseudo-arbre sans cycle.
5: $A_2 \leftarrow A_1$
6: $U(v) \leftarrow \{u \in V \mid (u,v) \in A_2\}$ prédécesseurs de v dans A_2
7: **for** $(x,y),(y,x) \in \Theta_2$ **do**
8: $\quad A_2 \leftarrow A_2 \setminus \{(x,y),(y,x)\}$
9: $\quad$ **if** $|U_{A_2}(x)| = 0$ **then**
10: $\quad\quad A \leftarrow A_2 + \{(y,x)\}$
11: $\quad$ **else if** $|U_{A_2}(y)| = 0$ **then**
12: $\quad\quad A \leftarrow A_2 + \{(x,y)\}$
13: $\quad$ **else**
14: $\quad\quad P_x \leftarrow Dijkstra(A_2, s, x, D)$
15: $\quad\quad P_y \leftarrow Dijkstra(A_2, s, y, D)$
16: $\quad\quad d_x \leftarrow D(P_y) + D((y,x))$
17: $\quad\quad d_y \leftarrow D(P_x) + D((x,y))$
18: $\quad\quad$ **if** $d_x < d_y$ **then**
19: $\quad\quad\quad A_2 \leftarrow A_2 + \{(y,x)\}$
20: $\quad\quad$ **else**
21: $\quad\quad\quad A_2 \leftarrow A_2 + \{(x,y)\}$
22: $\quad\quad$ **end if**
23: $\quad$ **end if**
24: **end for**
25: $A_3 \leftarrow A_2$
26: $L_{A_3} \leftarrow \{v \in V \mid \nexists (v,w) \in A_3, \forall w \in V\}$ l'ensemble des feuilles de A_3
27: **for** $v \in L_{A_3}$ **do**
28: $\quad A_3 \leftarrow$ Chemin_unique(A_3, v) (Alg. 3.6)
29: **end for**
30: $A_4 \leftarrow A_3$
31: $L_{A_4} \leftarrow \{v \in V \mid \nexists (v,w) \in A_4, \forall w \in V\}$ l'ensemble des feuilles de A_4
32: $N \leftarrow L_{A_4} \setminus T$
33: **while** $N \neq \emptyset$ **do**
34: $\quad E_N \leftarrow \{(x,y) \in E \mid y = n, \forall n \in N\}$ l'ensemble des liens entrants de $n \in N$
35: $\quad A_4 \leftarrow A_4 \setminus E_N$
36: $\quad L_{A_4} \leftarrow \{v \in V \mid \nexists (v,w) \in A_4, \forall w \in V\}$ l'ensemble des feuilles de A_4
37: $\quad N \leftarrow L_{A_4} \setminus T$
38: **end while**
39: **return** A_4

Algorithme 3.6 Fonction de Chemin unique

Input : Un graphe dirigé $G_0 = (V, E)$ pondéré par C_0 et D respectivement coût et délai et une contrainte en délai Δ.
Un sous-graphe dirigé sans cycle $G_{A_1} = (V_{A_1}, A_1)$ défini par un ensemble de liens $A_1 \subset E$, une source $s \in V_{A_1}$, un noeud $m \in V_{A_1}$.

Output : Un ensemble $A_2 \subset A_1$ dans lequel il existe un et un seul chemin de s vers m le plus court en délai possible.

1: $G_{A_2} \leftarrow G_{A_1}$
2: $U_m \leftarrow \{u \in V_{A_2} \mid (u, m) \in A_2\}$ prédécesseurs de m dans A_2
3: $d_{min} \leftarrow \infty$
4: **for** $u_i \in U_m$ **do**
5: $\quad d_i \leftarrow 0$
6: $\quad c \leftarrow u_i$
7: $\quad$ **while** $c \neq s$ **do**
8: $\quad\quad U_c \leftarrow \{u \in V_{A_2} \mid (u, c) \in A_2\}$ prédécesseurs de c dans A_2
9: $\quad\quad$ **if** $|U_c| > 1$ **then**
10: $\quad\quad\quad A_2 \leftarrow$ Chemin_unique(A_2, c)
11: $\quad\quad$ **end if**
12: $\quad\quad$ Alors $U_c = \{b\}$ unique prédécesseur de c dans A_2
13: $\quad\quad d_i \leftarrow d_i + D(b, c)$
14: $\quad\quad c \leftarrow b$
15: $\quad$ **end while**
16: $\quad$ **if** $d_i < d_{min}$ **then**
17: $\quad\quad u_{min} \leftarrow u_i$
18: $\quad\quad d_{min} \leftarrow d_i$
19: $\quad$ **end if**
20: **end for**
21: **for** $u \in U_m \setminus \{u_{min}\}$ **do**
22: $\quad A_2 \leftarrow A_2 \setminus \{(u, m)\}$
23: **end for**
24: **return** A_2

3.5.2 Complexité

Theorème 6.
La complexité temporelle de l'algorithme Raffinage est :

$$T_{Raffinage} = O((|V| + |E|) * |V| * log(|V|) + |V|^3 * |E|^2) \tag{3.4}$$

Démonstration.

Complexité des différentes étapes de l'Algorithme 3.5 :

1 Cette étape repose sur l'algorithme de recherche en profondeur (DFS). Sa complexité est $T_{DFS} = O(|V| + |E|)$.

7-24 Cette boucle dépend du nombre de cycles de longueur 2, qui est limité par $2 * |E|$. Le cas le plus couteux au sein de cette boucle est le troisième (13-21), de complexité $O(2 * T_{Dijkstra})$. Cette étape a alors une complexité $O((|V| + |E|) * T_{Dijkstra}) = O((|V| + |E|) * (|E| + |V| * log(|V|)))$.

26 $O(|A_3| * |V|) \sim O(|E| * |V|)$ car $A_3 \subset E$.

27-29 La fonction *Chemin_unique* a une complexité $T_{CU} = O(|V|^2 * |E|^2)$, d'après le Théorème 7. Cette étape a alors une complexité $O(|V|^3 * |E|^2)$ car $L_{A_3} \subset V$.

31 $O(|A_4| * |V|) \sim O(|E| * |V|)$ car $A_4 \subset E$.

33-38 Cette étape consiste à retirer le dernier lien des branches dont la feuille n'est pas une destination de l'algorithme. À chaque itération de la boucle "While", au moins un lien est retiré du graphe, et par conséquent un nœud puisque l'étape précédente garanti l'unicité de chemin. On a donc au maximum $|V|$ itérations. Les opérations de la boucle ont une complexité $O(|E| * |V|)$. Cette étape à alors une complexité $O(|E| * |V|^2)$.

On a alors :

$$\begin{aligned} T_{Raffinage} &= O(|V| + |E| + (|V| + |E|) * (|E| + |V| * log(|V|)) + |E| * |V| \\ &\quad + |V|^3 * |E|^2 + |E| * |V| + |E| * |V|^2) \\ &\sim O((|V| + |E|) * |V| * log(|V|) + |V|^3 * |E|^2) \end{aligned}$$

□

Theorème 7.
La complexité temporelle de l'algorithme Chemin_unique est :

$$T_{CU} = O(|V|^2 * |E|^2) \tag{3.5}$$

Démonstration.
Complexité des différentes étapes de l'Algorithme 3.6 :

2 Nécessite un parcours de A_2, la complexité est de $O(|A_2|) \sim O(|E|)$.

4-20 La boucle "For" a un nombre d'itérations $|U_m|$ avec $U_m \subset V$. La boucle "While" utilise au maximum $|A_1| - 1$ itérations, puisque l'algorithme remonte les prédécesseurs d'un prédécesseur de m dans $A_2 \subset E$.

Les lignes 8 et 12 impliquent $O(2 * |E|)$ opérations à chaque itération de la boucle "While".

On a alors une complexité $O(|V| * |E| * |E| * T_{recursion})$.

Concernant le nombre d'appels récursif, une fois un nœud traité par la fonction, il n'a plus qu'un seul prédécesseur. S'il est à nouveau rencontré lors du parcours inverse du graphe, la fonction ne sera pas à nouveau appelée. On a ainsi un nombre d'appels à cette fonction limitée à $|V| - 1$ appels récursifs.

On a ainsi une complexité $O(|V|^2 * |E|^2)$.

21-23 $O(|Um| - 1) \sim O(|V|)$

On a alors :

$$T_{CU} = O(|E| + |V|^2 * |E|^2 + |V|) = O(|V|^2 * |E|^2)$$

□

3.6 Algorithmes génétiques

3.6.1 Introduction

Si les heuristiques présentées dans les parties précédentes permettent d'obtenir des solutions réalisables, il n'existe aucune garantie quand à la qualité du résultat en termes d'optimisation. Les Algorithmes Génétiques (AG) sont des algorithmes d'optimisation évolutionnaires, c'est-à-dire qu'ils sont fondés sur les concepts de l'évolution. Il s'agit d'une classe d'algorithmes utilisant le même principe, mais dont la composition dépend grandement du ou des problèmes traités. De nombreux ouvrages traitent du sujet, notamment [33]. Ces travaux de thèse tentent de proposer un algorithme génétique pour résoudre le problème DMDT.

Pour un problème d'optimisation donné, il existe un espace de solutions candidates réalisables ou non. Par réalisable, on entend qui correspond aux diverses contraintes du problème. Ces solutions peuvent être encodées sous forme d'une chaîne de caractères. Par exemple, dans le cas d'un programme linéaire à N inconnues, les solutions candidates peuvent être encodées comme une chaîne de longueur N, dont chaque élément correspond à une variable. De manière générale, on appelle ces chaînes encodées des "chromosomes". Chaque élément composant la chaîne d'un chromosome est appelé "gène" et peut prendre différentes valeurs appelées "allèles".

Les AG ont pour objectif d'explorer l'espace des solutions candidates à l'aide des chromosomes de manière itérative. À chaque itération, une population de chromosomes est générée. Trois types d'opérateurs sont appliqués à chaque génération :

- croisement : deux chromosomes sont combinés pour donner naissance à un ou plusieurs nouveaux individus ;
- mutation : certains gènes des nouveaux individus sont modifiés aléatoirement ;
- reproduction : des chromosomes sont conservés dans la nouvelle génération.

À chaque AG correspond une fonction d'évaluation : c'est la fonction qui définit la qualité de chaque solution candidate par rapport au problème donné. Dans le cas d'un programme linéaire, cela correspond à la fonction de coût. Suivant les implémentations, la fonction d'évaluation peut être croissante ou décroissante, et doit être adaptée suivant le problème. Elle est employée pour diriger l'exploration des solutions vers un optimum car elle permet de discriminer les individus. En effet, le croisement et la reproduction qui en dépendent assurent l'optimisation locale, tandis que la mutation assure l'exploration de tout l'espace des solutions.

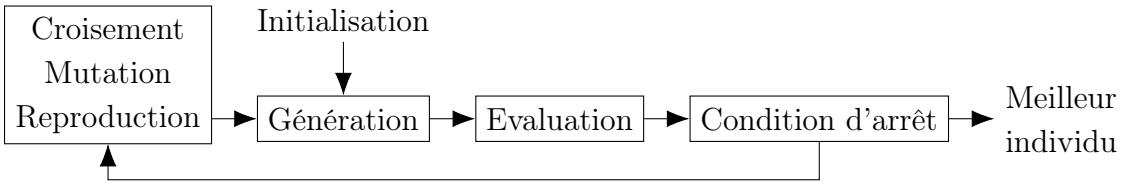


FIGURE 3.5 – Principe de fonctionnement d'un algorithme génétique

Les AG sont généralement implémentés suivant le processus suivant, repris en figure 3.5 :

1. Création d'une population initiale
2. Évaluation de chaque individu
3. Création d'une nouvelle génération
   (a) Sélection de parents et croisement
   (b) Mutations aléatoires
   (c) Reproduction
4. Évaluation de chaque individu
5. Si la condition d'arrêt est satisfaite, retourner le meilleur individu ;
   sinon, revenir à l'étape 3.

3.6.2 Etat de l'art des approches par algorithme génétique

Un état de l'art sur les algorithmes d'explorations a déjà été présenté en section 3.1.

Concernant les AG, le problème du calcul d'arbres de Steiner a déjà été largement couvert, notamment dans [34].

En particulier, le calcul d'arbres contraints en délai a été abordé dans [35] et [36] qui proposent d'utiliser une structure d'arbre comme encodage. Il s'agit de manipuler directement l'arbre plutôt qu'un chromosome encodant l'arbre. De même, [37] utilise cette technique pour l'établissement de Spanning Tree sous contraintes.

L'encodage proposé dans [38] est relativement complexe, mais plus adapté au calcul de Spanning Trees. De même, les travaux présentés dans [23] se concentrent sur le calcul d'une paire de Spanning Trees disjoints, et l'encodage est peu adapté à notre cas.

De manière similaire aux algorithmes d'exploration, il n'existe pas à notre connaissance une solution au problème DMDT par AG dans la littérature. Ces travaux de thèse cherchent à proposer un AG conçu pour résoudre le problème DMDT.

3.6.3 Proposition d'un algorithme génétique

Encodage des chromosomes

Ces travaux de thèse proposent d'utiliser un encodage "Path-oriented" défini dans [39] et utilisé dans [34]. Chaque gène correspond à un couple source-destination.

Pour chaque paire source-destination, une table des routes possibles est calculée au préalable grâce à un parcours en profondeur du graphe (DFS) modifié. Cela permet de trouver toutes les routes possibles dites "simples", c'est-à-dire sans cycles. Il est alors possible d'introduire une contrainte en délai en supprimant les routes trop longues. Afin de réduire la complexité globale de l'AG, il faut limiter la taille de ces tables des routes. On verra néanmoins plus tard que cela irait à l'encontre de l'objectif d'une optimisation fine.

Les gènes sont des entiers correspondants au numéro de la route choisie dans la table associée à la paire source-destination. Cette méthode d'encodage est illustrée en figure 3.6

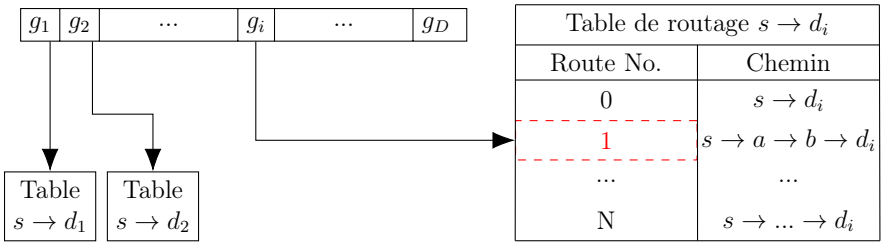


FIGURE 3.6 – Encodage "Path-oriented"

Étant donné que les deux arbres doivent joindre toutes les $|T|$ destinations, il faut deux gènes par destination, soit des chromosomes de taille $2 * |T|$. L'ordre des gènes dans le chromosome est défini suivant deux approches. Il est possible encoder un arbre puis le second, on parlera alors d'encodage concaténé. On peut également choisir d'alterner les gènes pour chaque arbre, on parlera alors d'encodage alterné. Ces types d'encodages sont présentés dans la figure 3.7.

Dest.	a	b	...	z	a	b	...	z
Route	3	2	...	2	1	7	...	4

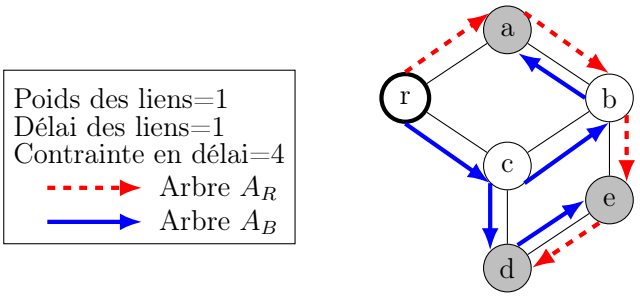
(a) Encodage concaténé

Dest.	a	a	b	b	...	z	z
Route	3	1	2	7	...	2	4

(b) Encodage alterné

FIGURE 3.7 – Encodages des deux arbres

La figure 3.8 présente un exemple d'encodage concaténé d'une paire d'arbres suivant le procédé décrit.



Chromosome (a, d, e) :

1	4	3	2	1	1

Table des routes $r \to a$

N.	Route
1	$r \to a$
2	$r \to c \to b \to a$

Table des routes $r \to d$

N.	Route
1	$r \to c \to d$
2	$r \to a \to b \to c \to d$
3	$r \to c \to b \to e \to d$
4	$r \to a \to b \to e \to d$

Table des routes $r \to e$

N.	Route
1	$r \to c \to d \to e$
2	$r \to c \to b \to e$
3	$r \to a \to b \to e$
4	~~$r \to a \to b \to c \to d \to e$~~

FIGURE 3.8 – Exemple d'encodage concaténé

Cet encodage ne garantit pas des individus viables : il est possible que des cycles soient générés. Ce phénomène sera néanmoins atténué par le fait que les arbres contenant des boucles sont plus coûteux et seront éliminés. D'autre part, il est possible d'éliminer ces individus après les étapes de mutation et croisement, en conservant à la place des individus de la génération précédente.

On peut noter que cet encodage a pour défaut qu'une solution peut-être encodée de deux manières, suivant l'ordre dans lequel les arbres sont placés. On peut ainsi obtenir des phénotypes identiques pour deux génotypes différents. Dans une certaine mesure, cela peut fausser la notion de diversité de la population.

Fonction d'évaluation

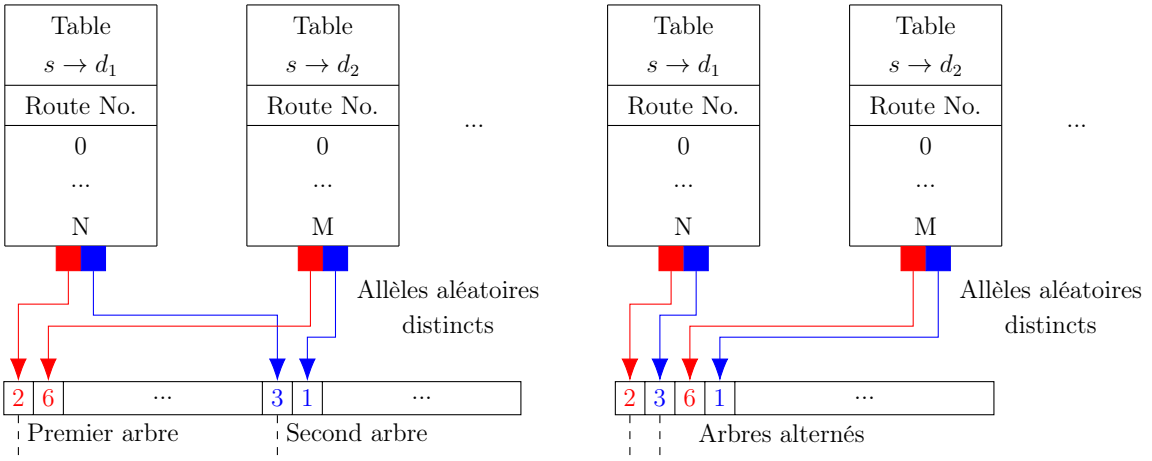
La continuité des arbres ainsi que la contrainte en délai sont assurées à la création des tables de routage. Afin d'assurer la cohérence de notation dans cette section, on cherchera à définir une fonction d'évaluation telle que les meilleurs individus aient une plus grande valeur d'adaptation. On proposera alors d'utiliser la fonction de coût définissant le problème DMDT.

Soient R et B deux vecteurs binaires représentants les deux arbres recherchés A_R et A_B. Chaque élément d'un vecteur représente l'utilisation d'un lien $(x, y) \in E$. Leur valeur est notée $R(x, y)$ et $B(x, y)$. Elle vaut 1 si le lien appartient à l'arbre correspondant, 0 sinon. Soit Z un vecteur dont les valeurs représentent chacune l'utilisation d'un lien $(x, y) \in E$ par les deux arbres A_R et A_B pour joindre une même destination. Cette valeur est nulle si le lien n'est pas commun aux deux arbres, sinon elle correspond au nombre de destinations jointes par les deux arbres en passant par ce lien.

$$\mathcal{F}(A_R, A_B) = -\sum_{(x,y)\in E} C_0(x,y)(R(x,y) + B(x,y)) - 2\sum_{(x,y)\in E} C_0(x,y) \sum_{(x,y)\in E} Z(x,y) \quad (3.6)$$

Population initiale

Des chromosomes aléatoires sont créés en sélectionnant aléatoirement des routes dans les tables de routes, voir figure 3.9. On a ainsi, dès le départ, une grande diversité dans la population.



(a) Encodage concaténé (b) Encodage alterné

FIGURE 3.9 – Génération d'un individu aléatoire

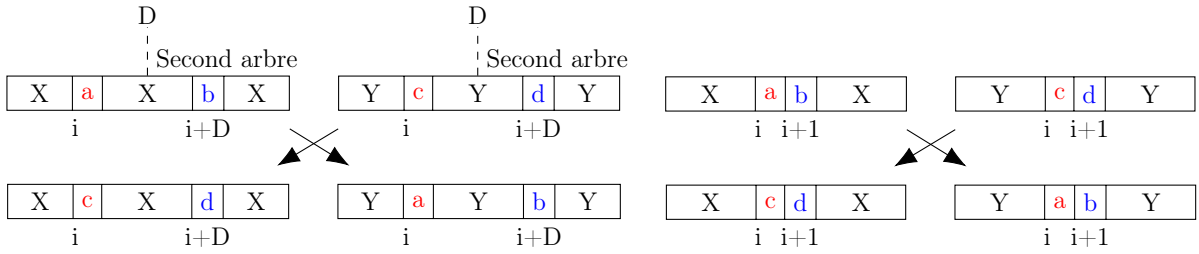
On peut introduire dans la population deux individus particuliers issus d'autres heuristiques. Cette technique est appelée "seeding", et est étudiée dans la littérature, notamment dans [40] et [41].

Ces individus sont des solutions réalisables, ayant par défaut une meilleure valeur d'adaptation que des individus aléatoires. Ils fournissent une limite basse pour l'optimum. Ils sont néanmoins différents l'un de l'autre en termes de génotype et phénotype. On diminue ainsi le temps nécessaire à la convergence, tout en conservant la diversité nécessaire à l'exploration de l'espace des solutions. De plus, cela garantit que des individus d'une certaine qualité sont déjà présents dans la population initiale, pouvant fournir une solution rapide si nécessaire.

Croisement

Usuellement, une méthode de sélection aléatoire de parents est appliquée, avec équiprobabilité : "Roulette Wheel". À chaque génération, des doublons peuvent apparaître et être conservés dans la population s'ils possèdent une bonne valeur d'adaptation, un mécanisme d'élitisme est alors nécessaire. Les meilleurs individus ont statistiquement de plus en plus de chances de se reproduire.

On choisit de croiser les chemins utilisés par les deux arbres : il faut alors échanger deux gènes, voir figure 3.10. On peut imaginer qu'une paire d'arbres de bonne qualité contient une paire de chemins peu coûteux et disjoints ; il faut alors envisager d'échanger par paires lors du croisement. L'opération est *a priori* moins coûteuse si les deux chromosomes sont proches, c'est-à-dire si l'encodage alterne les arbres.



(a) Encodage concaténé (b) Encodage alterné

FIGURE 3.10 – Croisement par paires

Mutation

L'opérateur de mutation choisi dans ces travaux de thèse est une mutation aléatoire. Un gène aléatoire prend l'une de ses valeurs possibles aléatoirement, comme illustré dans la figure 3.11. La mutation fait ainsi apparaître de nouvelles routes dans la population.

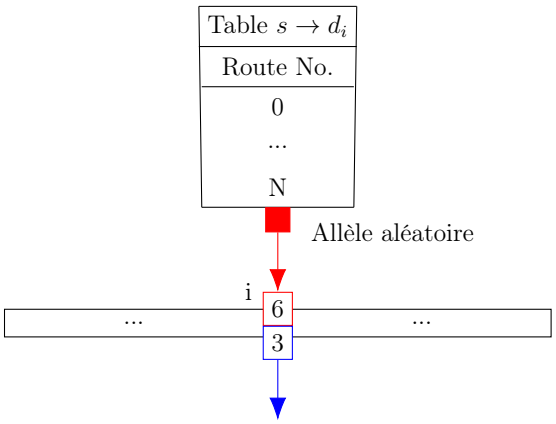


FIGURE 3.11 – Mutation aléatoire d'un individu

Sélection des individus reproduits

On se fondera sur une méthode élitiste : une partie des individus ayant les meilleures valeurs d'adaptation sera conservée dans la population. Le reste sera remplacé par les résultats des croisements et mutations. Le taux de reproduction est implicitement défini si le taux de croisement $t_{croisement}$ est différent de 1 par $t_{reproduction} = 1 - t_{croisement}$.

À noter que les nouveaux individus de la population ne doivent pas contenir de doublons, ou être identiques à l'un des individus reproduits.

Afin d'éviter une convergence prématurée, on pourrait également envisager l'insertion régulière d'un étranger (individu aléatoire) dans la population. Néanmoins, un étranger a peu de chance d'être sélectionné comme parent et ne déstabilisera pas suffisamment la population.

Condition d'arrêt

Deux conditions d'arrêt sont envisageables suivant les usages :
- un nombre fixé de générations ou un temps de calcul prédéfini : pour des contraintes opérationnelles, il peut être nécessaire de limiter le temps de calcul ;
- si le meilleur individu de la population n'a pas évolué pendant plusieurs générations. On peut alors considérer que la solution a été trouvée ou que la convergence est prématurée. Pour s'en assurer, on peut augmenter les chances de mutation du meilleur individu pour déstabiliser l'optimum local trouvé, ou insérer un étranger.

Comme mentionné plus haut, ni la fonction d'évaluation ni l'encodage ne garantissent l'absence de cycles ou de multiples chemins. Si l'algorithme génétique est interrompu avant une convergence complète, il se peut que de tels défauts existent. Il peut alors être nécessaire de raffiner la solution à l'aide de l'algorithme 3.5.

Variante à indépendance de noeud

Tel que présenté, l'algorithme assure une indépendance maximale de lien par destination. Il est possible de l'adapter pour une contrainte plus forte qui est l'indépendance de noeud. Il faut modifier la fonction d'évaluation en remplaçant Z représentant les liens communs par destination, par Z_{noeuds} le nombre de noeuds communs par destination entre les arbres. Si des heuristiques sont utilisées pour effectuer du "seeding", il faut prendre leur variante à indépendance de noeud.

L'avantage d'une indépendance de noeud est une plus grande robustesse aux pannes, mais la création d'arbres perd en souplesse. La faisabilité d'une telle indépendance dépend de la densité du graphe, c'est-à-dire du maillage du réseau.

3.6.4 Réglage des paramètres

Introduction

On appelle paramètres tous les éléments qui définissent une implémentation d'algorithme génétique : taille de population, probabilités de croisement et de mutation, opérateurs de croisement et de mutation, élitisme, sélection des parents... On les distingue comme paramètres quantitatifs et qualitatifs.

Eiben et Smit [42] ont rassemblé de nombreuses références et résumé le problème du paramétrage des algorithmes génétiques (PTP "Parameter Tuning Problem"), et plus généralement des algorithmes évolutionnaires.

Ce qui qualifie les performances d'un AG dépend de la méthode d'évaluation :
— Mean Best Fitness (MBF) : Pour un temps (ou opérations) donné, quelle qualité de solution est atteinte par l'AG ?
— Average number of Evaluation to Solution (AES) : Pour une qualité de solution donnée, combien de temps (ou opérations) l'AG atteint-il cette qualité ?
— Success Rate (SR) : Pour un temps et une qualité donnée, l'AG parvient-il à fournir une solution ?

Il faut également définir si l'AG est un spécialiste (pour un problème donné) ou un généraliste (pour toute une classe de problèmes). À noter que d'après le théorème "No Free Lunch", il n'existe pas d'algorithme qui soit la solution générale à tous les problèmes, et qu'un algorithme performant suivant certaines hypothèses le sera moins que d'autres sur des problèmes où ces hypothèses ne sont pas respectées.

Le choix des paramètres influe sur les performances de l'AG. On parle alors d'utilité ou d'intérêt des paramètres.

Il existe deux types de méthodes de paramétrage : les méthodes "offline", c'est-à-dire avant que l'algorithme ne soit exploité, et les méthodes "online" qui modifient les paramètres pendant les itérations de l'algorithme (également appelées "Self-adaptative").

Les méthodes "offline" de paramétrage des AG ou "tuners" sont fondées sur un même principe : générer un jeu de paramètres et les tester afin d'établir leur intérêt.

L'une des méthodes les plus communes est l'utilisation d'un meta-AG pour résoudre le problème de paramétrage : d'une part, il est adapté à ce type de problème ; d'autre part, il permet de mettre facilement en évidence la sensibilité et l'impact de chaque paramètre. L'intérêt des meta-AG est discuté dans [43].

Il existe également des méthodes dites de "racing", qui mettent en compétition différents paramètres pour éliminer plus rapidement les individus plus faibles. Elles ont pour défaut d'être limitées à l'ensemble de solutions proposé à l'initialisation.

On notera en particulier la méthode proposée dans [44] qui combine meta-AG et "racing". L'avantage du "racing" est qu'il permet d'évaluer l'AG par rapport à des paramètres pour lesquels il est difficile de définir un espace de recherche, en particulier les paramètres qualitatifs.

Les méthodes "online" ou "self-adaptative" ou avec contrôle de paramètres sont des méthodes qui ajustent les paramètres dynamiquement pendant le fonctionnement de l'algorithme.

Un mécanisme simple peut-être observé dans [45] et [46] : l'estimation des valeurs d'adaptation minimale, maximale et moyenne de la population permet de définir la dispersion de la population et de modifier les paramètres de mutation et croisement en conséquence. Ce schéma est adapté pour un opérateur de mutation et un opérateur de croisement. Les deux opérateurs évoluent dans la même direction, et rien ne permettrait de différencier les deux opérateurs proposés ici. De plus, la méthode [45] nécessite malgré tout de définir d'autres paramètres.

On peut également mentionner les travaux de [47] dans lesquels, au cours d'un AG possédant plusieurs opérateurs de mutation, les probabilités évoluent au fil des générations suivant les progrès procurés par chacun.

À noter, [46] applique une méthode "offline" à l'algorithme adaptatif pour évaluer les opérateurs les plus pertinents.

Ces méthodes sont plus adaptées à des algorithmes dédiés à une seule utilisation. En effet, l'effort de paramétrage "offline" n'est pertinent que si l'algorithme doit être utilisé plusieurs fois, sur toute une gamme de problèmes.

Paramètres à définir

Les paramètres qualitatifs de l'algorithme génétique présenté dans ces travaux ont déjà été établis plus haut, contraints par l'encodage et la nature du problème. La Table 3.1 résume tous les paramètres et leurs valeurs.

TABLE 3.1 – Paramètres de l'algorithme génétique

Paramètre	**Type**	**Valeur(s)**
Encodage	Qualitatif	"Path-oriented"
Croisement	Qualitatif	Par paire
Mutation	Qualitatif	Aléatoire
Sélection de parents	Qualitatif	Aléatoire
Reproduction	Qualitatif	Élitisme
Taille de la population	Quantitatif	$P \in \mathbb{N}^{+*}$
Probabilité de mutation	Quantitatif	$p_{random} \in]0, 1[$
Taux de croisement	Quantitatif	$t_{croisement} \in]0, 1[$
Taux de reproduction	Quantitatif	$1 - t_{croisement}$

Dans cette partie, on cherchera à définir un paramétrage spécialiste pour une gamme de problèmes correspondant au cas réel d'application.

On a donc un problème d'optimisation à trois variables non-indépendantes dont les valeurs peuvent prendre un nombre infini de valeurs :

— taille de la population ;
— taux de croisement ;
— probabilité de mutation.

La taille de la population est un paramètre qui doit être limité : un grande population donne de meilleurs résultats dûs à la diversité de la population, mais augmente la complexité du calcul. En particulier, si la population est composée de tous les individus possibles, alors l'AG est l'équivalent d'une solution par force brute. Dans le cas de l'encodage choisi, le nombre total d'individus possible est P_{max} défini dans l'équation 3.7 :

$$P_{max} = 2 * \prod_{i=1}^{|T|} r_i \text{ avec } r_i \text{ le nombre de routes possibles pour la destination } i \qquad (3.7)$$

À noter qu'il existe des mécanismes qui définissent la taille de population de manière dynamique [48].

Ces travaux se limiteront à trouver des paramètres utiles aux types de problèmes pouvant être rencontrés dans le contexte industriel, décrits en section 3.7.

Proposition de méthode

Dans un premier temps, ces travaux proposent d'utiliser un meta-AG dont les paramètres sont arbitrairement fixés aux valeurs suivantes :

— fonction d'évaluation : MBF sur des instances représentatives ;
— chromosome à 3 gènes (voir figure 3.12) ;
— mutation aléatoire avec une probabilité 0.1 ;
— croisement par échange d'un gène ;
— élitisme avec un taux de reproduction de 0.2 ;
— population de 20 individus ;
— condition d'arrêt de 1000 générations.

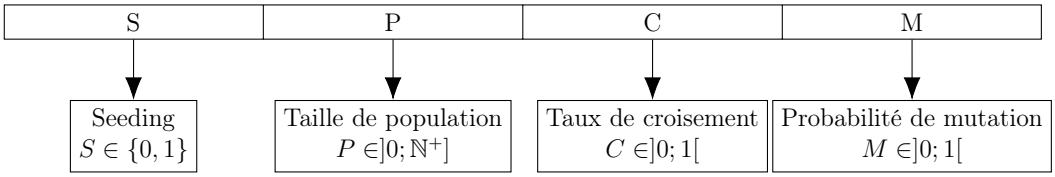


FIGURE 3.12 – Encodage des paramètres pour le meta-AG

Néanmoins, dans le cas d'une paramétrisation d'un AG pour une gamme de problème, une faible variation d'un paramètre aura peu d'influence sur l'utilité de ce paramètre. En effet, les valeurs de paramètre choisies ne sont pas optimales pour une problème donné, mais pour un ensemble de caractéristiques de problèmes. Par exemple, la différence de performance si la taille de population de l'AG est de 100 ou 101 est très faible. De même,

sur une taille de chromosome de 20, une faible variation de probabilité de mutation de 0.1 à 0.15 change l'espérance du nombre de gènes ayant muté de 1 (loi binomiale), ce qui n'est pas significatif.

Il est alors raisonnable de discrétiser l'ensemble des jeux de paramètres possibles afin d'éviter de tester plusieurs fois des situations très similaires. La discrétisation des valeurs est la suivante :

- S : 0 ou 1
- P : 20, 40, 60, 80, 100, 120
- C : 90%, 80%, 70%, 60%
- M : 0.05, 0.1, 0.2, 0.3, $\frac{1}{Taille\ de\ Chromosome}$

Les valeurs de population proposées sont comprises entre 20 et 120, tailles de population utilisées en pratique, comme montré dans un contexte différent dans [49]. La taille de population dépend fortement du problème, mais l'utilisation de ces valeurs usuelles permet de limiter l'étude.

Cette discrétisation limite l'étude à 240 jeux possibles, ce qui rend l'étude de tous les jeux envisageable suivant le temps de calcul accordé à l'algorithme génétique à paramétrer, et la taille d'échantillon de problèmes. On parlera d'évaluation complète des jeux de paramètres. Si l'échantillon choisi est élevé, on préférera utiliser le méta-AG proposé précedemment avec cette discrétisation, pour réduire le temps total de paramétrage.

Les résultats de l'évaluation complète des jeux de paramètres est présentée plus loin dans la partie expérimentale 3.7.3.

3.7 Expérimentations

L'objectif de cette partie expérimentale est d'évaluer et de valider les trois algorithmes pour un usage dans une architecture réseau.

Tout d'abord, les algorithmes permettent d'obtenir des solutions réalisables c'est-à-dire des arbres valides : continus, couvrant les destinations, sans cycles et dont les chemins sont uniques vers chaque nœud. D'une part, les processus heuristiques IS et RTF garantissent des solutions. D'autre part, l'encodage de l'algorithme génétique garantit des solutions fonctionnelles, c'est à dire des tracés continus et couvrant toutes les destinations, mais pouvant contenir de multiples chemins et des cycles. Néanmoins, les mécanismes de convergence permettent a priori d'effacer ce phénomène, et l'application de l'algorithme 3.5 le garantit.

On souhaite alors, à l'aide d'expérimentations, évaluer deux critères : le temps de calcul et la qualité de la solution, représentée par la fonction de coût définie en section 2.2.2.

3.7.1 Temps de calcul des heuristiques

Dans un premier temps on s'intéresse au temps de calcul nécessaire aux heuristiques. Bien qu'ayant une limite supérieure théorique de la complexité temporelle des algorithmes, il faut déterminer un temps de calcul en pratique. Le temps de calcul de l'algorithme génétique est discuté plus loin, étant donné qu'il dépend des critères d'arrêts définis.

Protocole expérimental

Étant donné le champ d'application de ces heuristiques, les expérimentations ont été menées sur un graphe représentant une partie de la topologie du réseau opérateur sur lequel porte ces travaux. Pour des raisons de secret industriel et de sécurité, la topologie exacte n'est pas publiée dans ces travaux.

Le graphe extrait de la topologie complète possède les caractéristiques suivantes : 314 nœuds, 728 liens unidirectionnels, et une architecture hiérarchique construite sur des boucles. La densité de ce graphe dirigé est $\rho_{TMS} = 0.0074$ défini dans l'équation 3.8. Le graphe est peu dense, mais la construction en boucles assure une redondance de chemin pour la plupart des nœuds.

$$\rho_{TMS} = \frac{|E|}{|V|(|V|-1)} = 0.0074 \tag{3.8}$$

Comme il existe différentes tailles de service au sein du réseau, cette expérience va étudier l'influence du nombre de nœuds destination dans le réseau.

On définit une instance caractéristique comme une source et un ensemble de destinations. Une source caractéristique est sélectionnée dans la partie "cœur de réseau" du graphe. Un ensemble de nœuds caractéristiques de 158 sites "destinations" du réseau a

été défini à partir de services existants. Les noeuds destination d'une instance sont sélectionnés parmi l'ensemble des destinations potentielles.

Les algorithmes exploitent l'algorithme CBF plutôt que l'algorithme LARAC, comme énoncé en 3.2.2.

Tous les calculs ont été menés sur une machine aux caractéristiques suivantes : 2.9 GHz CPU (2 core 4 Thread) et 16 Go de RAM. Les heuristiques sont implémentées en langage Python 2.7, en particulier à l'aide du module NetworkX [50].

Résultats

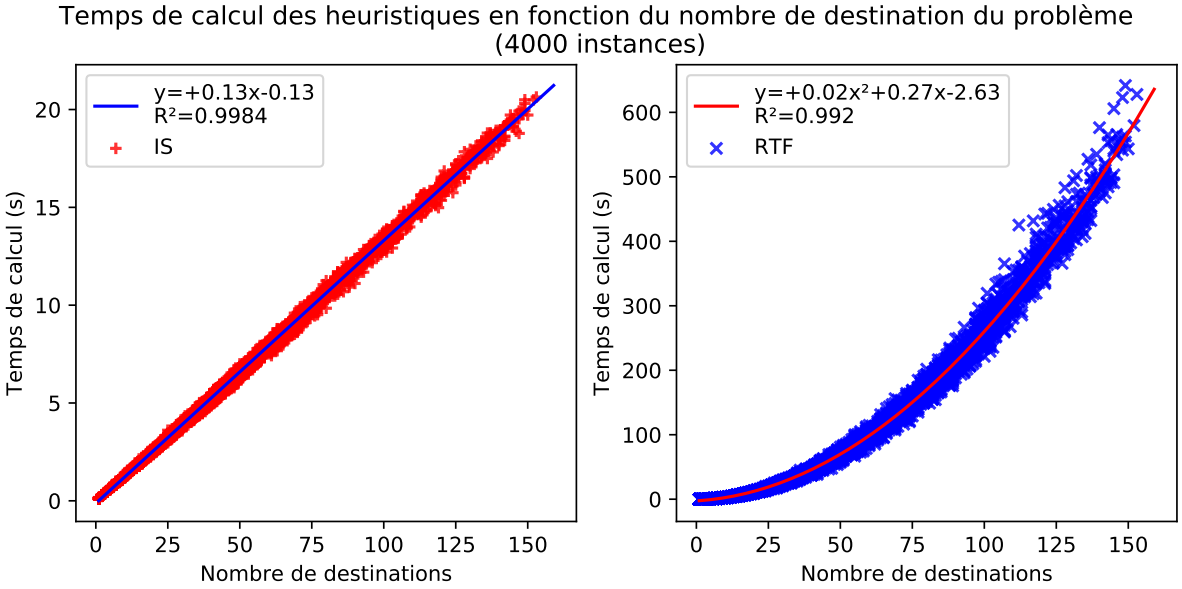


FIGURE 3.13 – Influence du nombre de destinations sur le temps de calcul des heuristiques

Concernant l'algorithme IS, le graphique de gauche de la figure 3.13 présente une croissance linéaire suivant le nombre de destinations $|T|$ du problème, avec un coefficient de détermination satisfaisant. La limite haute de complexité temporelle de l'algorithme évaluée plus haut comprenait un terme en $|T|^2$, qui est négligeable en pratique. Ce terme provient d'une boucle évaluée dans le pire des cas, qui correspond à une indépendance quasi totale de toutes les paires évaluées par IS. Dans le graphe utilisé pour ces expérimentations, la densité du graphe est trop peu importante pour que ce cas se présente.

Le graphique de droite de la figure 3.13 montre que le temps de calcul de l'algorithme RTF suit une loi quadratique en fonction du nombre de destinations, ce qui signifie que la limite haute en complexité temporelle présentée plus haut est plus proche de la limite haute exacte.

D'après les équations des courbes de régression, au-delà de 8 destinations, l'algorithme IS a un temps de calcul bien inférieur à l'algorithme RTF.

Il pourrait être intéressant d'étudier l'influence de la taille et de la densité du graphe sur ces heuristiques, ainsi que de leurs effets combinés. Cela relève du domaine de l'analyse de sensibilité globale des systèmes, et pourrait être abordé dans de futurs travaux.

3.7.2 Comparaison à des pseudo-optimums

Les expérimentations précédentes ont montré que l'algorithme IS était globalement plus intéressant sur le critère du temps de calcul. Cette partie cherche à déterminer si l'un des algorithmes fournit des solutions de meilleure qualité. Étant donné qu'il est impossible de connaître la solution optimale, l'évaluation de la qualité des solutions ne peut être que relative.

Sur le jeu de problème de l'expérimentation précédente, la comparaison des fonctions d'évaluation définie par la formule 2.5 (p. 26) donne le graphique présenté en figure 3.14. Il n'apparaît pas de supériorité majeure d'une heuristique par rapport à l'autre. En effet, pour un nombre de destinations similaire, on observe des cas ou chacun des algorithmes est significativement meilleur que l'autre. C'est-à-dire qu'autour d'une même abscisse, les points semblent équitablement répartis en ordonnée.

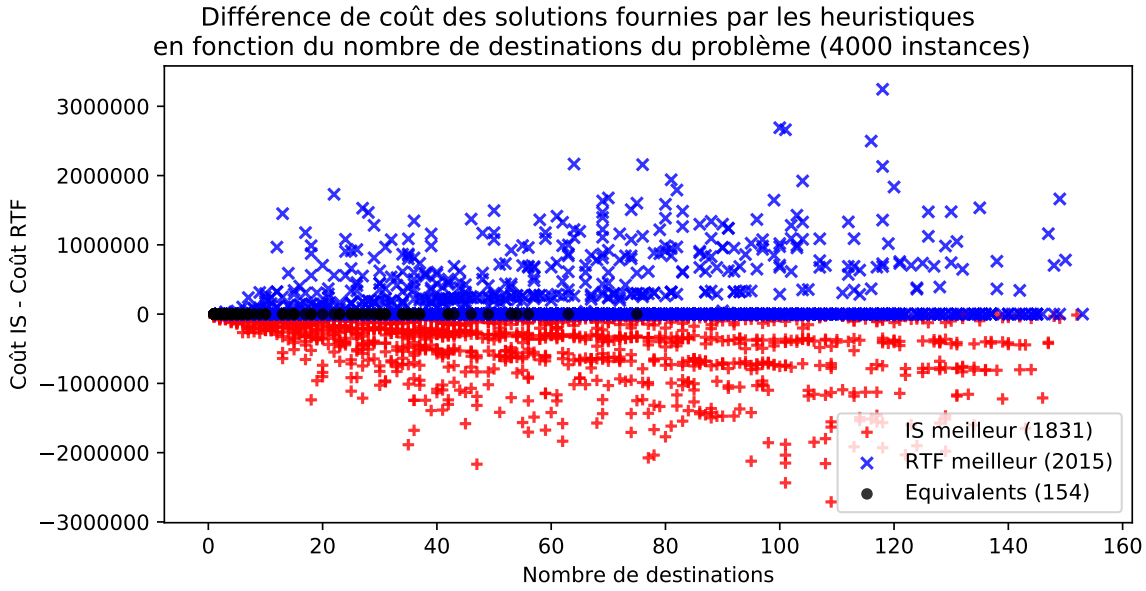


FIGURE 3.14 – Différence de qualité de solution fournie par les heuristiques suivant le nombre de destinations

On cherche alors à savoir si les heuristiques fournissent au moins des solutions cohérentes et exploitables en les comparant à des tracés existants. Pour ce faire, il est nécessaire d'établir un protocole de test.

Protocole

Certains services existants sur le réseau opérateur fonctionnent déjà avec deux arbres de diffusion redondants, dont les tracés ont été faits "à la main" et seront notés "MAN". On peut considérer ces tracés comme des pseudo-optimums, car des méthodes humaines n'ont pas trouvé de meilleure solution. Seuls 43 services ont été sélectionnés, dont le nombre de destinations est peu élevé. En effet, à cause de la complexité du problème, aucun tracé n'a été fait pour des cas plus importants. De plus, la source de ces services n'est pas nécessairement dans le cœur de réseau, étant donné qu'il s'agit de services dits régionaux. Pour chacun de ces tracés, une solution a été calculée par les deux heuristiques IS et RTF, et la fonction de coût comparée.

Cette série d'expérimentation utilise une topologie plus étendue que le cas précédent, mais toujours extraite de la topologie réelle : 928 nœuds, 1960 liens, $\rho = 0.0023$. Les autres conditions expérimentales sont les mêmes que précédemment.

Résulats

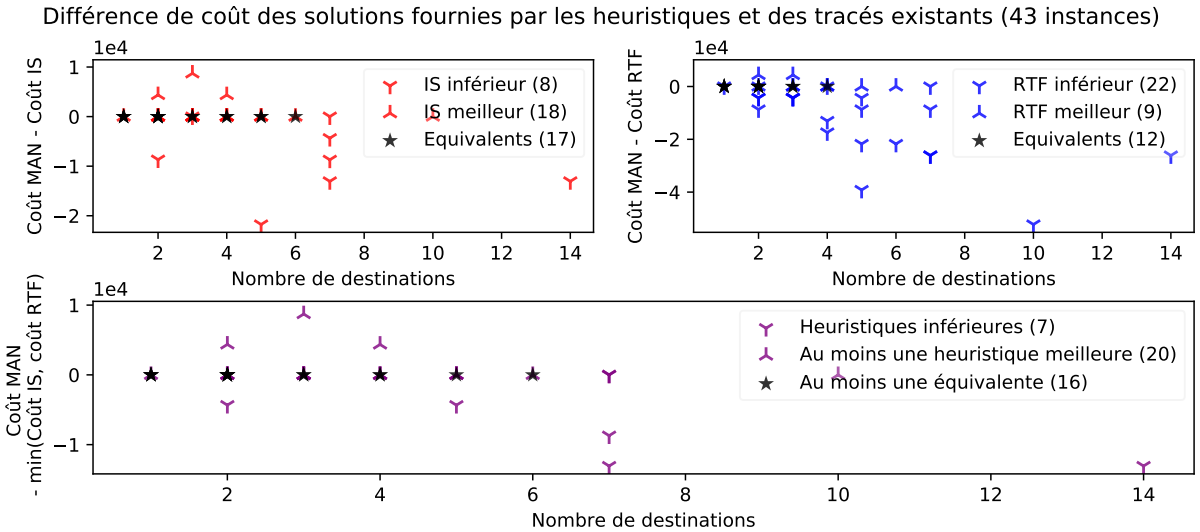


FIGURE 3.15 – Qualité de solution fournie par les heuristiques par rapport à des tracés existants

La figure 3.15 présente la comparaison des solutions fournies par les heuristiques IS et RTF et des tracés MAN. Malgré le nombre d'échantillons réduits, on remarque que dans la majorité des cas, les heuristiques fournissent une solution équivalente ou meilleure que le tracé MAN. Il arrive néanmoins qu'un tracé à la main soit plus intéressant que le résultat des heuristiques, ce qui illustre le fait que l'optimum n'est pas atteint par les heuristiques.

3.7.3 Algorithme génétique appliqué

Les expérimentations précédentes montrent que les heuristiques fournissent généralement des solutions satisfaisantes en un temps raisonnable, mais pas nécessairement optimales. En théorie, l'algorithme génétique donnera des solutions de meilleure qualité, si le temps accordé au calcul est suffisant.

Protocole

Parmi les 43 cas sélectionnés précédemment, 5 cas sont retenus pour cette expérience. Ces 5 cas sont représentatifs des différentes caractéristiques de topologie associées à chaque boucle du réseau.

La méthode d'évaluation des 240 jeux de paramètres proposée plus haut sera appliquée à ces 5 cas. Cela permettra d'une part, de définir les meilleurs paramètres pour la classe de problème étudiée, et d'autre part, de comparer les résultats de l'AG aux résultats des heuristiques IS et RTF. En particulier, si le paramètre de Seeding est à 1, les solutions IS et RTF sont inclues dans la population initiale.

Le temps de calcul de l'AG est limité à 5 minutes, afin de permettre une convergence suffisante, tout en conservant un temps de calcul raisonnable en pratique.

L'AG est implémenté sur MATLAB à l'aide de la "Global Optimisation Toolbox" [51].

Résultats

Les résultats sont présentés en figures 3.16 et 3.17. Les données sont composées de 4 paramètres et de la meilleure valeur de fonction d'évaluation parmi les individus trouvés. Il est proposé ici de représenter ces données en 5 dimensions sous forme d'arbre circulaire :

- le nœud central représente la valeur du paramètre Seeding S ;
- le premier cercle représente les valeurs de la taille de population P ;
- le deuxième cercle le taux de croisement C ;
- le troisième cercle la probabilité de mutation M ;
- le cercle extérieur représente les valeurs suivant une échelle de couleur.

On peut lire cette représentation dans les deux sens :

- pour une valeur donnée, en remontant l'arbre, on retrouve le jeu de paramètre donnant un tel résultat ;
- en partant du centre, on sélectionne les paramètres en descendant l'arbre. L'extrémité de la branche représente la valeur obtenue par ce jeu.

Pour des raisons de lisibilité, les données ont été séparées en deux arbres : la figure 3.16 dans laquelle le paramètre S vaut 0, et la figure 3.17 dans laquelle S vaut 1.

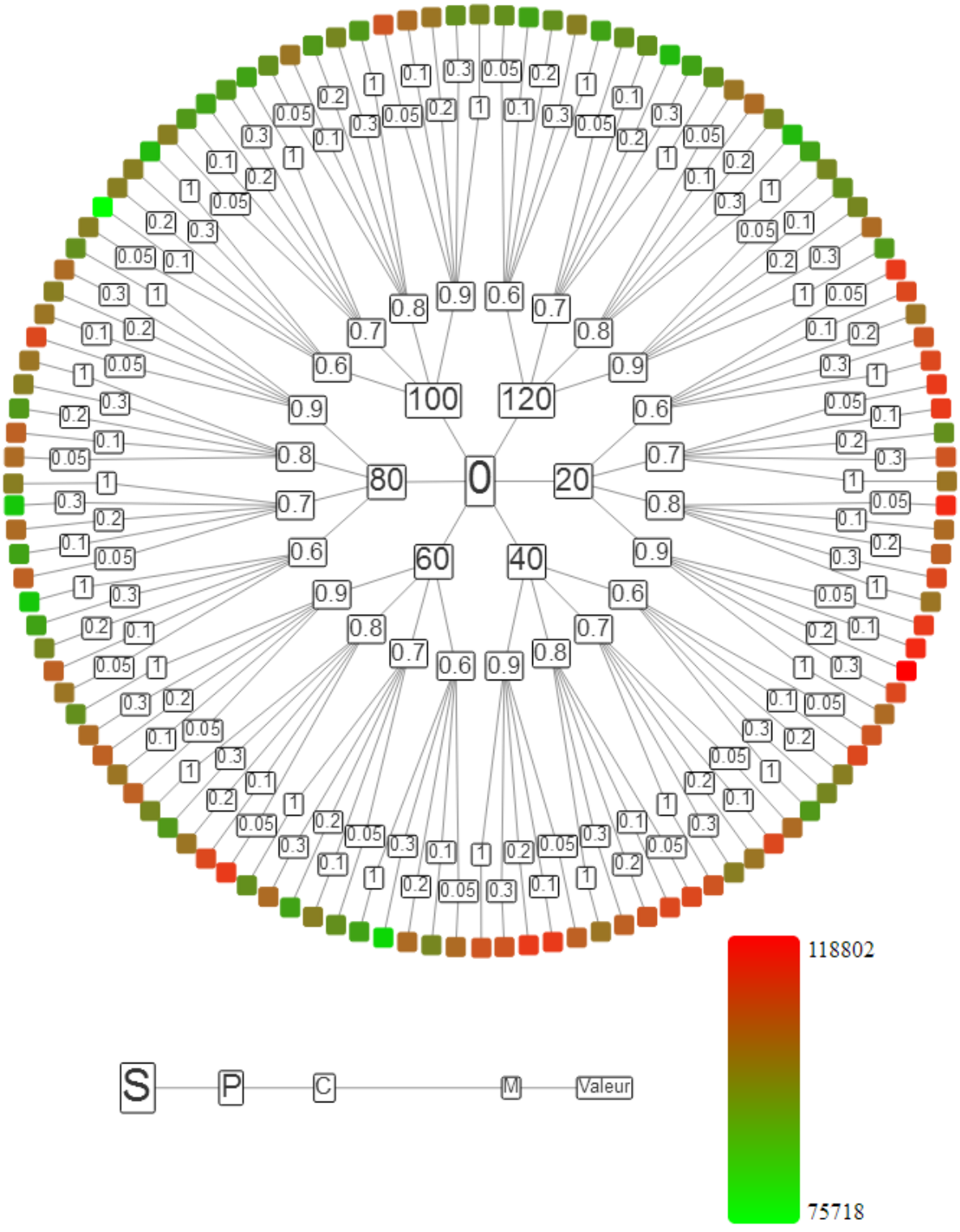


FIGURE 3.16 – Moyenne des fonctions d'évaluations des meilleures solutions de l'AG pour $S = 0$ suivant les jeux de paramètres

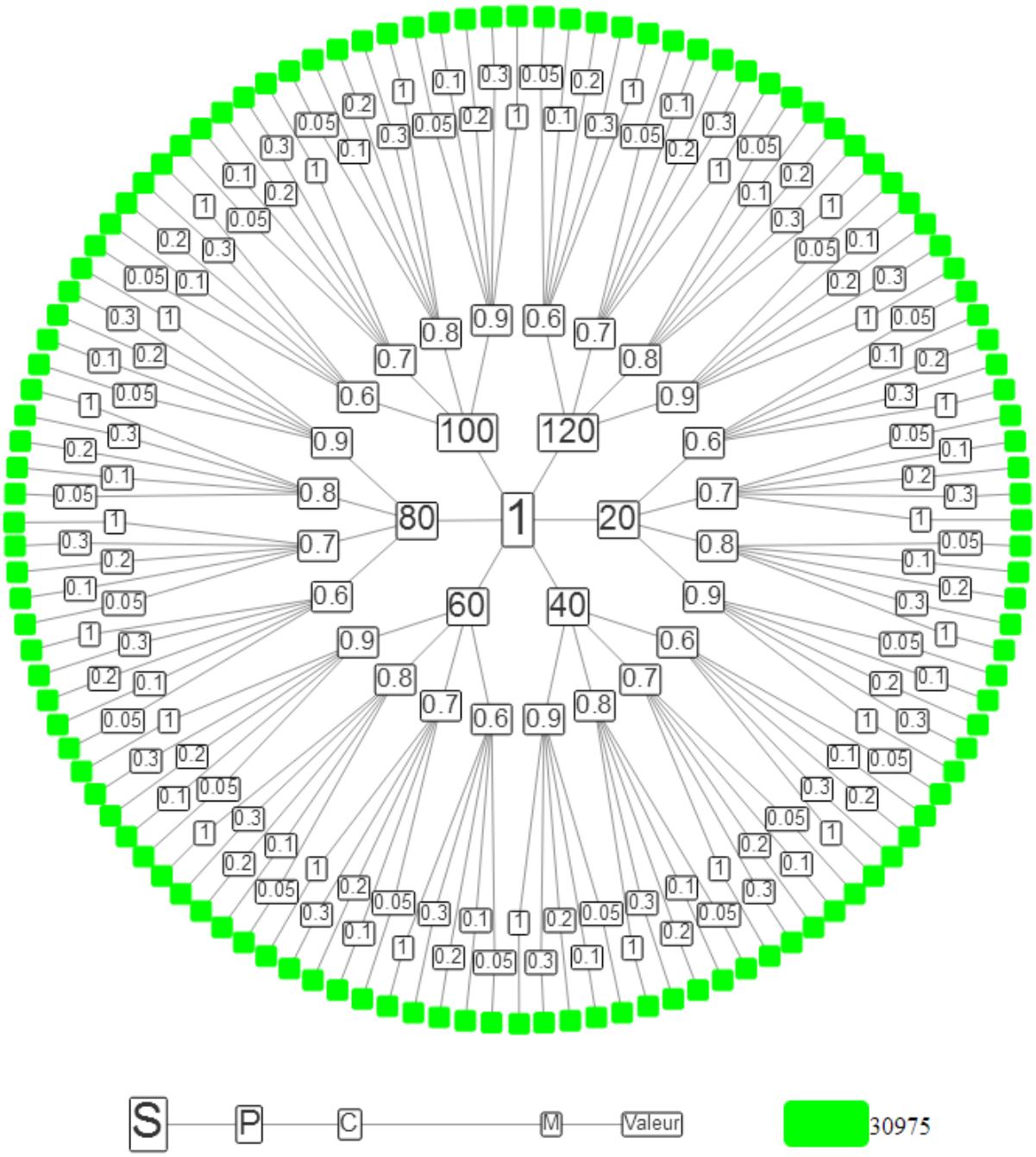


FIGURE 3.17 – Moyenne des fonctions d'évaluations des meilleures solutions de l'AG pour $S = 1$ suivant les jeux de paramètres

La comparaison des figures 3.16 et 3.17 illustre que le paramètre S est le plus influent de tous. En effet, quels que soient les autres paramètres, les jeux dans lesquels S=1 fournissent systématiquement une meilleure valeur moyenne d'évaluation. Elle correspond aux solutions fournies par les heuristiques IS et RTF. En d'autres termes, les solutions trouvées par l'AG sont au mieux équivalentes aux solutions des heuristiques, car même en injectant ces solutions dans la population initiale, l'AG ne parvient pas à obtenir un résultat de meilleure qualité.

Plusieurs interprétations de ce résultat sont possibles :
- les heuristiques IS et RTF fournissent une solution optimale : cela a déjà été invalidé par les expériences précédentes ;
- le protocole expérimental comporte un biais : la limitation en nombre de jeux ne semble pas aberrante, mais le temps limité de calcul peut être un facteur. Néanmoins, lors de l'expérimentation, le temps de convergence était systématiquement sous cette limite ;
- l'AG est très peu performant : comme évoqué, l'AG converge relativement rapidement. On peut alors penser que la qualité des solutions est dégradée par une convergence prématurée. Cela peut être dû à la non-continuité de la fonction d'évaluation, au trop grand nombre de solutions, ou encore à un encodage mal adapté.

Bien que l'algorithme génétique garantit la validité des solutions, et la présence de l'optimum dans l'espace des possibles, le résultat obtenu n'est pas satisfaisant. De plus, la phase préliminaire de remplissage des tables de routage est en pratique très longue.

3.8 Conclusion

Trois solutions algorithmiques au problème DMDT ont été proposées et évaluées dans ce chapitre. Les heuristiques IS et RTF se fondent sur des principes intuitifs et rapides, mais présentent des performances variables.

Les résultats expérimentaux aboutissent aux recommandations suivantes :
- si le problème présente plus de 8 destinations : préférer IS pour profiter de sa rapidité ;
- si un délai de calcul court est requis : préférer IS pour sa rapidité ;
- si délai de calcul plus long est possible : appliquer IS et RTF afin de conserver le meilleur résultat.

Les résultats obtenus par les heuristiques sont d'une qualité toujours suffisante pour l'exploitation sur le réseau réel par le protocole Seamless Multicast. Néanmoins, l'amélioration de leurs performances peut être étudiée dans un cadre plus général.

Un algorithme génétique a également été proposé, mais l'expérimentation montre que malgré une conception cohérente et garantissant des solutions valides, la qualité des solutions obtenues est inférieure ou équivalente aux deux heuristiques proposées.

Le chapitre 4 décrit une architecture réseau implémentant le principe du Seamless Multicast, exploitant les algorithmes présentés dans ce chapitre.

Chapitre 4

Implémentation Software-Defined Networking

Sommaire

4.1 Le Software-Defined Networking 80
4.1.1 Présentation 80
4.1.2 Le protocole OpenFlow 82
4.1.3 Etat de l'art relatif au transport audiovisuel en SDN 84
4.2 Architecture réseau proposée 85
4.3 Description technique de l'implémentation 86
4.4 Protocole expérimental 89
4.4.1 Réseau d'expérimentation 89
4.4.2 Paramètres du contrôleur 89
4.4.3 Règles de commutation 90
4.4.4 Émulation de récepteurs Seamless 91
4.4.5 Déroulement du protocole 91
4.5 Résultats 92
4.5.1 Panne impactant un seul arbre, indépendance stricte 93
4.5.2 Panne impactant un seul arbre, indépendance maximale 96
4.5.3 Panne impactant les deux arbres 98
4.5.4 Création d'une nouvelle liaison 100
4.6 Conclusion 102

Afin d'implémenter les concepts de l'architecture Seamless Multicast, il est quasiment obligatoire d'utiliser les technologies offertes par le Software-Defined Networking (SDN). L'objectif de ce chapitre est de définir une implémentation complète des concepts et algorithmes présentés dans les chapitres 2 et 3.

Ce chapitre présente le paradigme réseau du SDN, qu'on définit généralement comme une architecture centralisée, à la différence des réseaux traditionnels IP. Un état de l'art de la diffusion exploitant les technologies associées au SDN est abordé, et une architecture Seamless Multicast suivant les principes du SDN est proposée.

Ce chapitre présente également une implémentation de l'architecture proposée dans le chapitre 2 utilisant la technologie OpenFlow qui est aujourd'hui la technologie SDN dominante. Cette technologie est présentée pour introduire les éléments d'architecture. Le contrôleur ainsi construit est déployé pour évaluer la proposition sur des cas réels.

4.1 Le Software-Defined Networking

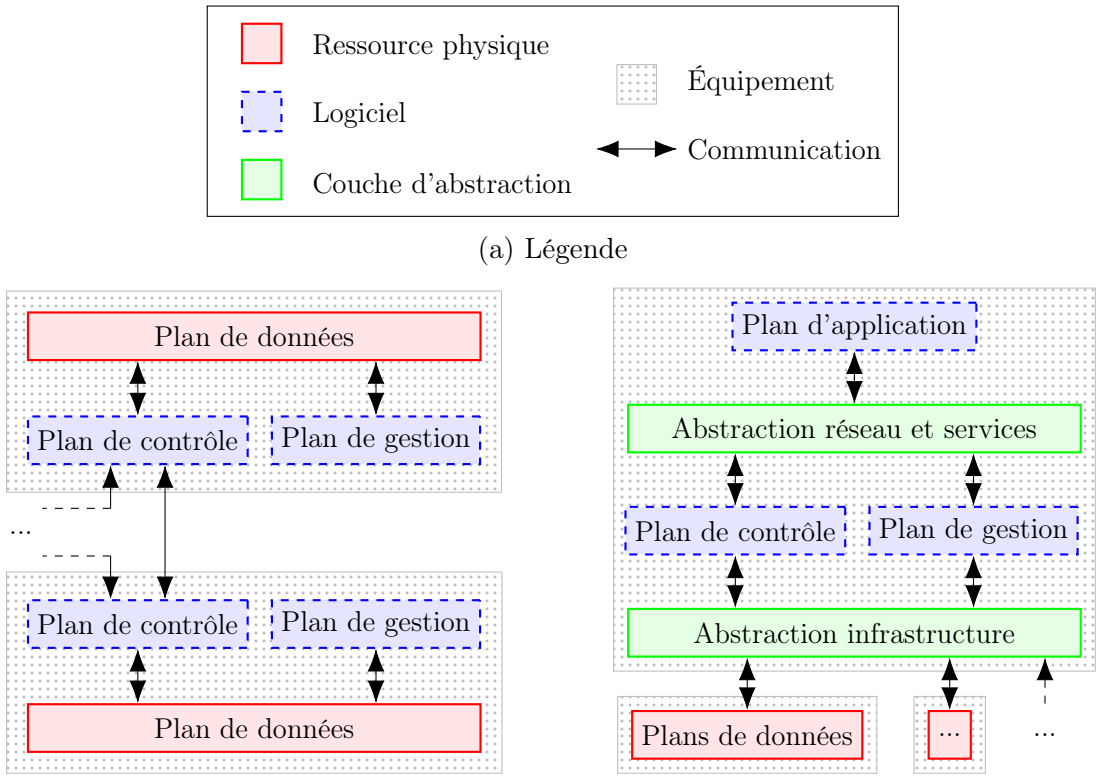
4.1.1 Présentation

Au sein d'un réseau, on distingue généralement des "plans". Ce sont des éléments logiques responsables de différentes fonctions. On présentera ici trois principaux plans :

- le plan de contrôle : il s'agit de l'élément décisionnel d'une architecture réseau. Une connaissance de la topologie lui est nécessaire pour prendre des décisions. Dans le cas d'un routeur, le plan de contrôle échange des informations avec les autres éléments du réseau et définit des règles locales de traitement d'un paquet ;
- le plan de données : aussi appelé plan de commutation, c'est la partie opérationnelle d'un réseau, qui transporte les paquets de données. Il applique les décisions du plan de contrôle. Dans le cas d'un routeur, le plan de données commute les paquets entre les différents ports suivant les règles définies par le plan de contrôle ;
- le plan de gestion : ce plan contient tous les éléments se rapportant à la gestion d'un réseau, notamment la configuration, la supervision et la métrologie.

Dans une architecture traditionnelle réseau fondée sur le protocole IP, le plan de contrôle est distribué dans les équipements. C'est la communication entre les équipements, qu'on appelle protocole de routage, qui permet la cohérence du plan de contrôle. Ce protocole de routage est soumis à un temps de convergence.

Le Software-Defined Networking (SDN) est un paradigme d'architecture réseau dans lequel le plan de contrôle est programmable et physiquement séparé du plan de données. Autrement dit, la partie décisionnelle du réseau est gérée par une entité extérieure au réseau programmable à travers une couche d'abstraction, séparée de la partie opérationnelle des équipements formant le réseau. On parle de "contrôleur" pour désigner l'entité correspondant au plan de contrôle. Les nœuds du réseau reliés à ce contrôleur sont appelés "équipements de commutation", et portent le plan de données. La figure 4.1 compare les architectures traditionnelles et SDN, résumant le contenu de la RFC 7426 [52].



(a) Légende

(b) Architecture traditionnelle distribuée

(c) Architecture en couches SDN

FIGURE 4.1 – Comparaison des architectures traditionnelles et SDN

Les technologies SDN permettent de s'affranchir des spécificités constructeurs grâce à des fonctions d'abstraction. Chaque opérateur peut alors implémenter des protocoles spécifiques indépendamment des stratégies des équipementiers. De plus, la programmabilité du réseau augmente l'agilité des services, et permet de traiter des problèmes de calcul différemment, de manière centralisée, ou sur des équipements dédiés plus efficaces. Les architectures SDN ont notamment été étudiées pour l'architecture d'Internet et la connexion des IXPs [53, 54]. Des approches existent également pour les réseaux électriques intelligents "Smart Grid" [55, 56], pour des applications VoIP [57], ou encore des environnements de type Data Center [58].

La centralisation du plan de gestion peut permettre des analyses de trafic plus complètes telle que [59] sans nécessiter d'architecture de supervision supplémentaire. De même, la vue complète du réseau permet d'appliquer de nouvelles techniques, par exemple pour la sécurité par identification d'attaques proposée par [60].

De nouvelles problématiques apparaissent dans ces architectures. En effet, un contrôleur centralisé peut être considéré comme un point unique de panne du réseau (Single Point of Failure - SPOF), compromettant la sûreté et la sécurité du réseau. Il existe notamment des modèles d'architecture où les contrôleurs sont distribués [61].

D'autre part, étant donné que le réseau est basé sur du logiciel, la notion de "bug" doit être prise en compte, avec des processus de développement appropriés.

4.1.2 Le protocole OpenFlow

OpenFlow est une technologie SDN [62] soutenue par l'Open Networking Foundation (ONF), un consortium d'entreprises dont l'objectif est de travailler sur l'évolution de standards ouverts pour le réseau. On y trouve notamment les grands acteurs du marché : Cisco, Nokia, Ciena, Broadcom, Google...

Le protocole OpenFlow est une interface de programmation du plan de données, c'est-à-dire la liaison entre l'intelligence d'un contrôleur et les équipements réseau. Il ne s'agit pas d'une définition du fonctionnement interne des commutateurs réseau.

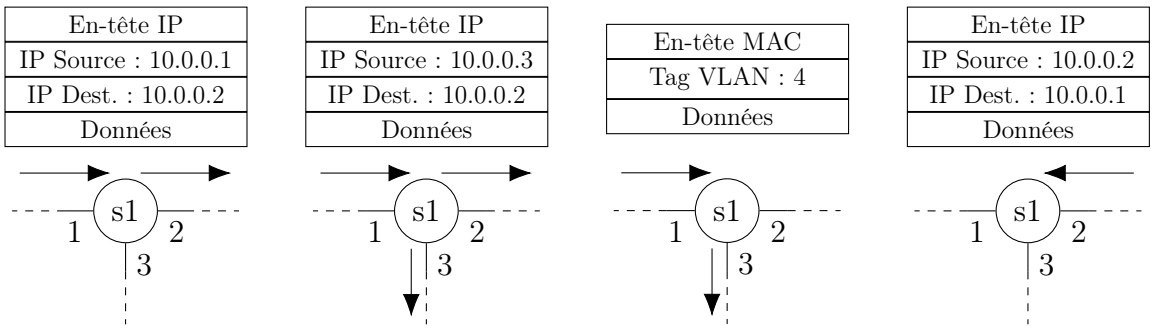
OpenFlow est basé sur un concept de "Match/Action" : des règles de correspondance de paquet "Match" déclenchent diverses actions sur un commutateur. Lorsqu'un paquet arrive sur un commutateur, celui-ci vérifie l'ensemble des règles définies et applique l'action de la règle correspondant au paquet reçu.

Par analogie, la table de commutation d'un commutateur Ethernet niveau 2 est un ensemble de règles de correspondance sur adresse MAC dont les actions sont de transmettre le paquet aux ports associés.

Les règles OpenFlow permettent de définir des correspondances sur de nombreux paramètres : adresses MAC, adresses IP, tag VLAN, label MPLS, port d'entrée... Elles permettent également l'application de diverses actions et combinaison d'actions : commutation vers un ou plusieurs ports, commutation vers le contrôleur, modification d'adresse... Quelques exemples de règles sont présentés dans la figure 4.2.

Match	**Actions**
IP Source = 10.0.0.1 & IP Destination = 10.0.0.2	Commuter vers port 2
IP Source = 10.0.0.3	Commuter vers port 2 et 3
Tag VLAN 4	Commuter vers port 3
Tous les autres cas	Abandonner le paquet

(a) Règles installées sur le commutateur

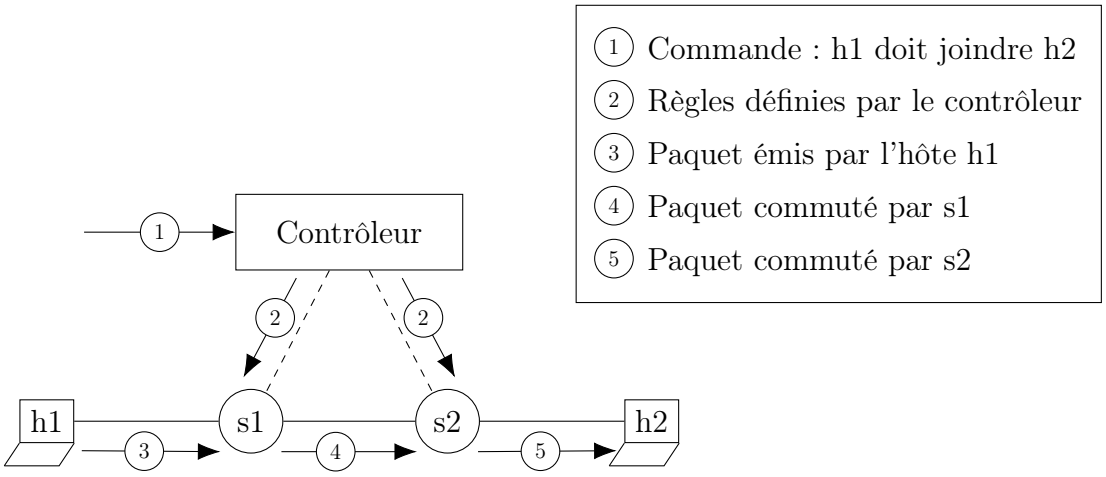


(b) Diffusion simple (c) Multidiffusion (d) VLAN (e) Abandon de paquet

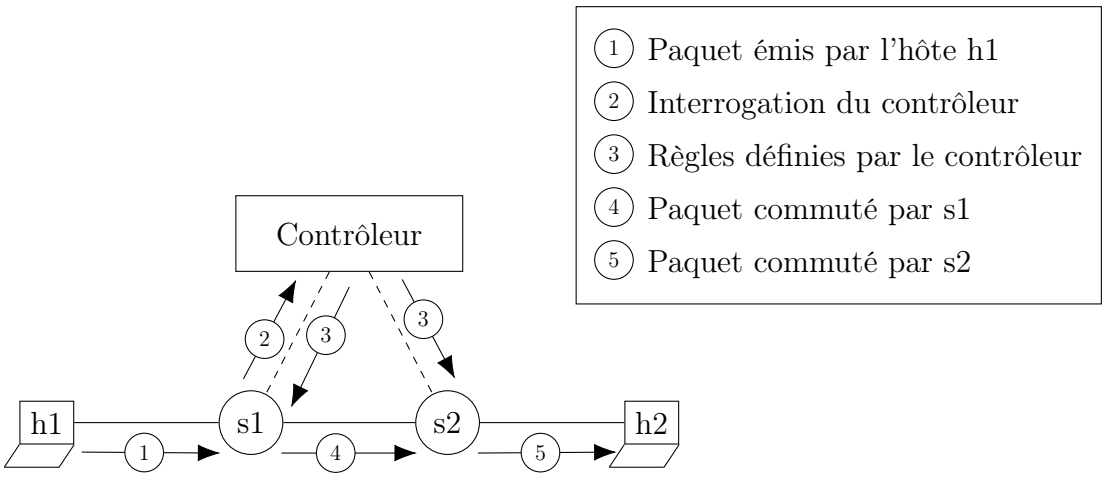
FIGURE 4.2 – Exemples d'applications de règles OpenFlow

On appelle "Flow" le triplet "Match", actions et paramètres. Les paramètres peuvent contenir des notions de priorité, de durée ou des statistiques. Le contrôleur définit les Flows à appliquer sur les commutateurs du réseau suivant le plan de contrôle, et les communique au réseau à travers des paquets protocolaires OpenFlow. Chaque commutateur, à la réception de tels paquets, modifie son ensemble de Flows suivant les ordres reçus.

La table des Flows peut être remplie proactivement par le contrôleur, ou réactivement à un évènement notifié par un commutateur. La figure 4.3 illustre ces deux modes de remplissage des tables des Flows dans le cas d'une transmission point-à-point.



(a) Remplissage proactif de la table des Flows



(b) Remplissage réactif de la table des Flows

FIGURE 4.3 – Modes de remplissage de la table des Flows dans le protocole OpenFlow

Les actions appliquées par le commutateur et leur implémentation dépendent des équipements. Seuls les Flows et la communication avec le contrôleur sont spécifiées dans le protocole OpenFlow.

Il est notable que depuis l'émergence et l'utilisation très répandue d'OpenFlow, un amalgame existe avec la définition SDN. Néanmoins de telles architectures étaient déjà envisagées dans le cadre des réseaux ATM à commutation de cellules [63], où les équipements recevaient des tables de commutation d'une entité extérieure.

4.1.3 Etat de l'art relatif au transport audiovisuel en SDN

Des cas de transport point-à-point résilients sont abordés dans la littérature [64, 65, 66]. On y distingue des approches proactives et réactives.

Concernant les cas de multidiffusion, [67, 68] offrent un aperçu de la littérature sur le sujet. On s'intéresse en particulier aux architectures de multidiffusion visant à améliorer la disponibilité.

Dans [69], l'architecture déploie proactivement des chemins de secours dans le réseau et les active en cas de panne. Cette solution améliore la disponibilité par rapport à une architecture traditionnelle, mais exploite de multiples diffusions simples, ce qui est plus couteux qu'une multidiffusion. De plus, il existe toujours un temps de détection de panne durant lequel une ou plusieurs destinations sont impactées.

De manière similaire, [70] propose une architecture SDN proactive exploitant des arbres de multidiffusion pré-calculés en cache pour assurer une forme de reroutage rapide.

[71] propose une sécurisation par sous-arbres comme évoqué en 2.3.1, en exploitant des chemins de secours pré-calculés au sein des arbres.

L'approche présentée dans [72] consiste à répartir les destinations sur plusieurs arbres de multidiffusion, et de préparer des inter-connexions de secours entre les arbres. Ainsi en cas de panne, les parties d'un arbre qui sont déconnectées de la source se raccordent rapidement à un autre arbre.

Ces travaux de thèse proposent une implémentation des concepts et algorithmes présentés aux chapitres 2 et 3, dont l'approche diffère des solutions disponibles dans la littérature, car elle exploite à la fois redondance et routage dynamique.

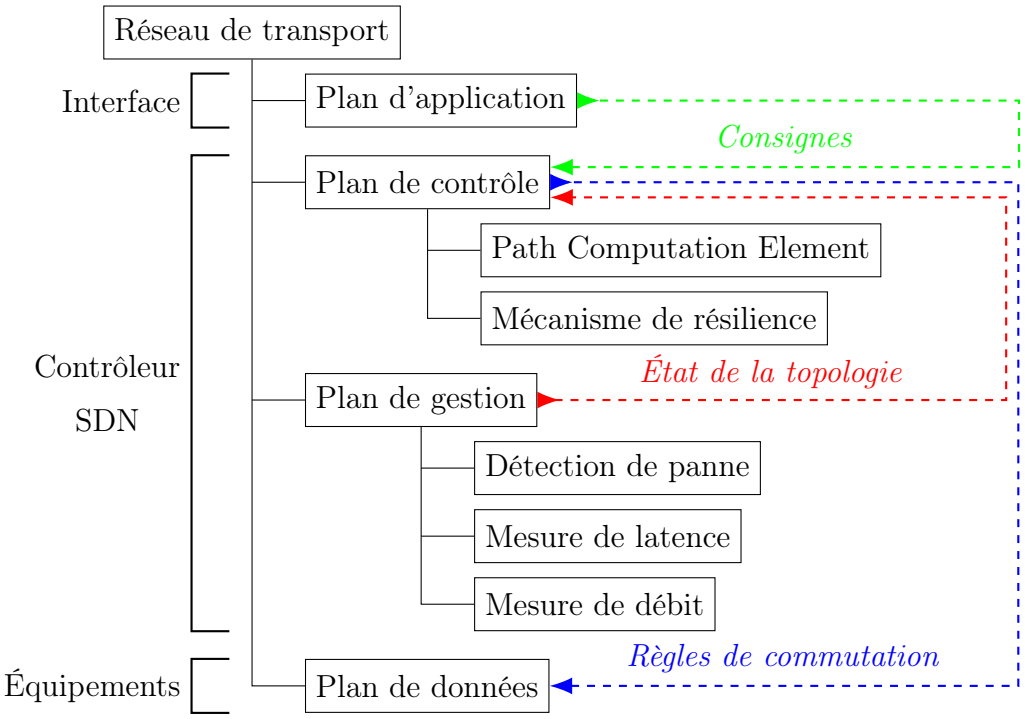


FIGURE 4.4 – Architecture réseau

4.2 Architecture réseau proposée

Le principe de l'architecture Seamless Multicast est le déploiement d'une paire résiliente d'arbres redondants. Étant donné la complexité du problème DMDT qui doit être résolu pour obtenir une paire d'arbres respectant les contraintes, l'architecture proposée repose sur un PCE (Path Computation Element) [73]. Un PCE est une entité capable de calculer des chemins dans un réseau à partir du graphe correspondant et d'un ensemble de contraintes. Ce PCE peut être porté par un nœud du réseau, ou encore par une entité extérieure. Dans le cadre de l'architecture SDN proposée, le PCE est localisé dans le contrôleur, bien que les architectures fondées sur des PCE ne soient pas restreintes à l'utilisation d'un superviseur centralisé. Cette centralisation permet de s'affranchir du défaut de visibilité d'un nœud réseau. En effet, les protocoles permettant à un nœud de connaître la topologie peuvent être limités, et ne comportent pas de données extérieures nécessaires à des calculs contraints. [74] offre une vue d'ensemble de la littérature sur les architectures possibles fondées sur un PCE.

Pour définir une architecture SDN correspondant aux besoins du Seamless Multicast, il est nécessaire de décrire les éléments constituant chacun des plans. La figure 4.4 illustre la décomposition de l'architecture suivant les différents plans du réseau.

Tout d'abord, le plan d'application constitue l'interface avec l'opérateur du réseau. C'est dans cette interface que peuvent être définis les services, c'est-à-dire les ensembles source et destinations associées entre lesquelles doit être transporté le flux audiovisuel. Ce sont ces services qui définissent des consignes pour le plan de contrôle.

Le plan de contrôle est constitué de deux éléments. Le premier est le PCE, c'est-à-dire le composant permettant le calcul d'une paire d'arbres aux propriétés décrites dans le chapitre 2. Les algorithmes du chapitre 3 constituent ce PCE. Le second composant du plan de contrôle est le mécanisme de résilience présenté au chapitre 2. Ces deux composants doivent avoir connaissance de la topologie et de ses caractéristiques. C'est le rôle du plan de gestion de lui transmettre ces informations.

Le rôle du plan de gestion est décomposé en différentes fonctions. Pour chaque élément du réseau, nœud ou lien, le plan de gestion s'assure de l'état opérationnel et collecte les informations de latence et de débit. La fonction de détection de panne permet de signaler un incident au plan de contrôle, afin que le mécanisme de résilience soit activé. Les fonctions de collecte de données servent à fournir des éléments de pondération au Path Computation Element du plan de contrôle.

Enfin, le plan de données reçoit des règles de commutation de paquet du plan de contrôle et les applique.

Le plan de données est formé par l'ensemble des équipements du réseau. Les plans d'application, de gestion et de contrôle forment le contrôleur du réseau. En termes d'implémentation, ces plans peuvent-être physiquement séparés dans des machines distinctes, mais logiquement restent partie du contrôleur.

4.3 Description technique de l'implémentation

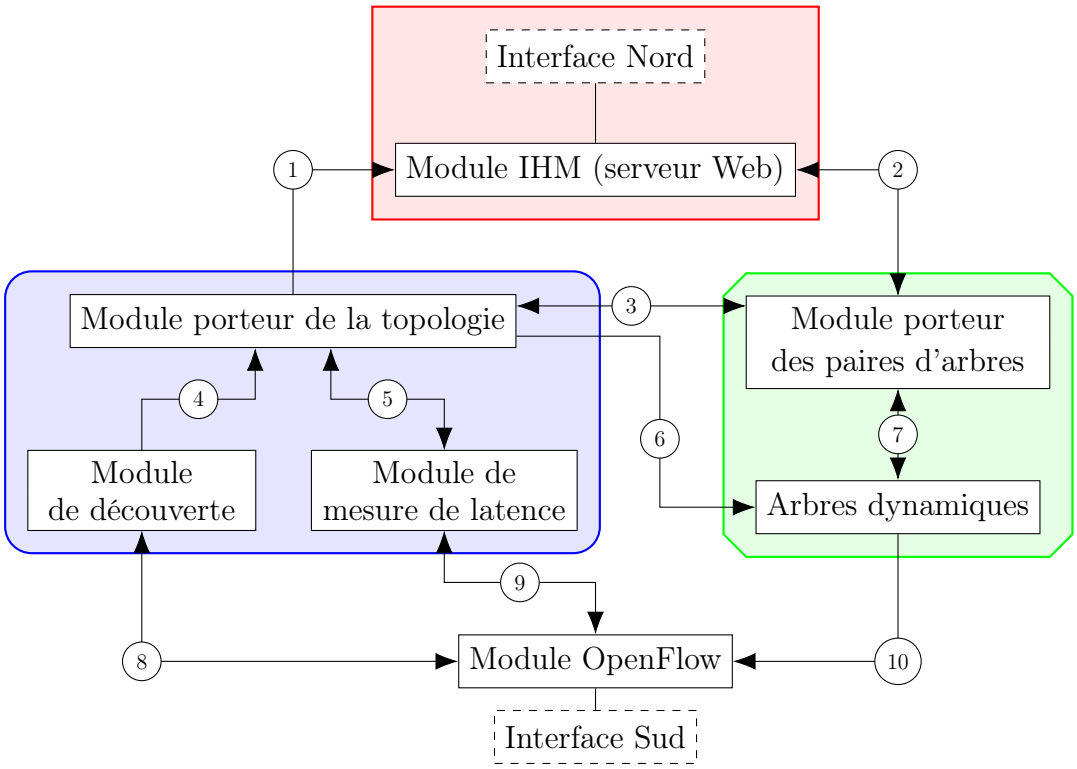
Dans le cadre de ces travaux, le contrôleur POX [75] a été choisi comme base pour développer le contrôleur correspondant à l'architecture proposée. L'architecture logicielle du contrôleur présentée en figure 4.5 est divisée en modules.

Le plan d'application est l'interface avec l'opérateur du réseau dite "Interface Nord". Il comporte un module offrant un serveur Web pour interface utilisateur. L'interface en question permet de visualiser le réseau géré par le contrôleur et les services déployés. Elle permet également à l'opérateur de commander des services en précisant :

— un nœud source ;
— un ensemble de nœuds de destination ;
— une adresse IP associée au service ;
— une contrainte de longueur.

Le plan de gestion comprend un module gardant en mémoire la topologie et ses caractéristiques. Deux modules le tiennent à jour : un module de découverte du réseau, et un module de mesure de latence.

Le module de découverte du réseau s'assure de la découverte de nouveaux nœuds et liens. Basé sur le module "discovery" du contrôleur POX original, ce module assure également la surveillance de l'état des liens. Son principe général de fonctionnement est le suivant : lorsqu'une connexion OpenFlow à un nœud est ouverte ou fermée, il signale un changement de topologie. Tous les nœuds reçoivent une règle du contrôleur indiquant que tout paquet de type "découverte" doit être remonté au contrôleur. Sur ordre du contrôleur,



Plan d'application
Plan de gestion
Plan de contrôle
1 Mise-à-jour de la topologie sur requête
2 Commande d'arbres, retour pour affichage
3 Modification de la topologie et affectation de la bande passante
4 Évènements sur la topologie
5 Mise en place des tests de latence et stockage des résultats
6 Demande de topologie à jour pour calcul
7 Lancement de calcul et déploiement initial ou en cas de changement
8 Paquets de "découverte" (émission et réception)
9 Paquets de test de latence (émission et réception)
10 Déploiement des Flows

FIGURE 4.5 – Architecture de l'implémentation du contrôleur SDN

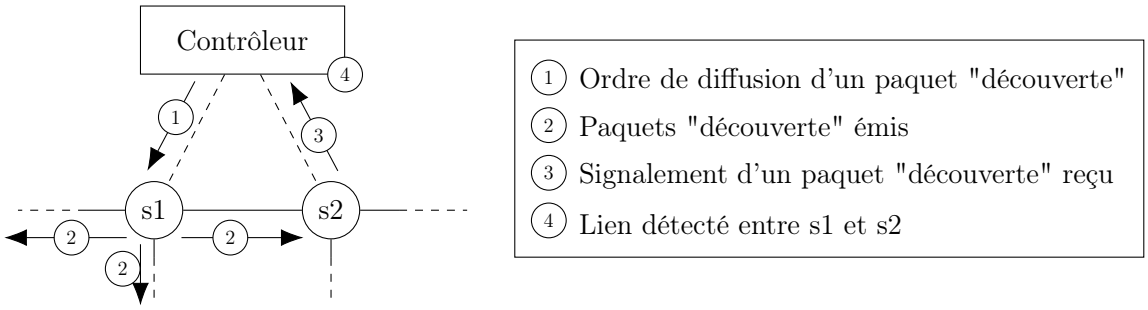


FIGURE 4.6 – Principe de fonctionnement du module de découverte de topologie

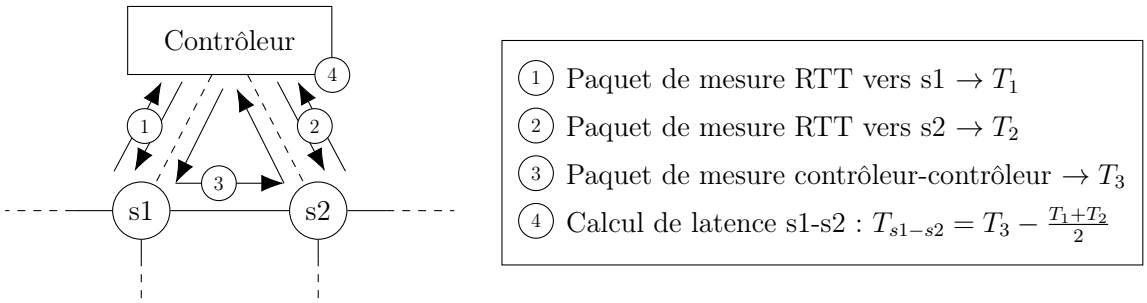


FIGURE 4.7 – Principe de fonctionnement du module de mesure de latence

un tel paquet est émis régulièrement, par chaque nœud, sur tous ses ports. Tout nœud recevant ce type de paquet le signale au contrôleur conformément à la règle établie, ce qui permet au module d'interpréter la présence d'une liaison. Si une liaison est désactivée, plusieurs paquets consécutifs manquant indiqueront au module la chute de la liaison. La figure 4.6 illustre le fonctionnement de ce module.

Le module de latence assure la mesure de latence unidirectionnelle pour tous les liens en mémoire. Son fonctionnement est fondé sur les travaux de [76]. Le principe de cette mesure est illustré dans la figure 4.7. Trois paquets sont émis depuis le contrôleur pour mesurer la latence entre le contrôleur et les deux commutateurs , et la latence du contrôleur au contrôleur via le lien entre les deux commutateurs. Ainsi, on obtient une approximation de la latence du lien. En effet, la mesure nécessite trois paquets qui seront soumis à une gigue, et la symétrie des liens vers le contrôleur est une hypothèse requise.

Le plan de contrôle est composé de deux modules complémentaires permettant le calcul et la gestion des arbres déployés sur le réseau. Le mécanisme de déploiement dynamique présenté au chapitre 2 et le calcul par l'un des algorithmes présenté dans le chapitre 3 sont implémentés dans ces modules.

Le plan d'application interagit avec le plan de gestion pour afficher à l'opérateur des informations sur la topologie. L'opérateur peut alors commander un service à travers une interface graphique, décrite en Annexe B. Cette commande sera transmise au plan de contrôle par le plan d'application. Le plan de contrôle, à l'aide des informations relevées par le plan de gestion, déploie le service sur le réseau. Les modules de découverte du plan de gestion alerte en cas de panne ou d'évolution de la topologie. Cette alerte est répercutée sur le plan de contrôle qui applique le mécanisme dynamique.

4.4 Protocole expérimental

4.4.1 Réseau d'expérimentation

Afin de valider le comportement du contrôleur Seamless implémenté, différents scénarios sont étudiés en déploiement réel.

Différentes topologies réseau sont déployées à l'aide du logiciel de virtualisation Mininet [77]. La virtualisation est un procédé permettant d'exécuter sur une machine hôte une application ou un système d'exploitation dans un environnement isolé, quelle que soit la plateforme physique de l'hôte. La virtualisation est un moyen d'émulation : reproduire le fonctionnement d'un système dans un autre système à travers des interfaces.

L'émulation, et par extension la virtualisation, diffère de la simulation. La simulation consiste à reproduire le comportement général d'un système à travers un modèle conceptuel. Si la simulation facilite les expérimentations, l'objectif de cette partie est d'étudier le comportement du contrôleur dans un cadre réel. Ainsi, l'approche par émulation permet de valider plus justement le comportement du contrôleur. À noter toutefois que l'échelle de temps peut ne pas être respectée.

Mininet [77] est un logiciel permettant de déployer un ensemble de composants virtuels :

- des commutateurs virtuels. Dans le cadre de cette expérimentation, on utilisera des commutateurs OpenVSwitch [78], dont on exploitera les fonctionnalités OpenFlow version 1.0 ;
- des hôtes virtuels, qui sont des sous-processus Unix du système hôte, et qui possèdent des interfaces Ethernet indépendantes ;
- des liaisons virtuelles, créant des liens entre des interfaces Ethernet d'hôtes ou de commutateurs. Les caractéristiques de ces liaisons peuvent être contrôlées : bande passante, latence, gigue, taux de pertes...

4.4.2 Paramètres du contrôleur

Les différents modules du contrôleur ont été présentés en 4.3. Les paramètres de certains modules doivent être définis pour compléter le cadre expérimental. Les valeurs temporelles fixées pour cette expérience ne sont pas représentatives du système réel, mais sont adaptées à l'observation du comportement.

Le module de détection de panne émet un paquet toutes les 100 ms, et déclare la perte d'une liaison si 3 paquets consécutifs sont perdus. Autrement dit, le temps de détection d'une panne vaut entre 300 ms et 400 ms.

Moins critique, le module de mesure de latence effectue une mesure toutes les 2 s. Cette période correspond à des valeurs usuelles sur le réseau étudié. C'est-à-dire que le jeu de 3 paquets permettant le calcul de latence d'un lien est émis à une période de 2 s par lien.

Enfin, les modules de calcul et déploiement des arbres utilisent l'algorithme IS, fondé sur l'algorithme CBF, respectivement présentés en 3.3 et 3.2.2. Concernant le déploiement, une temporisation entre les différentes actions est définie. En effet, l'installation et la suppression des règles OpenFlow définies par le contrôleur peut mettre un peu de temps à être appliquée et fonctionnelle sur les commutateurs. On définit une temporisation de 3 s après un ensemble d'ordres de déploiement et 2 s après un ensemble d'ordres de suppression.

4.4.3 Règles de commutation

Afin de déployer sur le réseau les arbres de multidiffusion calculés par le contrôleur, deux formats de règles OpenFlow sont définis. Les règles de commutation choisies portent sur l'adresse IP source, et le tag VLAN. L'adresse source sert à identifier le flux, à la manière des réseaux traditionnels. Le tag VLAN permet de différencier les deux arbres de multidiffusion sur le réseau. L'utilisation de tags VLAN a été sélectionnée pour des raisons de simplicité, d'autres critères auraient put être implémentés, comme un label MPLS par exemple. Les paquets correspondant à la règle sont commutés vers un port de sortie, correspondant au saut suivant. La figure 4.8 illustre le déploiement utilisé dans le cadre de cette expérimentation.

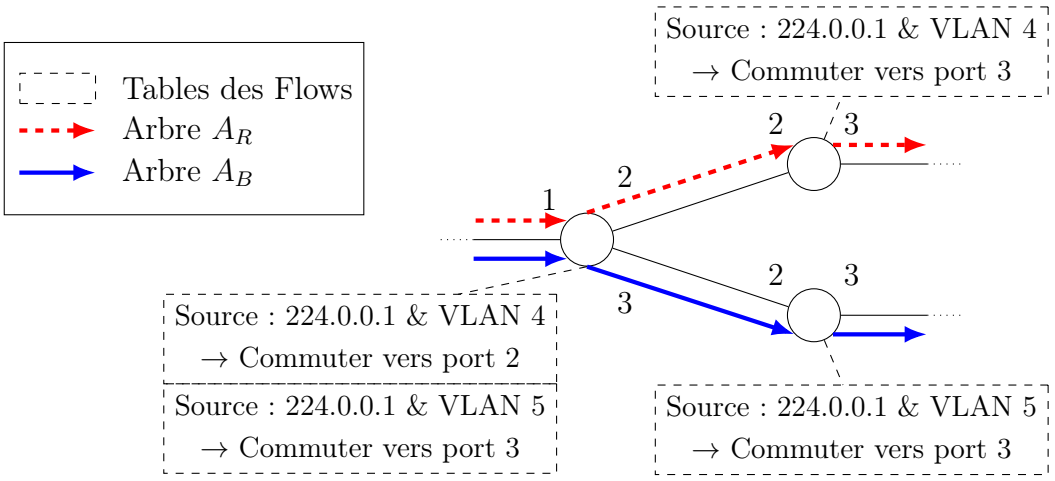


FIGURE 4.8 – Exemple de déploiement de règles OpenFlow définies dans le cadre expérimental

Par défaut, le port 1 de tous les commutateurs a été réservé pour la connexion à un hôte. Ainsi, le contrôleur déploie des règles qui dirigent les flux vers le port 1 de chacun des nœuds terminaux des arbres de multidiffusion. Lors de la suppression d'un arbre, toutes les règles sont supprimées, excepté les règles vers les hôtes. Néanmoins, si le nœud portant une destination est également sur le chemin vers une autre destination, alors le processus de suppression retirera la règle de commutation entière. La commutation en sortie du réseau sera rétablie au déploiement.

4.4.4 Émulation de récepteurs Seamless

Afin d'évaluer le comportement du contrôleur, un émetteur et des récepteurs Seamless sont nécessaires. Deux hôtes virtuels permettront d'émuler l'émission et la réception de paquets. La figure 4.9 présente le modèle de comportement d'un émetteur et d'un récepteur simulant un comportement Seamless. Le trafic émis sur le réseau de test est composé de paquets IP numérotés par un identifiant noté i. Chaque paquet est émis en deux exemplaires au même identifiant sur deux VLANs distincts. Les numéros de VLAN 4 et 5 ont été choisis arbitrairement. Les paquets sont émis environ toutes les 200 ms, la précision de cette période étant limitée par la machine hôte de l'expérience. Une période d'émission de 200 ms permettrait d'observer les cas de pertes, car le temps de détection d'une panne et de réparation est d'au moins 300 ms.

Dans le cadre de cette expérimentation, on travaille sous l'hypothèse que les mémoires tampon des équipements de réception sont adaptées au réseau.

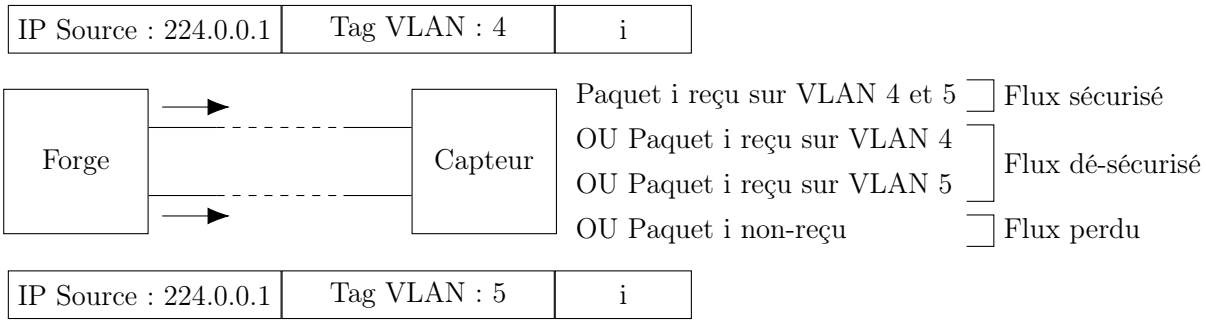


FIGURE 4.9 – Émulation de réception Seamless

4.4.5 Déroulement du protocole

Le protocole expérimental se décompose de la manière suivante :

1. Démarrage du contrôleur ;
2. Démarrage de la topologie virtuelle :
   (a) Démarrage des hôtes émetteurs et récepteurs ;
   (b) Démarrage des commutateurs ;
   (c) Démarrage des liaisons.
3. Commande d'un service de multidiffusion à travers l'interface ;
4. Évènement sur le réseau.

L'analyse des relevés sur les hôtes récepteurs permet de définir si le comportement du contrôleur au cours de l'expérience est bien celui attendu. En particulier, on souhaite valider les réactions aux évènements, c'est-à-dire la conservation d'au moins l'un des flux, y compris dans les périodes de bascule d'arbres.

4.5 Résultats

Dans cette partie, les différents scénarios sont présentés, suivis des résultats expérimentaux observés. L'expérience a pour objectif la vérification du comportement du contrôleur et ses conséquences, en particulier l'état du flux final. Pour chaque scénario, la topologie et les états de déploiement attendus sont présentés avant les résultats et leur analyse qui mettra en évidence les comportements réels du contrôleur et du réseau. La figure 4.10 résume l'approche de cette expérimentation.

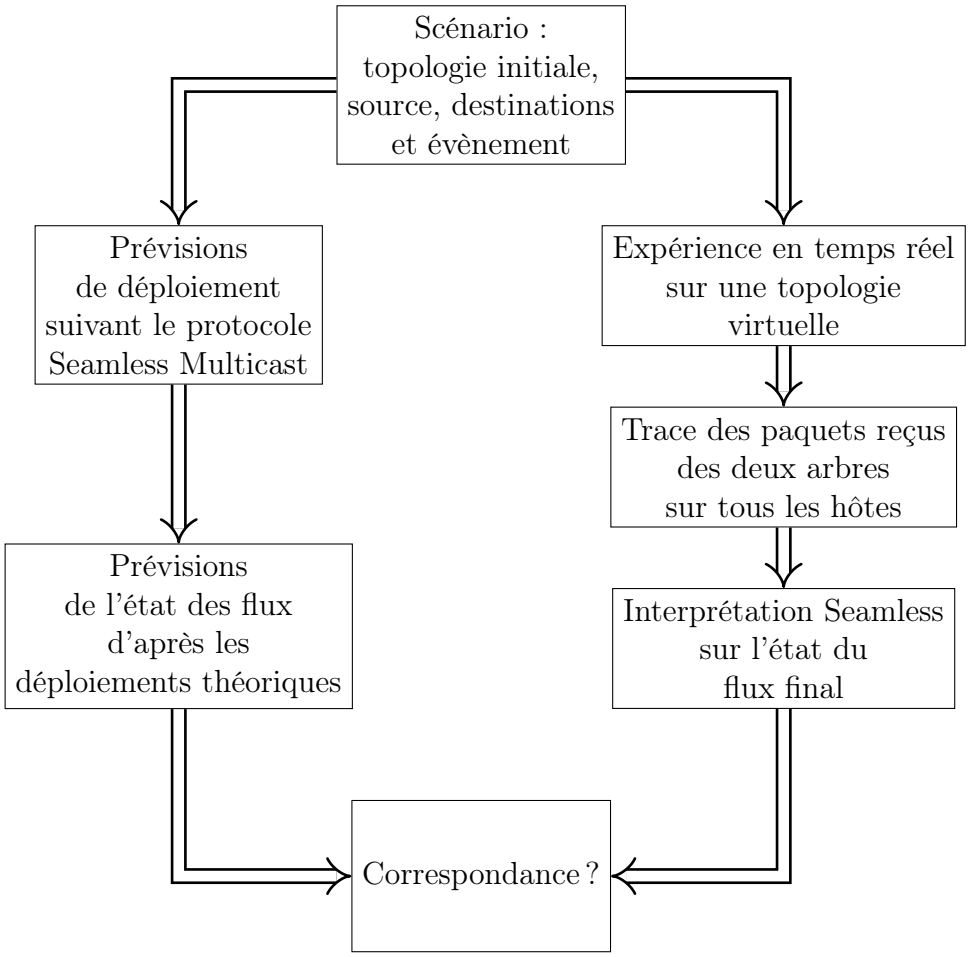


FIGURE 4.10 – Principe expérimental vérification du comportement de l'implémentation

4.5.1 Panne impactant un seul arbre, indépendance stricte

Dans ce scénario, on déploie le contrôleur sur la topologie présentée en figure 4.11b. Le nœud 1 porte un hôte $h1$ qui émet le flux. Les nœuds 5 et 6 portent deux hôtes $h5$ et $h6$ qui sont les destinations du flux. Le lien entre les nœuds 1 et 2 sera mis en défaut.

Prévisions

Les figures 4.11c, 4.11d et 4.11e présentent les états attendus du réseau au cours du scénario. Le déploiement initial est illustré par la figure 4.11c : deux arbres strictement indépendants sont déployés et alimentent correctement les destinations. Au moment de la panne, une branche de l'arbre A_B sera impactée, et l'hôte $h5$ ne recevra plus qu'un seul flux, comme décrit en figure 4.11d. Le contrôleur calculera alors une nouvelle paire d'arbres, et déploiera dans un premier temps le nouvel arbre A_B, puis le nouvel arbre A_R (identique dans cet exemple) pour arriver à un état final présenté en figure 4.11e.

On s'attend donc à observer un état initial sécurisé, puis un état dé-sécurisé sur $h5$ avant de revenir à un état stable. Il est possible d'observer des états dé-sécurisés transitoires au moment des re-déploiements de nouveaux arbres.

Résultats

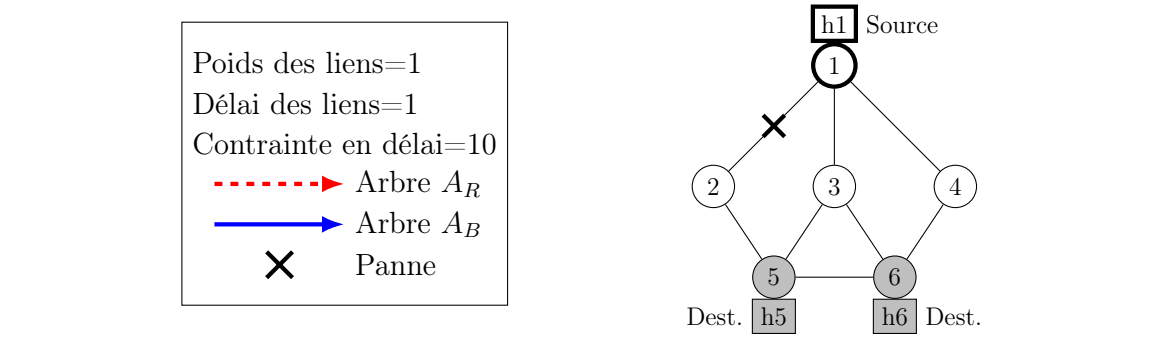
Les deux graphiques de la figure 4.12 présentent les paquets reçus par les hôtes $h5$ et $h6$ au cours de l'expérience, ainsi que l'état du flux final correspondant, suivant le comportement Seamless défini en 4.4.4. L'arbre A_R présenté dans la figure 4.11 correspond à tout ce qui est transporté suivant le VLAN 4, et l'arbre A_B correspond au VLAN 5. Lorsque des paquets identiques sont reçus à travers chacun des arbres, ils sont marqués comme identiques dans le graphique, et l'état du flux associé est considéré sécurisé.

Le données de l'expérience ont été tronquées, les états stables initiaux et finaux perdurent au delà des limites du graphique fourni. En particulier, le temps 0 du graphe ne correspond pas à l'instant de démarrage du récepteur, mais à un temps arbitrairement choisi comme début de l'observation. Ce choix a été fait pour faciliter la lecture.

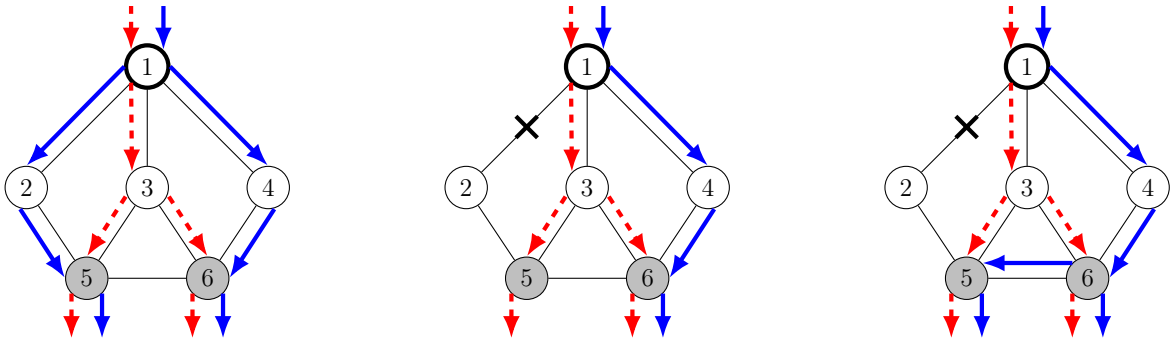
L'évènement du scénario, la suppression du lien du nœud 1 vers le nœud 2, apparaît sur le graphique autour de 3 s. Avant cela, on observe l'état stable du réseau, correspondant à l'état attendu présenté dans la figure 4.11c.

Environ 1 s après l'évènement prévu dans le scénario, on observe une coupure du flux de l'arbre A_B sur l'hôte $h5$. Ce délai entre l'évènement et la coupure est lié au trafic en transit sur le lien du nœud 2 vers 5, toujours actif au moment de la panne.

Plus d'une seconde après la panne, soit le temps de détection de la panne et de calcul d'une nouvelle paire, on observe une coupure du flux de l'arbre A_B sur l'hôte $h6$. Cette coupure est due au redéploiement de l'arbre A_B : l'ordre de suppression émis vers 3.4 s et plus tard vers 5.4 s l'ordre de déploiement.



(a) Légende (b) Scénario



(c) État initial attendu : 5 et 6 transmettent les deux flux

(d) État intermédiaire attendu : 5 ne transmet plus qu'un seul flux

(e) État final attendu : 5 et 6 transmettent les deux flux

FIGURE 4.11 – Scénario A : Panne impactant un seul arbre, indépendance stricte possible

Vers 7.4 s, soit 2 s après le déploiement, $h6$ retrouve le flux de l'arbre A_B, ce qui correspond au temps de transport du flux du nœud 1 vers le nœud 6, alors que $h5$ retrouve le flux 3 s après le déploiement. Les flux reçus par $h5$ sont décalés dans le temps, c'est-à-dire que le flux de l'arbre A_B est acheminé 1 s plus tard que le flux de l'arbre A_R.

L'ordre de suppression de l'arbre A_R est envoyé vers 8.4 s, et l'ordre d'installation vers 10.4 s. $h5$ et $h6$ retrouvent ainsi le flux de l'arbre A_R 2 s plus tard vers 12.4 s.

On observe ainsi un état dé-sécurisé sur $h5$ suite à la panne, et d'autres états dé-sécurisés dus aux redéploiements des arbres 1 et 2.

L'absence d'état du flux sur le graphe de $h5$ entre 9.5 s et 10.5 s est lié au modèle d'interprétation du Seamless présenté en 4.4.4. En effet, les paquets de l'arbre A_B reçus dans cet intervalle correspondent aux paquets de l'arbre A_R reçus 1 s plus tôt, et donc à l'état sécurisé de l'intervalle décalé. Les paquets de l'arbre A_B immédiatement reçus après 10.5 s sont interprétés comme un état dé-sécurisé.

Un second phénomène est observable sur $h5$ entre 12.5 s et 14.5 s. Deux états sont superposés dans le graphe. Cela est à nouveau dû au décalage temporel des flux de l'arbre A_R et A_B. En effet, les paquets de l'arbre A_B reçus au début de cet intervalle existent en un seul exemplaire, alors que les paquets de l'arbre A_R ont une redondance sur l'arbre A_B plus tard. Ainsi, $h5$ reçoit dans le même intervalle des paquets non-sécurisés et sécurisés.

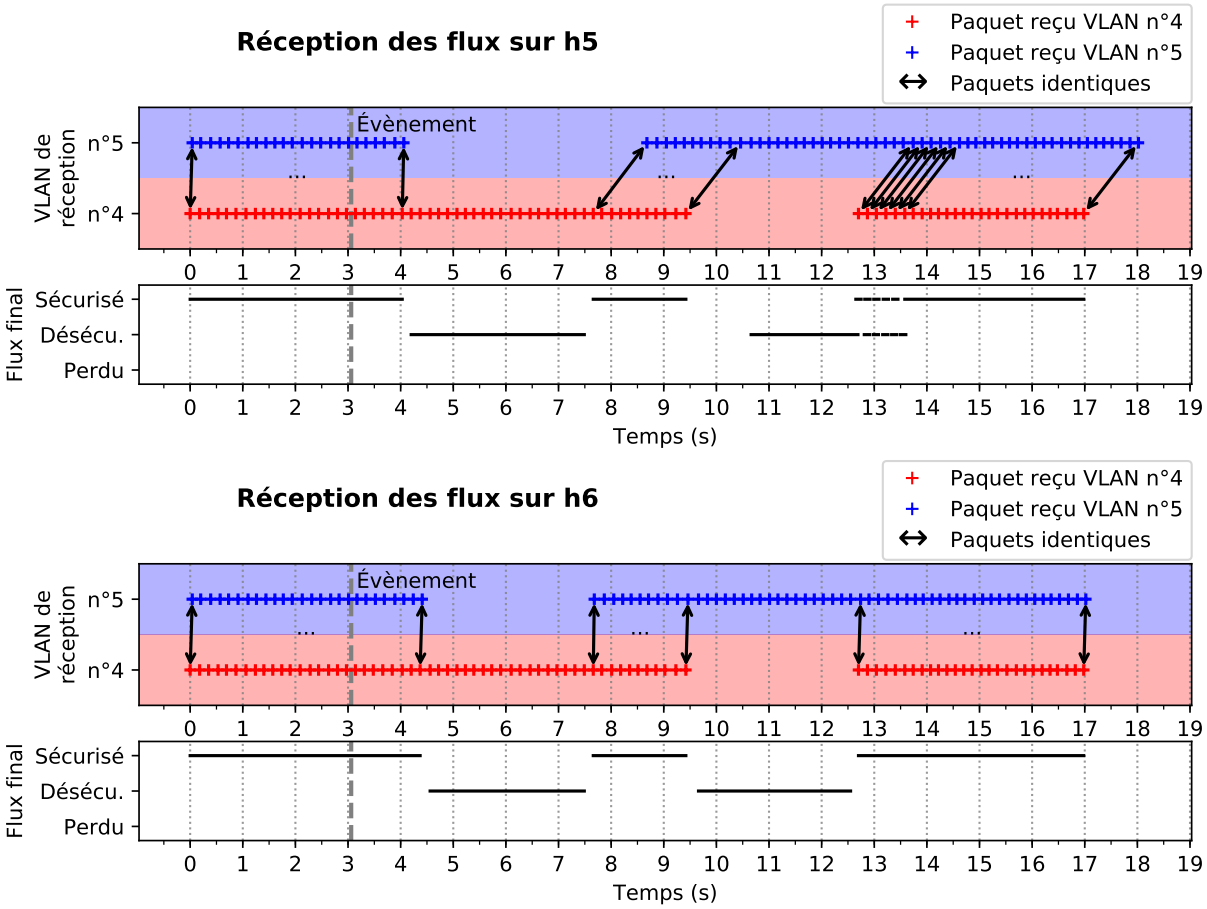


FIGURE 4.12 – Résultats du scénario A

On peut résumer la lecture du graphe de la manière suivante :

3.0 s Évènement : le lien entre les nœuds 1 et 2 est inactif ;

3.4 s L'ordre de suppression de l'arbre A_B est émis par le contrôleur ;

4.0 s $h5$ perd le flux de l'arbre A_B environ 1 s après l'évènement, car une partie des paquets était en transit sur le lien entre les nœuds 2 et 5 ;

4.4 s $h6$ perd le flux de l'arbre A_B suite à la suppression ;

5.4 s L'ordre de déploiement de l'arbre A_B est émis par le contrôleur ;

7.4 s $h6$ reçoit le flux de l'arbre A_B, environ 2 s après le déploiement à cause du temps d'acheminement du flux ;

8.4 s $h5$ reçoit le flux de l'arbre A_B, environ 1 s plus tard que $h6$;

8.4 s L'ordre de suppression de l'arbre A_R est émis par le contrôleur ;

10.4 s L'ordre de déploiement de l'arbre A_R est émis par le contrôleur ;

12.4 s $h5$ et $h6$ reçoivent le flux de l'arbre A_R en même temps, environ 2 s après le déploiement à cause du temps d'acheminement du flux.

En conclusion, le contrôleur a bien le comportement attendu. Malgré une panne, les hôtes reçoivent à tout instant un flux, et le réseau tend à revenir à une situation sécurisée.

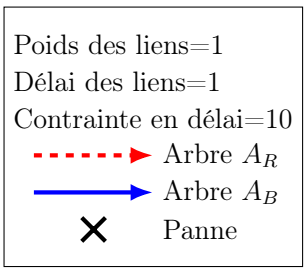
4.5.2 Panne impactant un seul arbre, indépendance maximale

Dans ce scénario, on déploie le contrôleur sur la topologie présentée en figure 4.13b. L'hôte $h1$ émet le flux, $h5$ et $h6$ sont destinataires. Le lien entre 1 et 2 sera mis en défaut.

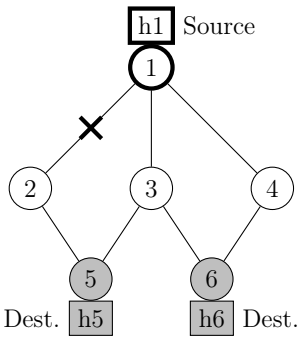
Prévisions

Les figures 4.13c, 4.13d et 4.13e présentent les états attendus du réseau au cours du scénario. Ce scénario est très similaire au premier présenté en 4.5.1. La différence majeure est due à l'absence de liaison entre 5 et 6. Ainsi, une indépendance stricte est impossible dans ce cas.

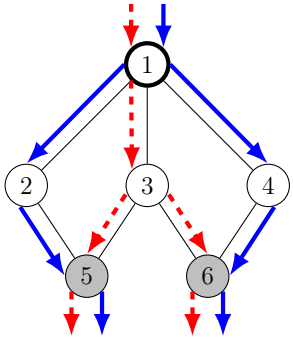
On s'attend donc à observer des résultats similaires : un état initial sécurisé, puis un état dé-sécurisé sur $h5$ avant de revenir à un état stable. Il est possible d'observer des états dé-sécurisés transitoires au moment des re-déploiements de nouveaux arbres.



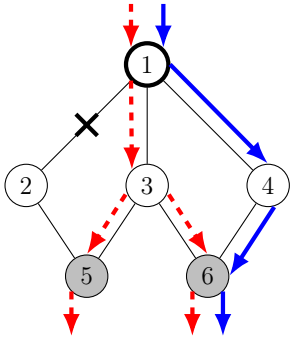
(a) Légende



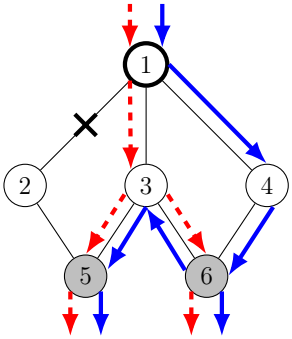
(b) Scénario



(c) État initial attendu : 5 et 6 transmettent les deux flux



(d) État intermédiaire attendu : 5 ne transmet plus qu'un seul flux



(e) État final attendu : 5 et 6 transmettent les deux flux

FIGURE 4.13 – Scénario B : Panne impactant un seul arbre, indépendance maximale

Résultats

On observe sur $h5$ une superposition d'états du flux entre 12.5 s et 16.5 s. Ce phénomène est expliqué plus en détail en 4.5.1.

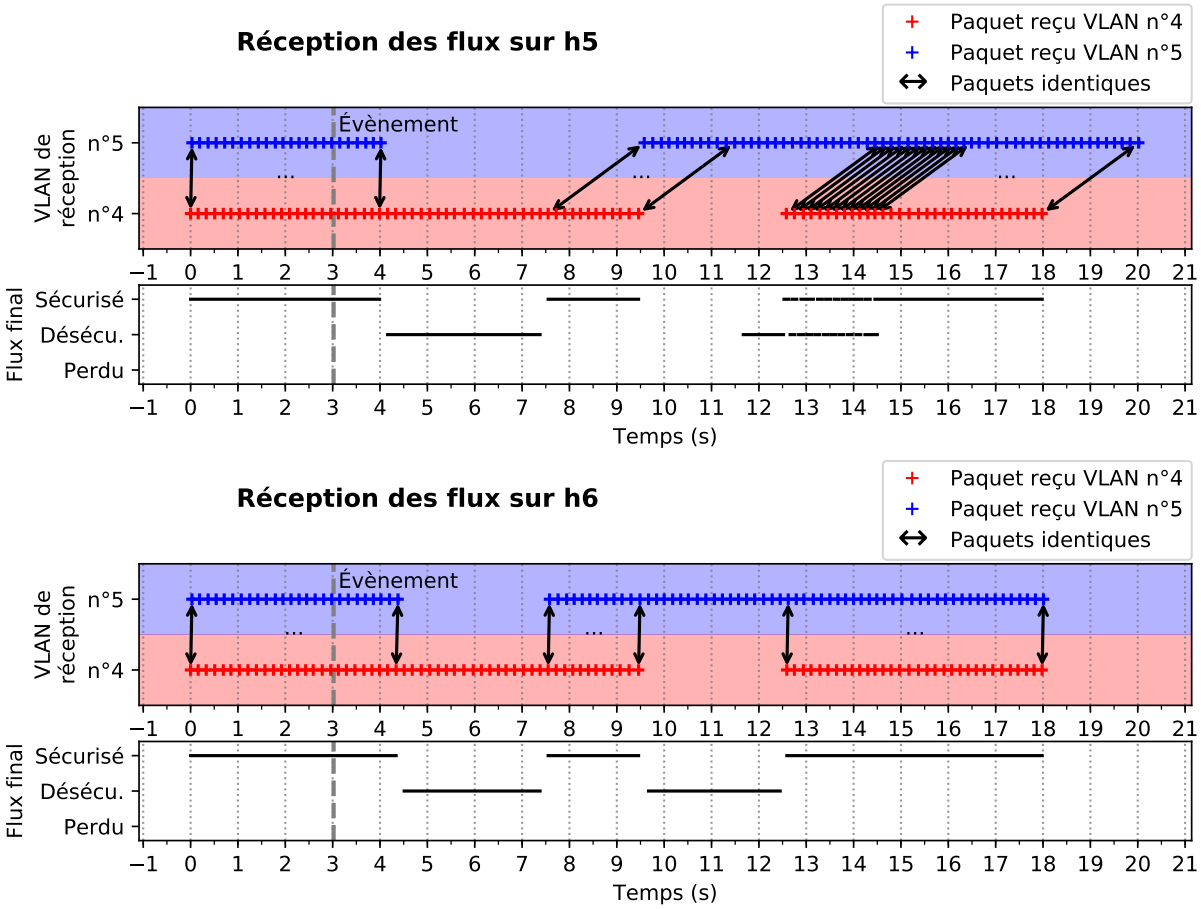


FIGURE 4.14 – Résultats du scénario B

On peut résumer la lecture du graphe de la manière suivante :

3.0 s Évènement : le lien entre les nœuds 1 et 2 est inactif ;

3.4 s L'ordre de suppression de l'arbre A_B est émis par le contrôleur ;

4.0 s $h5$ perd le flux de l'arbre A_B environ 1 s après l'évènement, car une partie des paquets était en transit sur le lien entre les nœuds 2 et 5 ;

4.4 s $h6$ perd le flux de l'arbre A_B suite à la suppression ;

5.4 s L'ordre de déploiement de l'arbre A_B est émis par le contrôleur ;

7.4 s $h6$ reçoit le flux de l'arbre A_B, environ 2 s après le déploiement à cause du temps d'acheminement du flux ;

8.4 s L'ordre de suppression de l'arbre A_R est émis par le contrôleur ;

9.4 s $h5$ reçoit le flux de l'arbre A_B, environ 2 s plus tard que $h6$;

10.4 s L'ordre de déploiement de l'arbre A_R est émis par le contrôleur ;

12.4 s $h5$ et $h6$ reçoivent le flux de l'arbre A_R en même temps, environ 2 s après le déploiement à cause du temps d'acheminement du flux.

En conclusion, le contrôleur a bien le comportement attendu. Malgré une panne, les hôtes reçoivent à tout instant un flux, et le réseau tend à revenir à une situation sécurisée.

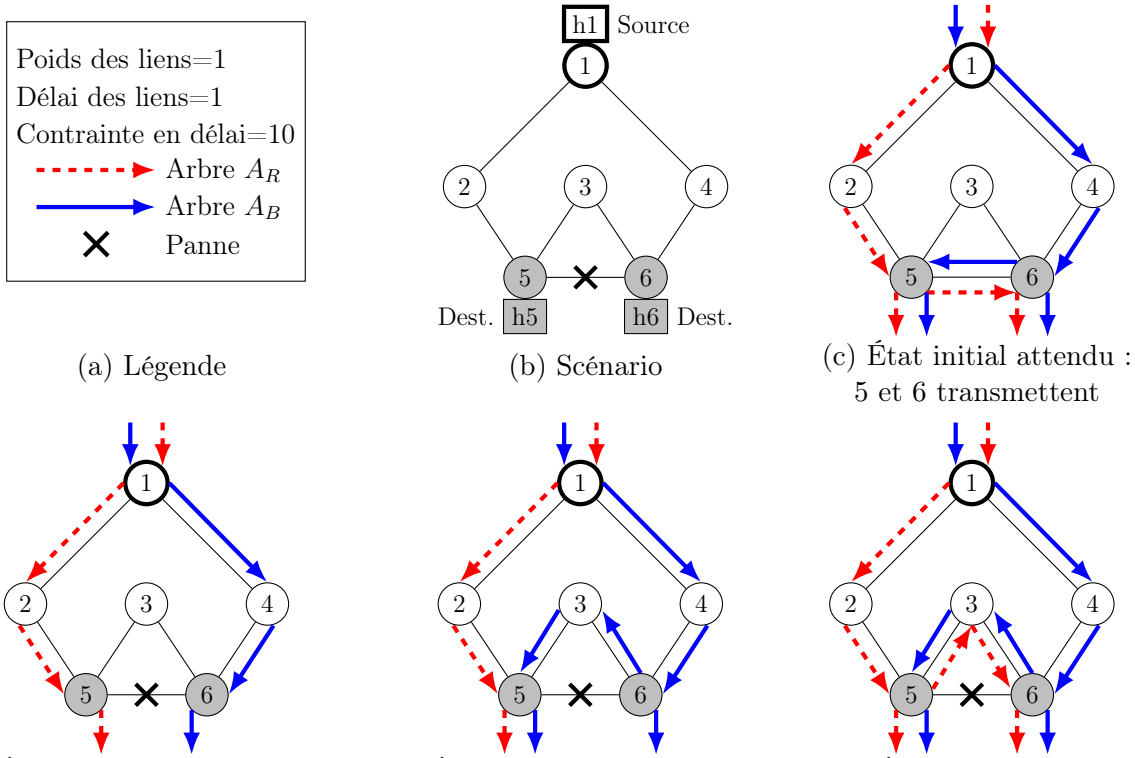
4.5.3 Panne impactant les deux arbres

Dans ce scénario, on déploie le contrôleur sur la topologie présentée en figure 4.15b. L'hôte $h1$ émet le flux, $h5$ et $h6$ sont destinataires. Le lien entre 5 et 6 sera mis en défaut.

Prévisions

Les figures 4.15c, 4.15d, 4.15e et 4.15f présentent les états attendus du réseau au cours du scénario. Dans l'état initial, les arbres partagent dans des sens différents le lien de 5 vers 6. La panne aura alors un impact sur les deux arbres, comme le montre alors la figure 4.15d. Le raccordement rapide de l'arbre A_R aura lieu suivant la figure 4.15e. Enfin, le basculement des deux arbres amènera au déploiement illustré en figure 4.15f.

On s'attend donc à observer un état sécurisé, puis un état dé-sécurisé sur $h5$ et $h6$. On devrait ensuite observer une sécurisation sur $h6$ et puis sur $h5$ ramenant à un état stable.



(a) Légende

(b) Scénario

(c) État initial attendu : 5 et 6 transmettent

(d) État intermédiaire attendu : 5 et 6 ne transmettent plus qu'un seul flux

(e) État intermédiaire attendu : 5 ne transmet plus qu'un seul flux

(f) État final attendu : 5 et 6 transmettent les deux flux

FIGURE 4.15 – Scénario C : Panne impactant les deux arbres

Résultats

On observe sur $h6$ une superposition d'états du flux entre 15.8 s et 17.8 s, voir 4.5.1.

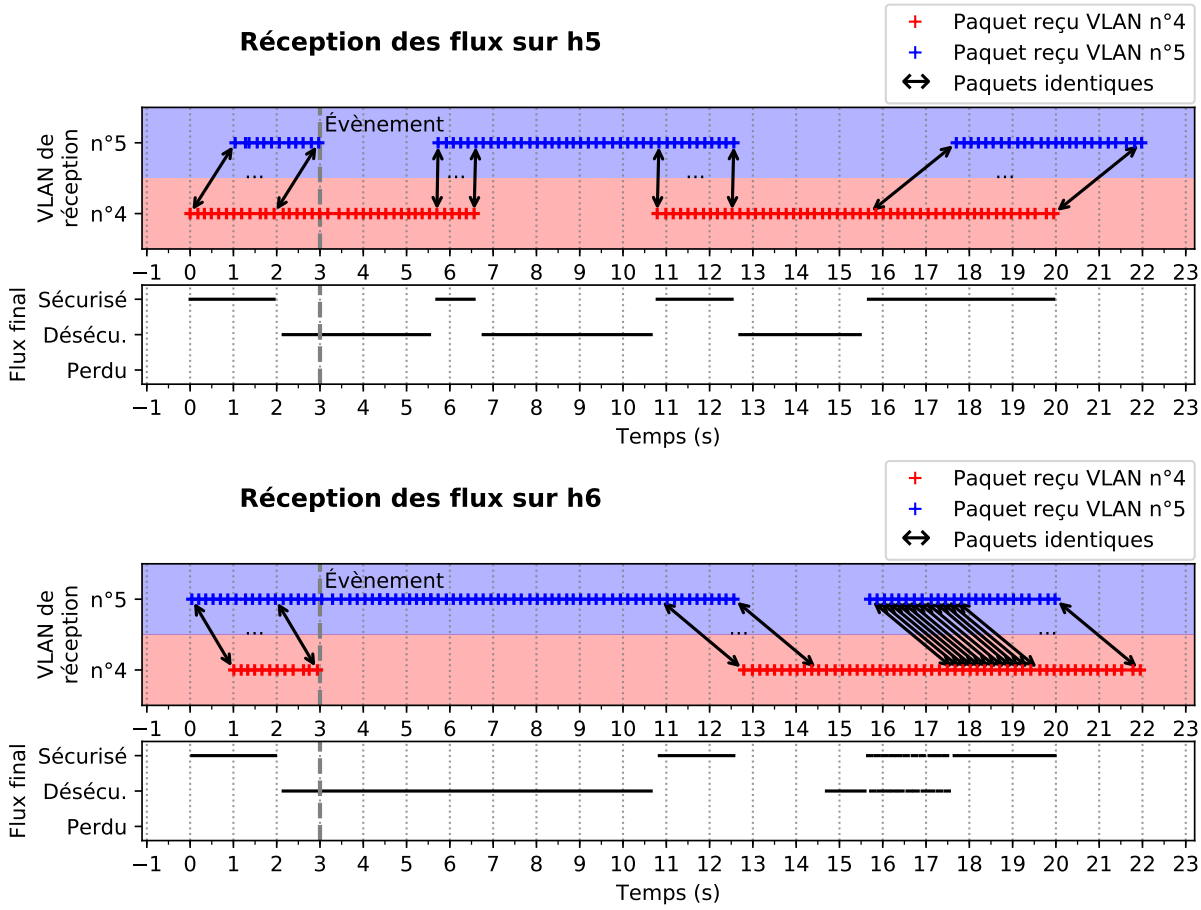


FIGURE 4.16 – Résultats du scénario C

On peut résumer la lecture du graphe de la manière suivante :

3.0 s Évènement : le lien entre les nœuds 1 et 2 est inactif, $h5$ perd le flux de l'arbre A_B immédiatement et $h6$ celui de l'arbre A_R ;

3.8 s L'ordre de raccordement rapide de l'arbre A_B est émis ;

5.8 s $h5$ reçoit le flux de l'arbre A_B, environ 2 s après le déploiement ;

6.8 s L'ordre de suppression de l'arbre A_R est émis par le contrôleur, $h5$ perd immédiatement le flux car la règle de sortie a été supprimée (voir 4.4.3) ;

8.8 s L'ordre de déploiement de l'arbre A_R est émis par le contrôleur ;

10.8 s $h5$ reçoit le flux de l'arbre A_R, environ 2 s après le déploiement ;

11.8 s L'ordre de suppression de l'arbre A_B est émis par le contrôleur ;

12.5 s $h5$ et $h6$ perdent le flux de l'arbre A_B suite à la suppression ;

12.8 s $h6$ reçoit le flux de l'arbre A_R, environ 2 s plus tard que $h5$. Grâce au décalage des flux, aucun paquet n'est considéré perdu ;

13.8 s L'ordre de déploiement de l'arbre A_B est émis par le contrôleur ;

15.8 s $h6$ reçoit le flux de l'arbre A_B ;

17.8 s $h5$ reçoit le flux de l'arbre A_B.

Le contrôleur a bien le comportement attendu. Malgré une panne impactant les deux arbres, les hôtes reçoivent à tout instant un flux, et le réseau tend à revenir à une situation sécurisée.

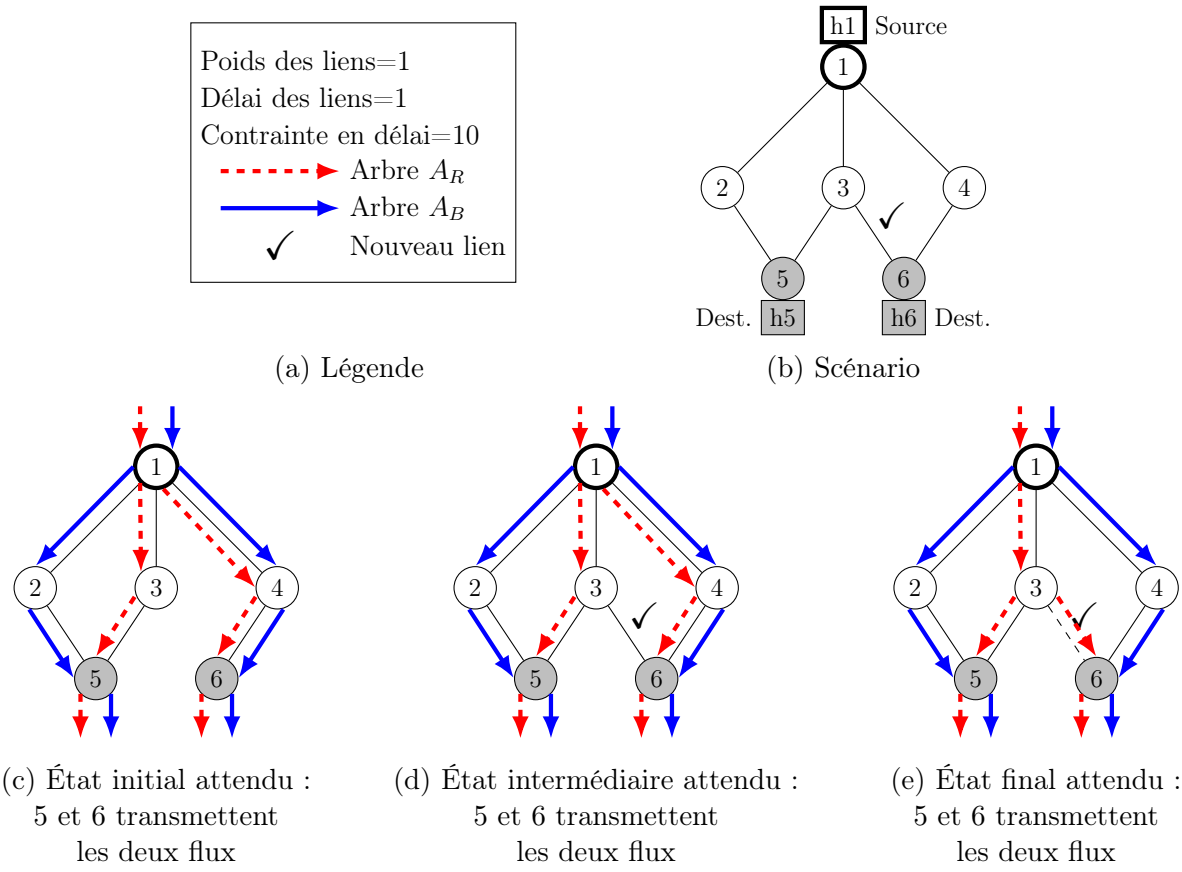
4.5.4 Création d'une nouvelle liaison

Dans ce scénario, on déploie le contrôleur sur la topologie présentée en figure 4.17b. L'hôte $h1$ émet le flux, $h5$ et $h6$ sont destinataires. Le lien entre les nœuds 3 et 6 n'existe pas au départ, et sera ajouté au cours de l'expérience.

Prévisions

Les figures 4.17c, 4.17d et 4.17e présentent les états attendus du réseau au cours du scénario. Le déploiement initial prévu est illustré par la figure 4.17c. Étant donné qu'il n'existe qu'un seul chemin vers le nœud 6, les deux arbres empruntent le même chemin. Lorsque le lien est ajouté, aucune des destinations ne sera affectée comme l'illustre la figure 4.17d. De nouveaux arbres plus indépendants seront calculés et déployés l'un après l'autre pour arriver à la figure 4.17e.

On s'attend donc à observer un état initial sécurisé, puis de possibles états dé-sécurisés transitoires au moment des redéploiements de nouveaux arbres, avant de revenir à un état sécurisé.



(a) Légende

(b) Scénario

(c) État initial attendu : 5 et 6 transmettent les deux flux

(d) État intermédiaire attendu : 5 et 6 transmettent les deux flux

(e) État final attendu : 5 et 6 transmettent les deux flux

FIGURE 4.17 – Scénario D : Nouveau lien permettant une meilleure situation

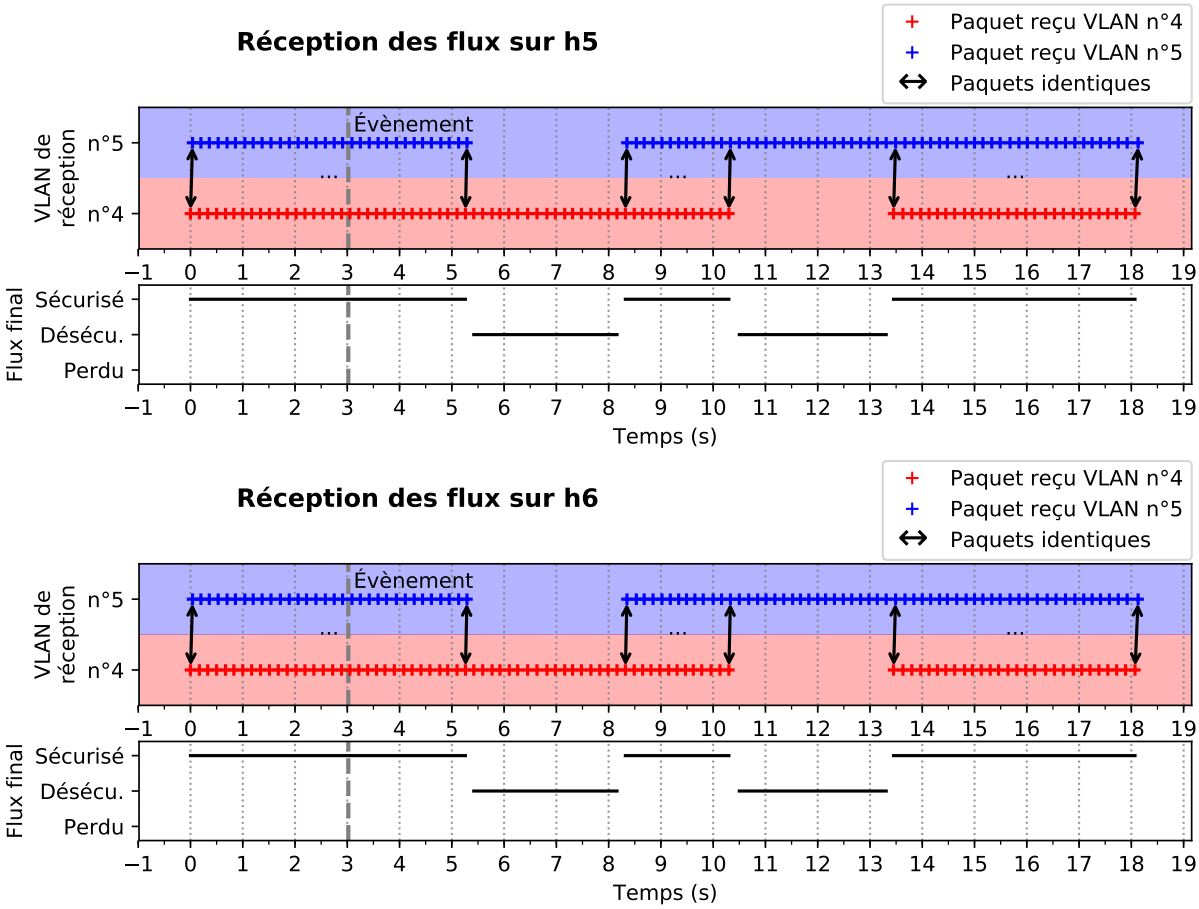


FIGURE 4.18 – Résultats du scénario D

Résultats

On peut résumer la lecture du graphe de la manière suivante :

3.0 s Évènement : le lien entre les nœuds 5 et 6 est ajouté ;
4.3 s L'ordre de suppression de l'arbre A_B est émis par le contrôleur ;
5.3 s $h5$ et $h6$ perdent le flux de l'arbre A_B en même temps suite à la suppression ;
6.3 s L'ordre d'installation de l'arbre A_B est émis par le contrôleur ;
8.3 s $h5$ et $h6$ reçoivent le flux de l'arbre A_B en même temps, environ 2 s après le déploiement à cause du temps d'acheminement du flux ;
9.3 s L'ordre de suppression de l'arbre A_R est émis par le contrôleur ;
10.3 s $h5$ et $h6$ perdent le flux de l'arbre A_R en même temps suite à la suppression ;
11.3 s L'ordre d'installation de l'arbre A_R est émis par le contrôleur ;
13.3 s $h5$ et $h6$ reçoivent le flux de l'arbre A_B en même temps.

Le contrôleur a bien le comportement attendu. En cas d'évolution de la topologie, le contrôleur déploie une nouvelle paire d'arbres plus optimaux. Durant ce déploiement, les hôtes reçoivent à tout instant un flux, et le réseau revient à une situation sécurisée.

4.6 Conclusion

Dans ce chapitre, l'architecture Seamless Multicast proposée au chapitre 2 ainsi que ses algorithmes constituants proposés au chapitre 3 ont été implémentés.

Le Software-Defined Networking, un principe d'architecture réseau centralisé, a été utilisé pour cette implémentation. Grâce à la séparation du plan de contrôle et du plan de données, le protocole associé au Seamless Multicast peut exploiter la vision complète et en temps réel de la topologie.

Une implémentation complète utilisant le protocole OpenFlow a été présentée dans ce chapitre. Le contrôleur développé a permis de valider expérimentalement, sur un réseau virtuel, le comportement associé à l'architecture Seamless Multicast. Autrement dit, quelle que soit l'évolution du réseau, le contrôleur assure bien le transport du flux vers les destinations grâce à la combinaison de redondance et de résilience proposée.

Le contrôleur a été testé sur des topologies limitées, mais le passage à l'échelle sur un réseau de taille réelle est parfaitement possible.

En effet, d'une part, il a été montré au chapitre 3 que les heuristiques utilisées par le protocole Seamless Multicast passent à l'échelle.

D'autre part, les problématiques de mise en mémoire de la topologie et de ses caractéristiques sont liées aux limitations matérielles de la machine hôte du contrôleur, qui doit simplement être choisie selon le besoin.

L'augmentation de la taille du réseau, et donc de la taille des services gérés par le protocole, implique une probabilité d'impact plus grande, comme montré en 2.1.3. Cela pourrait amener le protocole à un fonctionnement trop fréquent du mécanisme dynamique. Néanmoins, la fiabilité du réseau étudié est connue, et la probabilité d'occurrence d'évènements est faible. Le passage à l'échelle ne devrait pas être impacté.

Enfin, les temps de déploiement et les différentes temporisations doivent être étudiés sur le réseau réel pour valider un passage à l'échelle et une mise en exploitation. En effet, suivant les temps de transport, les temps de déploiement, et les mémoires tampons utilisées sur le réseau, un paramétrage différent du contrôleur peut être nécessaire.

Conclusion

Ces travaux de thèse s'inscrivent dans la problématique du transport audiovisuel, dans un contexte de diffusion de la TNT. En particulier, cette thèse traite du transport de flux audiovisuels temps réel sur un réseau contraint, soumis à de fortes exigences en disponibilité de service. Une architecture réseau appelée Seamless Multicast a été proposée dans cette thèse : l'acheminement redondant d'un flux par deux arbres de multidiffusion les plus disjoints possible, évoluant dynamiquement avec le réseau.

Le premier chapitre de cette thèse présente le contexte lié au transport de flux audiovisuels, en particulier dans le cadre de la multidiffusion, ainsi que les contraintes du réseau étudié qui ont orienté ces travaux.

La proposition du protocole de transport Seamless Multicast fait l'objet du Chapitre 2 de cette thèse. Deux arbres de multidiffusion sont établis entre une source et de multiples destinations. Le problème appelé DMDT correspondant au calcul d'une telle paire d'arbres sous contraintes est formellement décrit dans ce chapitre. En cas de panne ou d'évolution du réseau, le protocole assure la convergence vers une nouvelle paire optimale d'arbres tout en préservant la continuité de service pour chaque destination.

Le chapitre 3 présente trois propositions d'algorithmes pour la résolution du problème DMDT : deux heuristiques d'exploration ainsi qu'un algorithme génétique. Des expérimentations sur des cas caractéristiques du réseau étudié ont montré la qualité suffisante des solutions trouvées par les deux heuristiques, et la compatibilité des temps de calcul à cette application temps réel. Néanmoins, des performances limitées ont été mises en évidence pour l'algorithme génétique, à la fois en temps de calcul et en qualité de solution.

Enfin, dans le dernier chapitre de cette thèse, une architecture réseau implémentant le Seamless Multicast a été proposée. Fondé sur le Software-Defined Networking, un contrôleur réseau centralisé exploite une vue globale du réseau pour appliquer les algorithmes et mécanismes du Seamless Multicast. Il est ainsi possible de déployer des services de multidiffusion temps réel à très haute disponibilité sur un réseau. Une implémentation du protocole Seamless Multicast a été réalisée à des fins de test et de preuve de concept. Des expérimentations avec le contrôleur développé ont permis de valider son fonctionnement sur des cas fondamentaux.

Les principales contributions scientifiques de cette thèse sont :

— le Seamless Multicast, un nouveau protocole visant à assurer le transport d'un flux audiovisuel temps réel d'une source vers de multiples destinations, sans perte, y compris dans les phases transitoires ;

— la formulation du problème DMDT du calcul d'une paire d'arbres maximalement indépendants, contraints en délai, dont le coût total est minimal ;
— la proposition et l'étude dans un premier temps d'un algorithme appelé SHERPA permettant le calcul d'une paire de chemin maximalement indépendants, contraints en délai, dont le coût total est minimal ;
— la proposition et l'étude dans un second temps de trois algorithmes visant à résoudre le problème DMDT ;
— la validation expérimentale des algorithmes d'une part, et d'autre part du Seamless Multicast à travers une implémentation sur réseau virtuel.

Étant donné que ces travaux de thèse ont été réalisés dans le cadre d'une collaboration entre le laboratoire CRAN et l'entreprise TDF, il est également important de noter les contributions industrielles :
— un simulateur implémentant les algorithmes proposés sur le réseau de TDF, non-montré dans cette thèse ;
— une implémentation SDN de travaux de cette thèse, qui servira non seulement de prototype pour de futurs déploiements Seamless Multicast mais aussi plus généralement de preuve de concept SDN. En effet, ce contrôleur démontre les possibilités de déploiement d'ingénieries sur-mesure.

Concernant les perspectives de ces travaux de thèse, plusieurs sujets peuvent être approfondis.

L'algorithmique associée au problème DMDT ouvre des possibilités d'étude. Si les algorithmes proposés sont adaptés à une application temps réel, des solutions plus optimales pourraient être nécessaires à d'autres applications. Ainsi, d'autres propositions de résolution sont à envisager.

De plus, l'évaluation de l'influence des paramètres d'une instance du problème DMDT sur les performances d'algorithmes peut être investiguée plus en profondeur. En effet, dans le cadre de ces travaux de thèse, seule l'étude de l'influence du nombre de destinations sur le temps de calcul a été étudiée, l'application industrielle ne faisant pas varier fortement les autres paramètres. Ainsi, des études de types Analyse de Sensibilité Globale [79] pourraient être envisagées pour étudier des algorithmes dans un cadre plus général.

Étant donné que l'objectif de ces travaux est d'assurer la robustesse d'un service de transport, il faudrait également considérer la robustesse du plan de contrôle. En effet, dans une architecture SDN centralisée, le contrôleur est lui même un point unique de panne. Les aspects de sûreté de fonctionnement matérielle et logicielle, ainsi que la sécurité ont étés largement étudiés dans la littérature.

Du point de vue industriel, grâce au simulateur implémentant les algorithmes proposés dans ces travaux, le travail de conception de nouveaux services est grandement facilité. Mais l'objectif à terme est l'implémentation du Seamless Multicast sur le réseau en production. Pour cela, le contrôleur développé au cours de ces travaux servira de prototype à un modèle plus complet, sans doute fondé sur une solution commerciale existante. Le déploiement sur un réseau en exploitation nécessite également une étude industrielle préalable.

Annexe A

Modélisation du Seamless Multicast à l'aide d'un réseau d'automates finis

A.1 Automates finis

Un automate fini est défini par un ensemble d'états fini et un ensemble de transitions fini. À un instant donné, l'automate ne peut se trouver que dans un seul état. Un automate temporisé est un automate fini associé à un ensemble d'horloges aux valeurs réelles, avançant simultanément. Un réseau d'automates temporisés est un ensemble d'automates qui partagent des horloges, et plus généralement un même environnement. Ainsi, les automates peuvent être synchronisés.

Afin de valider le protocole Seamless Multicast développé dans ces travaux de thèse, cette partie propose une traduction du système en un réseau de deux automates, correspondant chacun au plan de contrôle et au plan de données.

A.2 Uppaal

La modélisation et la validation ont été effectuées sur Uppaal 4.0 [80, 81]. Uppaal est un outil pour la validation, la simulation et la vérification de systèmes décrits sous la forme de réseaux d'automates temporisés. Il est doté d'une interface graphique et de sa propre syntaxe de validation.

Sur Uppaal, les états des automates sont représentés par des disques grisés. L'état initial contient un cercle noir. Le nom des états est en rouge. Les transitions sont des flèches reliant les états. Chacune des transitions est associée à :

- une condition booléenne, en vert, utilisant le symbole d'égalité `"=="` ;
- puis une éventuelle synchronisation, utilisant les symboles `"?"` et `"!"`, pour respectivement attendre un signal, ou émettre un signal ;
- puis une éventuelle mise-à-jour de variables, en bleu foncé, utilisant le symbole d'affectation `"="`.

A.3 Modélisation

Contrairement au reste de cette thèse, les arbres sont ici nommées R et B plutôt que A_R et A_B, pour améliorer la lisibilité.

La figure A.1 présente l'automate décrivant le plan de contrôle, c'est-à-dire le comportement du protocole Seamless Multicast proposé dans ces travaux. Dans l'état initial, rien n'est déployé. La première transition est le déploiement initial d'une paire d'arbres, qui amène à l'état stable `"RB_deploye"`. Seules deux transitions sont possibles à partir de cet état, suivant la nature de l'évènement réseau, une panne ou une évolution.

S'il s'agit d'une panne, alors l'état `"analyse_evenement"` est atteint, et la suite dépend de la nature de l'impact de la panne. Les pannes réseau sans impact sur R et B n'apparaissent pas sur l'automate. Elles n'ont pas d'influence sur le plan de contrôle, et sont donc inutiles à cette modélisation. Une évolution du réseau à partir d'une situation stable déclenche immédiatement un nouveau calcul d'arbre, suivant la transition synchronisée à `"evol_reseau"`.

Une destination injoignable est ignorée au moment du calcul d'une nouvelle paire d'arbres afin d'éviter le blocage du protocole. Après le calcul, le déploiement des arbres se déroule dans l'ordre défini par la nature de l'impact, pour revenir à l'état stable `"RB_deploye"`.

Au cours des états `"paire_calculee"`, `"Deploiement_R"` et `"Deploiement_B"`, le processus peut être interrompu en cas de nouvelle panne réseau. Autrement dit, une telle interruption ne peut survenir après que les ordres de suppressions des règles de transport sur le réseau n'aient été émis. On évite ainsi la situation où des destinations ne seraient pas alimentées à cause d'une interruption. Une nouvelle panne ramène l'automate à l'état `"analyse_evenement"`, pour relancer le processus suivant l'impact de la nouvelle panne. Il est possible qu'un impact sur un même arbre survienne, ou sur le second arbre, changeant ainsi les déploiement à faire.

La figure A.2 présente l'automate décrivant le plan de données. Cet automate tient compte de l'état de la topologie relativement au tracé des arbres R et B transportant le flux sur le réseau. Il présente un état initial, où aucun arbre n'est déployé. Après le premier déploiement, seuls quatre états sont possibles :

dests_securisees	Toutes les destinations sont alimentées par les deux arbres ;
dests_jointes_par_R	Toutes les destinations sont alimentées par au moins l'arbre R. Autrement dit, au moins une n'est pas alimentée par l'arbre B ;
dests_jointes_par_B	Toutes les destinations sont alimentées par au moins l'arbre B. Autrement dit, au moins une n'est pas alimentée par l'arbre R ;
dests_non_jointes	Au moins une destination est alimentée uniquement par l'arbre R, et au moins une destination est alimentée uniquement par l'arbre B. Il est possible qu'il s'agisse de la même destination.

Les transitions entre ces états sont liées à des pannes réseau, des évolutions du réseau, et des déploiements ou suppressions des arbres R et B déclenchées par le plan de contrôle.

Les variables booléennes associées au réseau d'automates sont :

impact_R	La panne qui a eu lieu a impacté l'arbre R ;
impact_B	La panne qui a eu lieu a impacté l'arbre B ;
impact_RB	La panne qui a eu lieu a impacté les deux arbres ;
R_deploye	L'arbre R a été déployé sur le réseau ;
B_deploye	L'arbre B a été déployé sur le réseau ;
deployer_RB	L'arbre R doit être déployé avant l'arbre B ;
deployer_BR	L'arbre B doit être déployé avant l'arbre R ;
dest_injoignable	Au moins une destination est isolée de la source ;
dest_ignoree	Les destinations injoignables sont ignorées.

Le système complet est constitué d'une instance de plan de contrôle défini par le patron `Control_plane` et une instance de plan de données défini par `Data_plane`. En syntaxe Uppaal, la composition du système s'écrit de la manière suivante :

```
ctl = Control_plane();
data = Data_plane();
system ctl,data;
```

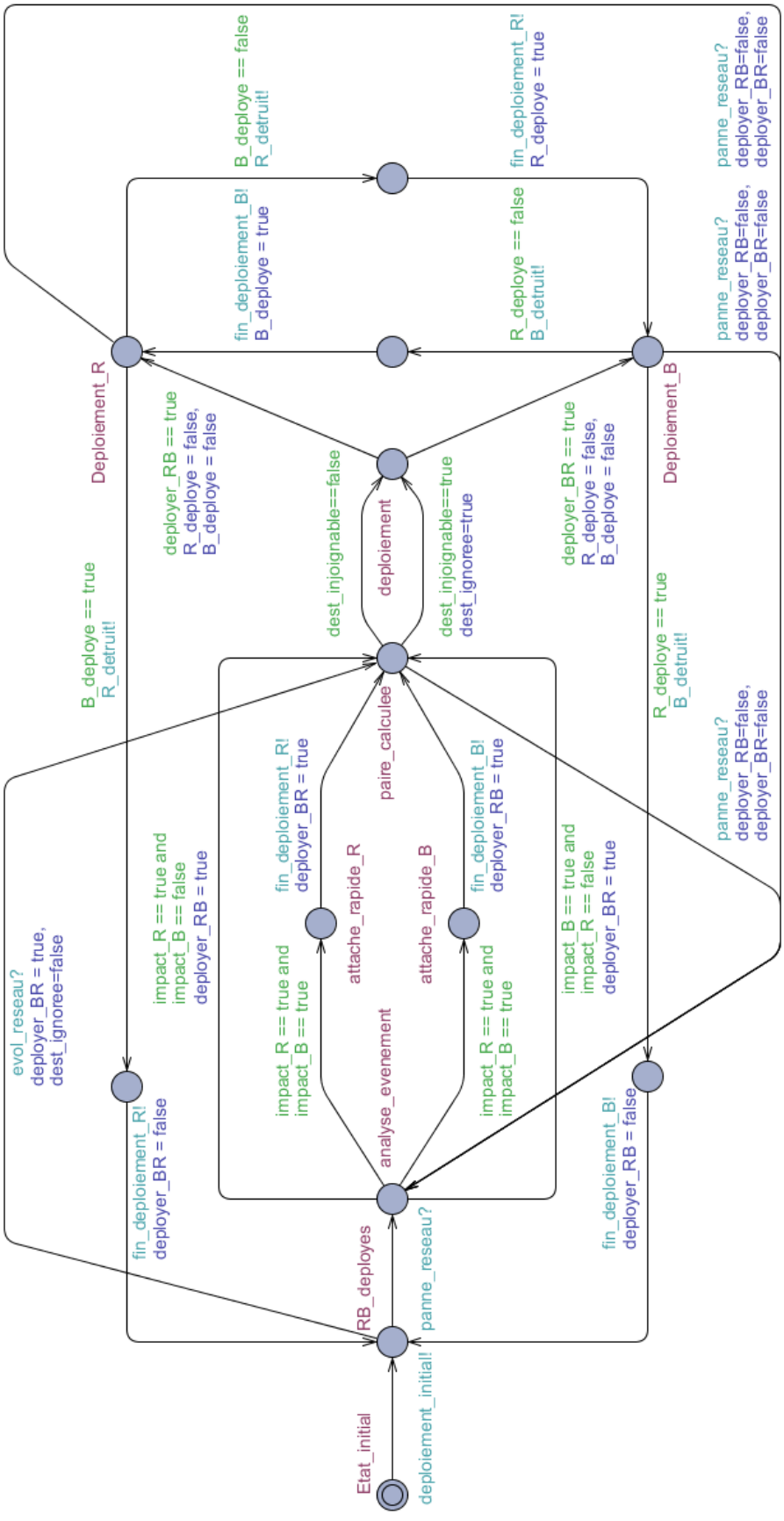


FIGURE A.1 – Patron du plan de contrôle sur UPPAAL

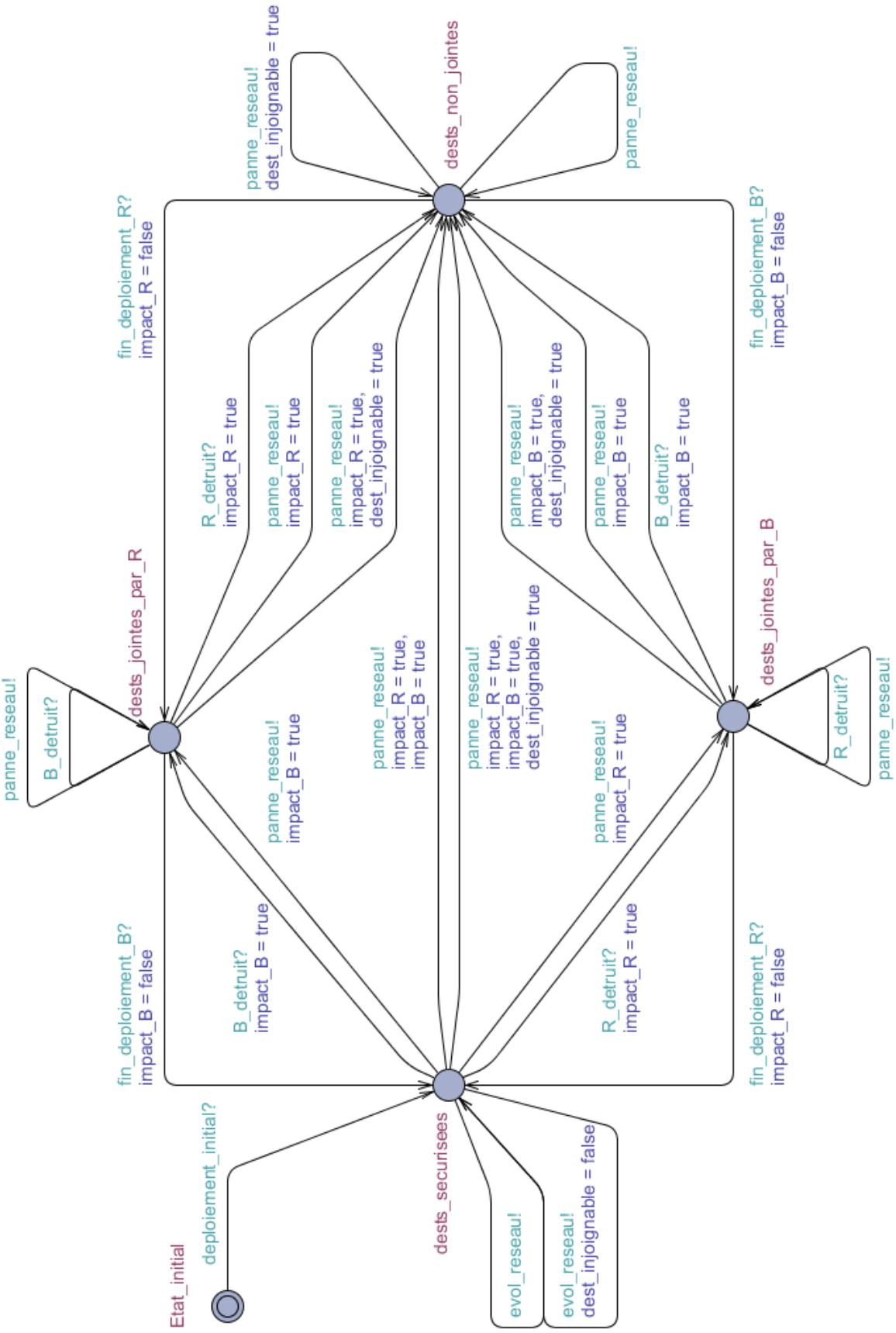


FIGURE A.2 – Patron du plan de données sur UPPAAL

A.4 Vérification

Uppaal permet de vérifier les modèles, c'est-à-dire de définir des propriétés sur les transitions et les états qui doivent être vérifiées. Uppaal dispose d'une syntaxe spécifique. Les propriétés peuvent s'écrire de cinq manières différentes :

- possibilité : `E<>` *Expression*. Il existe un ensemble de transitions amenant le système à un état vérifiant l'expression. En d'autres termes, le système peut à au moins un instant vérifier l'expression.
- invariance potentielle : `E[]` *Expression*. Il existe au moins un ensemble de transitions dont tous les états traversés vérifient l'expression. Autrement dit, il existe un déroulement dans lequel le système respecte toujours la proposition.
- inévitabilité : `A<>` *Expression*. Quel que soit l'ensemble de transition suivi par le système, le système arrive à un état vérifiant l'expression.
- invariance : `A[]` *Expression*. Quel que soit l'ensemble de transition suivi par le système, tous les états traversés vérifient l'expression. Une telle propriété est vérifiée à tout instant par le système.
- implication : *Expression 1* `-->` *Expression 2*. Si l'expression 1 est vraie, alors l'expression 2 est vraie.

Ces propriétés commencent par une double quantification chemin/états : les lettres `E` et `A` caractérisent respectivement l'existence $\exists$ et l'universalité $\forall$ de chemins, et les symboles `<>` ou `[]` caractérisent respectivement l'existence $\exists$ et l'universalité $\forall$ d'états.

On souhaite vérifier les propriétés suivantes :

- pas de blocage des automates : aucun ensemble de transitions n'amène le réseau d'automates à une situation où chacun des automates est bloqué dans un état. Autrement dit, le protocole Seamless Multicast n'est jamais bloqué.

```
A[] not deadlock == true
```

- sécurisation de l'état stable : si une nouvelle paire d'arbres a été déployée, le plan de données est bien dans un état sécurisé.

```
ctl.RB_deployes --> data.dests_securisees
```

- connexion rapide garantie : dans le cas où des destinations ne sont plus alimentées par aucun arbre, le protocole Seamless Multicast va bien rattacher rapidement l'un des deux arbres avant de converger vers une nouvelle paire. On vérifie pour cela que le système est dans l'un des états de rattachement, ou au moins en analyse de l'évènement, et que seuls ces trois états peuvent être atteints.

```
data.dests_non_jointes --> (ctl.attache_rapide_R
                            or ctl.attache_rapide_B
                            or ctl.analyse_evenement) == true
data.dests_non_jointes --> (ctl.attache_rapide_R
                            or ctl.attache_rapide_B)  == false
data.dests_non_jointes --> (ctl.analyse_evenement)    == false
```

Annexe B

Interface du contrôleur SDN

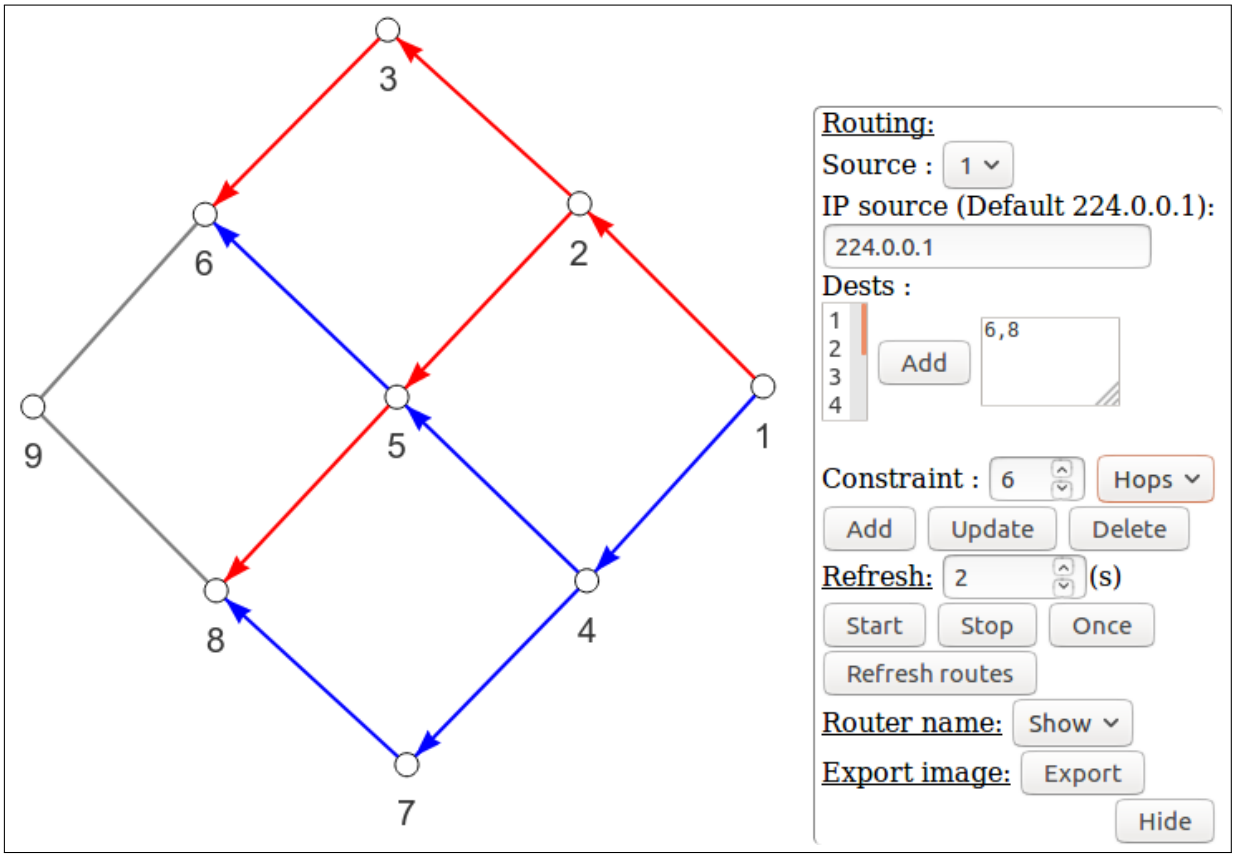


FIGURE B.1 – Illustration de l'interface du contrôleur développé

L'interface développée pour le contrôleur SDN est illustrée en figure B.1. La partie de gauche présente le réseau tel que vu par le contrôleur. Si des arbres sont déployées, ils sont également visibles. Le menu de droite permet de déployer des services, en choisissant un nœud portant une source, une adresse IP identifiant le service de multidiffusion, des nœuds portant les destinations et une contrainte en nombre de sauts ou en délai.

Glossaire

AG : « Algorithme Génétique », classe d'algorithme dans laquelle les solutions potentielles d'un problème sont traités comme des individus d'un processus d'évolution par générations, évalués par une fonction numérique

Arbre de multidiffusion : Ensemble des chemins utilisés sur un réseau pour une multidiffusion à partir d'une source dans le réseau. Un arbre ne comprend pas de cycle et chaque feuille n'est joignable que par un unique chemin

Bagotage : Dans le cas d'une liaison réseau, oscillation intermittente entre les états fonctionnel, dégradé, ou non-fonctionnel

BGP : « Border Gateway Protocol », protocole d'échange d'informations de routage entre des systèmes

CBF : « Constrained Bellman-Ford », algorithme d'exploration de graphe permettant de trouver le plus court chemin contraint en délai

Chemin : Ensemble de liens d'un graphe permettant de joindre une source à une destination

DMDT : « Delay-constrained Minimal Destination maximally-disjoint Trees », problème consistant à trouver dans un graphe la paire la moins coûteuse d'arbres redondants joignant une même source à un même ensemble de destinations, contraints en délai, les plus indépendants possible par destination

Flux audiovisuel : Suite de données dans le temps qui encode une information audio ou vidéo

Gigue : Variation de la latence au fil du temps

Heuristique : Méthode de calcul rapide fournissant une solution réalisable d'un problème, mais pas nécessairement un optimum

IGMP : « Internet Group Management Protocol », protocole IPv4 permettant à un routeur d'établir des groupes de clients de multidiffusion

IS : « Iterative SHERPA », heuristique de construction pour le problème DMDT

LARAC : « LAgrange Relaxation based Aggregated Cost », algorithme exploitant l'algorithme de Dijkstra itérativement afin de trouver le plus court chemin contraint en délai

Latence : Délai de transport entre la source et le destinataire d'un paquet

Multidiffusion : Communication d'un flux de données point-à-multipoint, exploitant la réplication des données (par opposition à de multiples diffusions)

MPBGP : « Multiprotocol Extensions for BGP », ensemble d'extensions au protocole BGP permettant notamment des ingénieries de multidiffusion

MVPN : « Multicast Virtual Private Network », ingénierie protocolaire permettant le transport de flux de multidiffusion sur un réseau, et l'abonnement de clients à ce flux

PIM : « Protocol-Independent Multicast », famille de protocoles IP permettant le transport de flux de multidiffusion sur des réseaux, exploitant les informations de routage de protocoles sous-jacents

QoE : « Quality of Experience », degré de satisfaction ou de contrariété d'un utilisateur d'une application ou d'un service

QoS : « Quality of Service », ensemble des caractéristiques d'un service permettant de répondre aux besoins et contraintes

Redondance : Utilisation d'éléments en plusieurs exemplaires dans un même système pour en assurer le fonctionnement en cas d'évènement impactant ou non

Résilience : Capacité d'un système à retourner à un état fonctionnel après un évènement impactant le système

Robustesse : Capacité d'un système à supporter des évènements, sans impact sur le fonctionnement

RTF : « Red Tree First », heuristique de construction pour le problème DMDT

SDN : « Software-Defined Networking », paradigme d'architecture réseau où plan de contrôle et plan de données sont portés par des entités séparées

SHERPA : « SHaring-Edges Restricted PAths », heuristique de construction permettant de trouver une paire de chemins contraints en délai, les plus indépendants possible par destination

SLA : « Service-Level Agreement », contrat définissant la qualité de service de la prestation d'un opérateur à un client

SNMP : « Simple Network Management Protocol », protocole de communication servant à gérer et superviser des équipements réseaux

SPOF : « Single Point Of Failure », point unique de panne, élément d'un système dont la panne met le système hors-service

TMS : Réseau principal opéré par TDF, dédié notamment au transport des chaînes de la TNT

TNT : « Télévision Numérique Terrestre », technologie de diffusion de signaux audiovisuels basé sur un réseau d'émetteurs Hertzien

Bibliographie

[1] ITU-T P.10/G.100, "Vocabulary for performance, quality of service and quality of experience," ITU, Tech. Rep., Nov. 2017.

[2] ITU-T E.800, "Terms and definitions related to quality of service and network performance including dependability," ITU-T, Tech. Rep., 2009.

[3] H. Rifai, S. Mohammed, and A. Mellouk, "A brief synthesis of QoS-QoE methodologies," in *2011 10th International Symposium on Programming and Systems*. Algiers, Algeria : IEEE, Apr. 2011, pp. 32–38.

[4] E. C. Rosen and R. Aggarwal, "RFC 6513 : Multicast in MPLS/BGP IP VPNs," IETF, RFC 6513, 2012.

[5] S. Pirlot, "Survivabilité dans les Réseaux de Transport de Vidéo et d'Audio sans Dégradation de la Qualité Perçue par l'Utilisateur," Thèse de doctorat, Université de Lorraine - ED IAEM, Nancy, France, 2016, dirigée par Lepage, Francis. [Online]. Available : http://www.theses.fr/2016LORR0082

[6] FRANCE, "Loi no. 2007-309 du 5 mars 2007 relative à la modernisation de la diffusion audiovisuelle et à la télévision du futur," in *Journal Officiel, no. 56*, Mar. 2007, p. 4347, NOR : MCCX0600104L.

[7] K. Obraczka, "Multicast Transport Protocols : a Survey and Taxonomy," *IEEE Communications Magazine*, vol. 36, no. 1, pp. 94–102, Jan. 1998.

[8] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, and L. Zheng, "RFC 7761 : Protocol Independent Multicast - Sparse Mode (PIM-SM) : Protocol Specification (Revised)," IETF, RFC 4760, 2016.

[9] S. Deering, "RFC 1112 : Host Extensions for IP Multicasting," IETF, RFC 1112, Aug. 1989.

[10] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "RFC 4760 : Multiprotocol Extensions for BGP-4," IETF, RFC 4760, 2007.

[11] A. Striegel and G. Manimaran, "A Survey of QoS Multicasting Issues," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 82–87, Jun. 2002.

[12] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-layer Multicast Protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[13] *ST 2022-7 :2013 : Seamless Protection Switching of SMPTE ST 2022 IP Datagrams*. The Society of Motion Picture and Television Engineers, 2013.

[14] A. Iselt, "A new synchronization algorithm for hitless protection switching in ATM networks," in *1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305)*. Scottsdale, AZ, USA : IEEE, 1999, pp. 370–376.

[15] A. Orda and A. Sprintson, "Efficient algorithms for computing disjoint QoS paths," vol. 1. IEEE, 2004, pp. 727–738.

[16] A. Gopalan and S. Ramasubramanian, "IP Fast Rerouting and Disjoint Multipath Routing With Three Edge-Independent Spanning Trees," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1336–1349, Jun. 2016.

[17] T. Hasegawa, K. Kamimura, H. Hoshino, S. Ano, and T. Hasegawa, "IP-based HDTV Broadcasting System Architecture with Non-stop Service Availability," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.* St. Louis, MO, USA : IEEE, 2005, p. 6 pp.

[18] P. Pan, G. Swallow, and A. Atlas, "RFC 4090 : Fast Reroute Extensions to RSVP-TE for LSP Tunnels," IETF, RFC 4090, 2005.

[19] A. Sundarrajan and S. Ramasubramanian, "Fast reroute from single link and single node failures for IP multicast," *Computer Networks*, vol. 82, pp. 20–33, May 2015.

[20] G.-m. Wei, C.-H. Lung, and A. Srinivasan, "Protecting a MPLS multicast session tree with bounded switchover time," in *SPECTS : 2010 International Symposium on Performance Evaluation of Computer and Telecommunication Systems : 11-14 July 2010.*, 2010, pp. 236–243.

[21] R. Forsati, M. Mahdavi, A. T. Haghighat, and A. Ghariniyat, "An efficient algorithm for bandwidth-delay constrained least cost multicast routing," in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on.* IEEE, 2008, pp. 001 641–001 646.

[22] G. Feng, "Delay constrained multicast routing : What can we learn from an exact approach ?" in *Global Communications Conference (GLOBECOM), 2012 IEEE.* IEEE, 2012, pp. 2809–2814.

[23] S. Kubler, J. Robert, J.-P. Georges, and E. Rondeau, "Dual path communications over multiple spanning trees for networked control systems," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1460–1470, Oct. 2012.

[24] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[25] Y. Bejerano and P. V. Koppol, "Optimal construction of redundant multicast trees in directed graphs," in *INFOCOM 2009, IEEE.* IEEE, 2009, pp. 2696–2700.

[26] Y. Bejerano, S. Jana, and P. V. Koppol, "Efficient Construction of Directed Redundant Steiner Trees," in *Local Computer Networks (LCN), 2012 IEEE 37th Conference on.* IEEE, 2012, pp. 119–127.

[27] Y. Bejerano and P. V. Koppol, "Link-coloring based scheme for multicast and unicast protection," in *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on.* IEEE, 2013, pp. 21–28.

[28] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[29] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, Apr. 1958.

[30] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2001, pp. 859–868.

[31] R. Widyono, *The design and evaluation of routing algorithms for real-time channels*. International Computer Science Institute Berkeley, 1994.

[32] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[33] E.-G. Talbi, *Metaheuristics : from design to implementation*. Hoboken, NJ : Wiley, 2009, oCLC : 230183356.

[34] R.-H. Hwang, W.-Y. Do, and S.-C. Yang, "Multicast routing based on genetic algorithms," *Journal of Information Science and Engineering*, vol. 16, no. 6, pp. 885–901, 2000.

[35] W. Zhengying, S. Bingxin, and Z. Erdun, "Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm," *Computer Communications*, vol. 24, no. 7-8, pp. 685–692, Apr. 2001.

[36] T. Lu and J. Zhu, "Genetic Algorithm for Energy-Efficient QoS Multicast Routing," *IEEE Communications Letters*, vol. 17, no. 1, pp. 31–34, Jan. 2013.

[37] S.-Y. Tseng, Y.-M. Huang, and C.-C. Lin, "Genetic algorithm for delay- and degree-constrained multimedia broadcasting on overlay networks," *Computer Communications*, vol. 29, no. 17, pp. 3625–3632, Nov. 2006.

[38] J. Knowles and D. Corne, "A new evolutionary approach to the degree-constrained minimum spanning tree problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 125–134, Jul. 2000.

[39] N. Shimamoto, A. Hiramatsu, and K. Yamasaki, "A dynamic routing control based on a genetic algorithm," in *IEEE International Conference on Neural Networks*. San Francisco, CA, USA : IEEE, 1993, pp. 1123–1128.

[40] B. Kazimipour, X. Li, and A. K. Qin, "A review of population initialization techniques for evolutionary algorithms," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. Beijing, China : IEEE, Jul. 2014, pp. 2585–2592.

[41] T. Friedrich and M. Wagner, "Seeding the initial population of multi-objective evolutionary algorithms : A computational study," *Applied Soft Computing*, vol. 33, pp. 223–230, Aug. 2015.

[42] A. E. Eiben and S. K. Smit, "Evolutionary Algorithm Parameters and Methods to Tune Them," in *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, pp. 15–36.

[43] J. Clune, S. Goings, B. Punch, and E. Goodman, "Investigations in meta-GAs : panaceas or pipe dreams ?" in *Proceedings of the 2005 workshops on Genetic and evolutionary computation - GECCO '05*. Washington, D.C. : ACM Press, 2005, p. 235.

[44] B. Yuan and M. Gallagher, "A Hybrid Approach to Parameter Tuning in Genetic Algorithms," in *2005 IEEE Congress on Evolutionary Computation*, vol. 2. Edinburgh, Scotland, UK : IEEE, 2005, pp. 1096–1103.

[45] L. Wang and T. Shen, "Improved adaptive genetic algorithm and its application to image segmentation," in *Image Extraction, Segmentation, and Recognition*, vol. 4550. International Society for Optics and Photonics, Sep. 2001, pp. 115–121.

[46] K. Almakadmeh and W. Alma'aitah, "Comparison of Crossover Types to Build Improved Queries Using Adaptive Genetic Algorithm," in *New Trends in Computing Sciences (ICTCS), 2017 International Conference on*. Amman, Jordan : IEEE, Oct. 2017, pp. 1–5.

[47] T.-P. Hong, H.-S. Wang, and W.-C. Chen, "Simultaneously applying multiple mutation operators in genetic algorithms," *Journal of heuristics*, vol. 6, no. 4, pp. 439–455, 2000.

[48] F. G. Lobo and C. F. Lima, "A review of adaptive population sizing schemes in genetic algorithms," in *Proceedings of the 2005 workshops on Genetic and evolutionary computation - GECCO '05*. Washington, D.C. : ACM Press, 2005, p. 228.

[49] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling," in *Federated Conference on Computer Science and Information Systems (FedCSIS 2013)*. Krakow, Poland : IEEE, 2013, pp. 371–376.

[50] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[51] "MATLAB Global Optimisation Toolbox," r2018a, The MathWorks, Natick, MA, USA.

[52] S. Denazis, E. Haleplidis, J. H. Salim, O. Koufopavlou, D. Meyer, and K. Pentikousis, "RFC 7426 : Software-Defined Networking (SDN) : Layers and Architecture Terminology," IETF, RFC 7426, 2015.

[53] G. Antichi, I. Castro, M. Chiesa, E. L. Fernandes, R. Lapeyrade, D. Kopp, J. H. Han, M. Bruyere, C. Dietzel, M. Gusat, A. W. Moore, P. Owezarski, S. Uhlig, and M. Canini, "ENDEAVOUR : A Scalable SDN Architecture For Real-World IXPs," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2553–2562, Nov. 2017.

[54] M. Bruyere, G. Antichi, E. L. Fernandes, R. Lapeyrade, S. Uhlig, P. Owezarski, A. W. Moore, and I. Castro, "Rethinking IXPs' Architecture in the Age of SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2667–2674, Dec. 2018.

[55] D. Gyllstrom, N. Braga, and J. Kurose, "Recovery from link failures in a Smart Grid communication network using OpenFlow," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. Venice, Italy : IEEE, Nov. 2014, pp. 254–259.

[56] N. Dorsch, F. Kurtz, F. Girke, and C. Wietfeld, "Enhanced Fast Failover for Software-Defined Smart Grid Communication Networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*. Washington, DC, USA : IEEE, Dec. 2016, pp. 1–6.

[57] C. A. Hasrouty, V. Autefage, C. Olariu, D. Magoni, and J. Murphy, "SDN-Driven Multicast Streams with Adaptive Bitrates for VoIP Conferences," in *2016 IEEE International Conference on Communications (ICC)*. Kuala Lumpur, Malaysia : IEEE, May 2016, pp. 1–7.

[58] S. Shukla, P. Ranjan, and K. Singh, "MCDC : Multicast routing leveraging SDN for Data Center networks," in *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*. Noida, India : IEEE, Jan. 2016, pp. 585–590.

[59] Q. Grandemange, Y. Bhujwalla, M. Gilson, O. Ferveur, and E. Gnaedinger, "An AS-level Approach to Network Traffic Analysis and Modelling," in *2017 IEEE International Conference on Communications (ICC)*. Paris, France : IEEE, May 2017, pp. 1–6.

[60] A. Aleroud and I. Alsmadi, "Identifying DoS Attacks on Software Defined Networks : A Relation Context Approach," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. Istanbul, Turkey : IEEE, Apr. 2016, pp. 853–857.

[61] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN Control : Survey, Taxonomy, and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[62] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow : Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[63] ITU-T I.361, "B-ISDN ATM Layer Specification," ITU, Tech. Rep., 1990.

[64] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band OpenFlow networks," in *9th International Conference on the Design of Reliable Communication Networks (DRCN)*, Budapest, Hungary, 2013.

[65] J. T. Araujo, R. Landa, R. G. Clegg, and G. Pavlou, "Software-Defined Network Support for Transport Resilience," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. Krakow, Poland : IEEE, May 2014, pp. 1–8.

[66] V. Padma and P. Yogesh, "Proactive Failure Recovery in OpenFlow Based Software Defined Networks," in *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*. Chennai, India : IEEE, Mar. 2015, pp. 1–6.

[67] W. Gu, X. Zhang, B. Gong, and L. Wang, "A Survey of Multicast in Software-Defined Networking," in *Proceedings of the 5th International Conference on Information Engineering for Mechanics and Materials*. Huhhot, Inner Mongolia : Atlantis Press, 2015.

[68] S. Islam, N. Muslim, and J. W. Atwood, "A Survey on Multicasting in Software-Defined Networking," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 355–387, 2018.

[69] A. Ghannami and C. Shao, "Efficient Fast Recovery Mechanism in Software-Defined Networks : Multipath routing approach," in *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. Barcelona, Spain : IEEE, Dec. 2016, pp. 432–435.

[70] D. Kotani, K. Suzuki, and H. Shimonishi, "A Design and Implementation of OpenFlow Controller Handling IP Multicast with Fast Tree Switching," in *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*. Izmir, Turkey : IEEE, Jul. 2012, pp. 60–67.

[71] V. Renganathan Raja, C.-H. Lung, A. Pandey, G.-m. Wei, and A. Srinivasan, "A subtree-based approach to failure detection and protection for multicast in SDN," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 7, pp. 682–700, Jul. 2016.

[72] X. Xiong and T. Chen, "MTM : A Reliable Multiple Trees Multicast for Data Center Network," in *2017 International Conference on Networking, Architecture, and Storage (NAS)*. Shenzhen, China : IEEE, Aug. 2017, pp. 1–7.

[73] A. Farrel, J.-P. Vasseur, and J. Ash, "RFC 4655 : A path computation element (PCE)-based architecture," IETF, RFC 4655, 2006.

[74] F. Paolucci, F. Cugini, A. Giorgetti, N. Sambo, and P. Castoldi, "A Survey on the Path Computation Element (PCE) Architecture," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1819–1841, 2013.

[75] "POX Controller." [Online]. Available : https://github.com/noxrepo/pox

[76] K. Phemius and M. Bouet, "Monitoring latency with OpenFlow," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*. Zurich, Switzerland : IEEE, Oct. 2013, pp. 122–125.

[77] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop : Rapid Prototyping for Software-Defined Networks," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10*. Monterey, California : ACM Press, 2010, pp. 1–6.

[78] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rjahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The Design and Implementation of Open vSwitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*. Oakland, CA : USENIX Association, 2015, pp. 117–130.

[79] B. Ioss and P. Lemaître, "A Review on Global Sensitivity Analysis Methods," in *Uncertainty Management in Simulation-Optimization of Complex Systems*, ser. Operations Research/Computer Science Interfaces Series, G. Dellino and C. Meloni, Eds. Boston, MA : Springer US, 2015, vol. 59, pp. 101–122.

[80] "UPPAAL 4.0." [Online]. Available : http://www.uppaal.org/

[81] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "UPPAAL 4.0," in *Third International Conference on the Quantitative Evaluation of Systems - (QEST'06)*. Riverside, CA : IEEE, 2006, pp. 125–126.

Résumé

Dans les réseaux de distribution de contenu, la Qualité d'Expérience (QoE) est principalement évaluée par un indicateur de performance : la continuité de service. C'est pourquoi la robustesse des réseaux est une problématique majeure pour les opérateurs. La société TDF opère un réseau traditionnel de transport de flux audiovisuels en temps réel à l'aide de protocoles de multidiffusion. Toute panne réseau provoque cependant un temps de convergence non-nul, qui implique des pertes et des impacts sur le contenu.

L'objectif des travaux de cette thèse est de définir une architecture pour inhiber les impacts sur le contenu lors des phases de cicatrisation du réseau. Cette architecture établit des paires d'arbres de multidiffusion redondants disjoints et dynamiques sur le réseau de transport. Les équipements de restitution pouvant exploiter cette redondance de chemins, il est possible de pallier à d'éventuelles pertes de paquet.

L'essentiel des travaux porte sur le développement et l'évaluation de différents algorithmes de calcul des arbres de distribution destinés à alimenter le moteur de routage de l'architecture. La spécification des protocoles et processus de mise en œuvre sur le réseau est également abordée. L'implémentation de l'ensemble s'appuie sur une architecture de type Software Defined Networking (SDN), dans laquelle un contrôleur centralisé exploite la connaissance des performances et de la planification en bande passante pour établir et maintenir les paires d'arbres de multidiffusion.

Mots-clés : Routage, Arbres disjoints, QoE, Seamless, Multicast, Flux audiovisuel temps-réel, SDN.

Abstract

In Content Delivery Networks (CDN), Quality of Experience (QoE) provides a major performance indicator that is continuity of service. As a consequence, network robustness has become a major concern for network operators. TDF operates a traditional transport network for real-time video and audio transport through multicast. Any failure on the network causes a recovery time implying loss and an impact in the content viewing.

This study's goal is to define an architecture preventing impact on the content during network healing time. This architecture computes and deploys redundant disjoint multicast trees on the transport network. As restitution equipments are able to exploit path diversity, packet loss consequences can be avoided.

This work's main part is the development and evaluation of different algorithm for the computation of distribution trees, as part of the routing element of the architecture. Protocol specification and deployment process are also considered. Implementation is based on a Software Defined Networking (SDN) architecture in which a central controller uses its knowledge of the performance and bandwidth allocation to compute and maintain pairs of multicast trees.

Keywords : Routing, Disjoint trees, QoE, Seamless, Multicast, Real-time audiovisual stream, SDN.