HAL
open science

RESOURCE ALLOCATION IN MULTI-DOMAIN WIRELESS SOFTWARE-DEFINED NETWORKS

Lunde Chen

To cite this version:

Lunde Chen. RESOURCE ALLOCATION IN MULTI-DOMAIN WIRELESS SOFTWARE-DEFINED NETWORKS. Networking and Internet Architecture [cs.NI]. Institut national des sciences appliquées de Toulouse, 2019. English. NNT : . tel-02128361v1

HAL Id: tel-02128361

https://laas.hal.science/tel-02128361v1

Submitted on 14 May 2019 (v1), last revised 23 Jul 2019 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *29/04/2019* **par :**

Lunde CHEN

RESOURCE ALLOCATION IN MULTI-DOMAIN WIRELESS SOFTWARE-DEFINED NETWORKS

JURY

Damien MAGONI	Professeur d'Université	Rapporteur
Thierry DIVOUX	Professeur d'Université	Rapporteur
Congduc PHAM	Professeur d'Université	Membre du Jury
Pascal BERTHOU	Maître de Conférences	Membre du Jury
Thierry GAYRAUD	Professeur d'Université	Directeur de thèse
Slim ABDELLATIF	Maître de Conférences	Directeur de thèse

École doctorale et spécialité :
EDSYS : Informatique 4200018
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur(s) de Thèse :
Thierry GAYRAUD et *Slim ABDELLATIF*
Rapporteurs :
Damien MAGONI et *Thierry DIVOUX*

Remerciements

Je tiens à exprimer toute ma reconnaissance à mes directeurs de thèse, Dr. Slim ABDELLATIF et Prof. Thierry GAYRAUD. Je les remercie de m'avoir encadré, orienté, aidé et conseillé, avec une patience et encouragement incroyable, avec leur niveau de connaissance dans le sujet de thèse mais aussi leur dedication pour la recherche. Durant ces 4 ans de thèse, ils m'ont aidé avec tout leur soutien à traverser des moments difficiles, surtout le début de ma thèse qui était compliqué. Je remercie Prof. Pascal BERTHOU pour avoir aidé à l'encadrement et pour son encouragement et son aide.

J'adresse mes sincères remerciements aux membres de jury, Prof.Damien MAGONI, Prof.Thierry DIVOUX et Prof. Congduc PHAM, pour leur commentaires et suggestions perspicaces.

Je suis reconnaissant à China Scholarship Council (CSC) et au CNRS pour m'avoir financé pendant ma thèse. Merci au LAAS et INSA de Toulouse pour m'avoir accueilli. Merci à mes amis et collègues qui sont tellement gentils et généreux.

Enfin, je remercie ma famille pour leur soutien, leur compréhension et leurs encouragements.

Résumé

La tendance à la numérisation de nombreux secteurs industriels tels que l'automobile, l'agriculture, les transports, la gestion urbaine, etc. révèle la nécessité de nouveaux usages des services de communication point-à-multipoint, tels que la fourniture massive de mises à jour logicielles et livraison fiable de messages d'alerte à la population, etc. D'un autre côté, la mise en logiciel des réseaux de nouvelle génération, avec notamment l'adoption croissante des réseaux définis par logiciel (SDN), apporte au réseau la flexibilité et les capacités de programmation permettant de prendre en charge des services de distribution point-à-multipoint de manière rentable.

Cette thèse contribue au problème général de la fourniture de services de communication point-à-multipoint avec des exigences de qualité de service (QoS) dans un réseau SDN multi-domaines. Il considère également que certains des domaines sont des réseaux multi-sauts sans-fil. Tout d'abord, une technique d'agrégation de topologie de domaine basée sur une arborescence de Steiner, combinée à un algorithme heuristique d'allocation de ressources, sont proposés pour prendre en charge des services point-à-multipoint couvrant plusieurs domaines. Ensuite, un service de découverte de topologie générique est proposé pour les réseaux multi-sauts sans-fil basés sur le SDN afin de permettre au contrôleur SDN de créer et de gérer une vue complète du réseau avec divers attributs de nœud et de liaison sans-fil. Le contrôleur peut alors exposer des vues personnalisées aux applications de contrôle du réseau, telles que, par exemple, l'application en charge de la fourniture de services point-à-multipoint sur un réseau multi-sauts sans-fil basé sur le paradigme SDN. Un algorithme basé sur la programmation linéaire en nombres entiers et un algorithme génétique sont également proposés pour l'allocation de liens virtuels point-à-multipoint sur un réseau sans-fil multi-radio, multi-canaux et multi-sauts basé sur SDN. Enfin, pour traiter le cas des services dynamiques point-à-multipoint, nous proposons un schéma de réallocation de ressources qui répond aux exigences changeantes tout en réduisant les interuptions de service.

Keywords : allocation de liens virtuels, réseaux définis par logiciel, communication point-à-multipoint, réseaux multi-domaines, réseaux multi-sauts sans-fil, qualité de service

Abstract

The movement towards the digitalization of many industry sectors such as automotive, agriculture, transportation, city management, etc. is revealing the need for novel usages of point-to-multipoint network delivery services, such as massive delivery of software updates to objects, secure and reliable delivery of alert messages to population, etc. On another side, the softwarization of next generation networks, with amongst, the increasing adoption of Software Defined Networks (SDN) is bringing to the network the flexibility and programming capabilities that enable the support of point-to-multipoint delivery services in an efficient and cost-effective way.

This PhD work contributes to the general problem of providing point-to-multipoint delivery services with Quality of Service (QoS) requirements in a multi-domain SDN network. It also considers that some of the domains are wireless multi hop networks. First, a Steiner tree based network domain topology aggregation combined with a resource allocation heuristic algorithm is proposed to support point-to-multipoint delivery services that span multiple domains. Then, a generic topology discovery service is proposed for SDN based wireless multi-hop networks to let the SDN controller build and maintain a comprehensive view of the network with various node and wireless link attributes. From there, customized views can be exposed by the controller to network control applications, as, for instance, the application in charge of provisioning point-to-multipoint services on a SDN based wireless multi-hop network. An Integer linear programming based algorithm and a genetic algorithm are also proposed for the embedding of point-to-multipoint services on a SDN based multi-radio, multi-channel and multi-hop wireless network. Last, to address the case of dynamic point-to-multipoint services, we propose a resource reallocation scheme that meets the changing requirements while reducing service disruption.

Keywords : virtual link embedding, software-defined networks, point-to-multipoint communication, multi-domain networks, wireless multi-hop networks, quality of service

Table des matières

Resume **iii**

Abstract **v**

1 Introduction **1**
1.1 Problem statement 2
1.2 Thesis contribution 3
1.3 Structure of the thesis 4

2 A Topology Discovery Service For Software Defined Wireless Multi-Hop Networks **7**
2.1 Introduction 7
2.2 Related works 8
2.3 Legacy topology discovery service for wired SDN networks 9
2.4 Limitation of existing topology discovery services for wireless SDN multi-hop networks 11
2.5 The proposed discovery scheme 12
2.5.1 Network representation at the controller 12
2.5.2 Topology construction and maintenance 17
2.6 Implementation and experimental results 21
2.6.1 Implementation on Open vSwitch and Ryu controller 21
2.6.2 Application and validation on an experimental testbed 22
2.6.3 Preliminary results 24
2.7 Conclusions 26

3 Resource Allocation for Virtual Link Embedding in Multi-hop Multi-radio Multi-channel Wireless SDN **27**
3.1 Introduction 27
3.2 Related works 28
3.3 Prerequisites and system model 29
3.3.1 Prerequisites 29
3.3.2 Network model 31
3.3.3 Interference model 32
3.3.4 Virtual links request model 32
3.4 ILP problem formulation 33
3.4.1 Resource-related assignment variables 33
3.4.2 Problem constraints 35
3.4.3 Objective function 36
3.5 Genetic Algorithm 37
3.5.1 General background 37

3.5.2 Encoding scheme 38
3.5.3 Initial population 38
3.5.4 Fitness function 39
3.5.5 Crossover schemes 41
3.5.6 Mutation schemes 43
3.5.7 Balanced resource allocation with dynamic link cost 43
3.6 Performance Evaluation 45
3.6.1 Heuristic algorithms for comparison 45
3.6.2 Network Model 46
3.6.3 Load Model 46
3.6.4 Simulation settings 46
3.6.5 Performance metrics 47
3.6.6 Performance results 48
3.6.7 Method selection : ILP vs GA 55
3.7 Conclusion 56

4 Re-embedding Schemes for Evolving Virtual Links in SDN Networks 59
4.1 Introduction 59
4.2 Related work 61
4.3 System model and problem statement 61
4.3.1 Substrate network model 61
4.3.2 Virtual network request model 61
4.3.3 Evolving virtual links request model 63
4.3.4 Virtual links re-embedding objective 63
4.4 ILP for evolving VLs problem 64
4.4.1 Resource-related assignment variables 64
4.4.2 Objective function and problem constraints 65
4.5 Genetic algorithm for evolving VLs problem 66
4.5.1 Encoding scheme 66
4.5.2 Initial population 67
4.5.3 Fitness function 68
4.5.4 Selection of parents and crossover scheme 69
4.5.5 Mutation scheme 69
4.6 Performance evaluation 69
4.6.1 Simulation settings 69
4.6.2 Performance metrics 72
4.6.3 Evaluation results 72
4.7 Conclusion 73

5 Topology Aggregation and Resource Allocation in Multi-Domain SDN Networks 75
5.1 Introduction 75
5.2 Related works 77

5.3 System model and notations 77
5.3.1 Multi-domain SDN architecture 77
5.3.2 System model and notations 79
5.3.3 Multi-domain requests model 81
5.4 Steiner tree construction 81
5.4.1 Integer Linear Programming for Steiner Tree Construction . 81
5.4.2 Shortest Path Heuristic for Steiner Tree Construction 82
5.4.3 Analyses of SPH-Steiner's properties 83
5.5 Resource allocation with Steiner tree as topology aggregation in multi-domain SDN 87
5.5.1 Key principles and main steps 88
5.5.2 Embedding without QoS constraints 89
5.5.3 Embedding with QoS constraints 90
5.6 Performance evaluation 93
5.6.1 Resource embedding without QoS constraints 94
5.6.2 Resource embedding with QoS constraints 96
5.7 Conclusions 100

6 Conclusions 103
6.1 Conclusions 103
6.2 Thesis technological scope 104
6.3 Perspectives 104
6.3.1 Topology discovery in wireless SDN 104
6.3.2 Resource allocation for virtual link embedding 104
6.3.3 Topology aggregation in multi-domain networks 105

Bibliographie 107

Table des figures

1.1 An illustration of multi-domain SDN with some of the domains being wireless ones. A point-to-multipoint communication is established across domains. 2

2.1 Main procedures for network topology construction in legacy SDN . 10
2.2 Illustration of the problem of multipoint links in wireless SDN 12
2.3 The proposed system architecture 17
2.4 LLDPDU content 19
2.5 Illustration of our implementation on OVS and Ryu 22
2.6 Experimental testbed 22
2.7 Out-of-band control 23
2.8 In-band control 23
2.9 Modified visualization with Ryu Topology Viewer showing RSSI information 24
2.10 Delay of OFDP messages from *packet-out* to *packet-in* for discovering two different links 25
2.11 OpenFlow control traffic throughput 25

3.1 An illustration of multicast advantage 29
3.2 An illustration of the substrate wireless SDN network, the conflict pairs between links and the “cliques" that are here represented by minimum ellipsoids covering midpoints of links. 31
3.3 Crossover of two parent trees to get a offspring 41
3.4 Mutation scheme I : break-down and reconnection 43
3.5 Mutation scheme II : channel mutation 43
3.6 Network model used in our performance evaluation 47
3.7 Clique utilization (all cliques, large cliques and small cliques) and acceptance rate with $\beta_1 = 1$ v.s $\beta_1 = 0$. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA. . 49
3.8 Clique utilization and acceptance rate with and without multicast advantage. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA. 50
3.9 Clique utilization balancing with ILP and GA. Arrival rate = 0.02. $\beta_3 = 15$. 51
3.10 Switch resource consumption of all nodes, computed with ILP, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$ 52
3.11 Switch resource consumption of all nodes, computed with GA, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$. 53
3.12 Group table consumption of ILP and GA. Arrival rate = 0.02. $\beta_2 = 5$, $\beta_3 = 15$. 53

3.13 Switch resource consumption with ILP and ILP-PS. The initial flow table size is set to 90. Arrival rate = 0.02. $\beta_2 = 5$, $\beta_3 = 15$. 54
3.14 Acceptance rate of the heuristic using 3 different link metrics as well as ILP, GA and ILP-PS, for different arrival rates. $\beta_2 = 5$, $\beta_3 = 15$. . 55
3.15 Clique load balancing of heuristic using the 3 different link metrics . 55
3.16 Switch resource balancing of heuristic using the 3 different link metrics 56
3.17 Acceptance rate of GA by varying number of generation and size of population, compared with ILP and ILP-PS. Arrival rate = 0.02 . . 56
3.18 Average Computation Time of each request of GA by varying number of generation and size of population, compared with ILP and ILP-PS. 57

4.1 Initial Embedding, with node A as the source, and node G as destination. The evolving of the request requires that node B and E should be added as destinations. 60
4.2 One possible solution : recomputation of a new tree which minimizes resource consumption, but this would introduce service disruption . 60
4.3 Another possible solution : a solution that is based on the existing embedding tree, which brings minimum reconfiguration overhead and minimizes service disruptions 61
4.4 Acceptance rate comparison 73
4.5 Reconfiguration cost accumulated in the time comparison 73
4.6 Resource consumption accumulated in the time comparison 74

5.1 Illustration of a two level-hierarchy of controllers for multi-domain networking 79
5.2 Overview of the multi-domain virtual embedding process : (a) Global physical multi-domain network ; (b) Domains' abstracted exposed topologies and the corresponding global abstract view ; (c) Embedding in each domain. 80
5.3 Coefficient of variance for total costs calculated with SPH-Steiner from different starting points 85
5.4 SPH-Steiner from multiple staring points : increase of total costs . . 86
5.5 Full mesh v.s. Steiner tree as topology aggregation for simulated topologies : performance degradation 86
5.6 Illustration of one domain's topology aggregation without QoS constraints. For the purpose of clarity, only link cost of links on the Steiner tree is shown. 90
5.7 Global view and global tree without QoS constraints 91
5.8 Illustration of one domain's topology aggregation with QoS constraints. For the purpose of clarity, only link residual bandwidth of links on the Steiner tree is shown. 92
5.9 Global view and global tree with QoS constraints 94
5.10 Full mesh v.s. Steiner tree as topology aggregation for simulated topologies : computation time 95

5.11 Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from ESNET 95
5.12 Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from GEANT 95
5.13 Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from INTERNET2 97
5.14 BW balancing comparison of ILP-Global v.s. Steiner-Aggregation . . 99
5.15 SW balancing comparison of ILP-Global v.s. Steiner-Aggregation . . 99
5.16 ac normal comparison of ILP-Global v.s. Steiner-Aggregation 100

Liste des tableaux

2.1 Metrics classifications 13
2.2 Summary for Topology Representations and Required Node & Link Attributes 16

3.1 Classification of virtual link resource allocation schemes for wireless multi-hop networks 30

4.1 Classification of virtual link re-embedding schemes in the literature . 62

5.1 Classification of virtual link embedding schemes for multi-domain networks 78

6.1 Thesis Technological Scope 106

Chapitre 1

Introduction

In application scenarios where a very large number of users or devices consume the same data or content, the use of point-to-multipoint communications, with one single transmission delivered to multiple devices instead of multiple transmissions on a per device basis, provides significant gains in network resource consumption, hence enabling cost effective broadcast/multicast network delivery services.

Up to recently, live TV or live audio/video media distribution were considered as the main use cases for point-to-multipoint transmissions. The movement of network generations from the provision of a general purpose connectivity service towards a broader range of services, with diverse requirements, that address the needs of various industry sectors such as as entertainment, automotive, IoT, public warning, health care, etc. paves the way for new and prominent point-to-multipoint communications use cases [Gomez-Barquero 2018]. To name a few, in addition to the distribution of high quality immersive audio-visual media content formats such as ultra-high definition television, 360-degree video, virtual and augmented reality (VR/AR), point-to-multipoint transmissions are envisioned in the IoT industry sector as an enabler for massive delivery of software and operating system updates to a large number of similar connected devices. In the automotive sector, it is considered for the delivery of media content and software updates in the vehicles but also for the reliable, potentially strict real-time delivery of vehicular information (e.g. road safety, driving assistance and autonomous driving relevant information). Also, in public warning systems, it is considered for the secure and reliable delivery of alert messages to general population. With all these application opportunities, point-to-multipoint transmissions are considered as a key technology enabler for 5G, and it is generally recognized that an enhanced form of Multimedia Broadcast Multicast Service (MBMS) [3GPP 2004] required for 5G. More precisely, 5G envisions to incorporate point-to-multipoint capabilities as built-in delivery features from the outset, integrating point-to-point and point-to-multipoint modes under one common framework and enabling a dynamic use of point-to-multipoint to maximize network resource usage and spectrum efficiency [Wu 2018] .

The progressive advent of Software Defined Networking (SDN) in operator networks and entreprise networks can bring a noticeable boost to the emergence of effective point-to-multipoint delivery services. The "logical" centralized control based on a thorough visibility of the network, combined with the fine-grained and programmable selection and forwarding treatments of flows in the network give the opportunity to devise novel effective resource allocation algorithms that optimize network resource consumption. QoS (Quality of Service) requirements can also be

taken into account, as well as services with varying characteristics. Clearly, SDN can play a key role to reduce point-to-multipoint service costs, extend their capabilities and broaden their scope of applications.

1.1 Problem statement

This research work addresses the general problem of providing point-to-multipoint delivery services with QoS requirements in heterogeneous multi-domain SDN networks. In this context, each SDN domain is under a distinct authoritative entity, and deploys its own SDN controller to control its forwarding devices according to its management policy.

In this work, we assume the architecture presented in Figure 1.1, and promoted by the Open Networking Foundation (ONF) [ONF 2016] and Internet Engineering Task Force (IETF) [Geng 2018] with a master controller on top of the domains' controllers. The master controller acts as an orchestrator when dealing with point-to-multipoint service establishment requests. It derives the SDN domains that are eligible to support the request and instructs their SDN controllers to provision their associated sub-requests.

One important challenge facing the provision of point-to-multipoint services relates to the multi-domain context with the scalability issues that it raises (particularly at the master controller), and the crucial need of each domain to hide its detailed topology, and, in place, disclose a topology abstraction with sufficient useful details that let the master controller select near-optimal data paths. *Finding a domain topology abstraction suited for point-to-multipoint services with accompanying scalable algorithms that are capable of computing data-paths that nearly minimize network resource consumption is one of the research problem addressed in this work.*

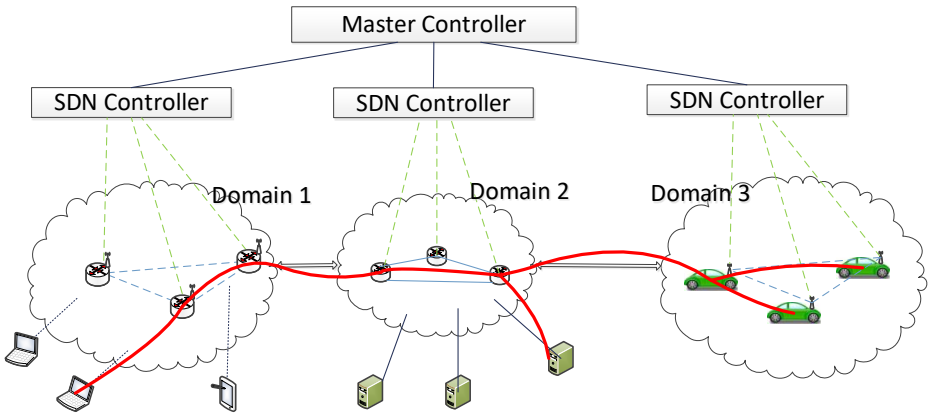

FIGURE 1.1 – An illustration of multi-domain SDN with some of the domains being wireless ones. A point-to-multipoint communication is established across domains.

From the multi-domain point-to-multipoint service request, selected SDN domains are instructed to embed (compute the data-paths and the required network

resources) their associated sub-request which can be point-to-point or point-to-multipoint. Network resource embedding for point-to-point services has been widely researched in the literature and particularly for wired networks. In this work, we consider the case of SDN based multi-radio multi-channel multi-hop wireless networks as one possible type of SDN domains, which find their main applications in vehicular networking, IoT, military and emergency response scenarios. In such networks, interference due to the vicinity of nodes and the shared and broadcast nature of the wireless medium have a significant impact on the overall network performance. Moreover, network nodes may be resource constrained and they may exhibit limited switching resources (i.e. limited number of forwarding/flow table entries). So the other research problem addressed in this work is : *how to support point-to-multipoint services on an SDN based wireless multi-hop network effectively, by considering the specificity of wireless transmissions and the switching resource limitation of SDN.* This entails : (1) defining the procedures and the logic that allow the SDN controller to collect, build and track the network map, i.e. the wireless network topology complemented with information related to wireless nodes attributes as well as wireless links attributes and performance metrics ; and (2) devising resource embedding algorithms that allow the SDN controller to map point-to-multipoint services with QoS requirements on the network map discovered and maintained by the controller.

The point-to-multipoint delivery service is in fact expressed as a point-to-multipoint virtual link that originates from a host belonging to a first domain and ends at multiple hosts (at least two) typically belonging to other domains. A bandwidth requirement is associated to the virtual link. Usually, and as considered in this work, virtual links are combined to form an overlay network, which represents the service provided to an application to meet all its communication requirements. Application requirements may change over time : new destinations may be added to or removed from a point-to-multipoint virtual link. Also, the bandwidth requirement of a virtual link may change over time. For an already deployed virtual link, a change in its characteristics may require changing a substantial part of its supporting data-paths in order to optimize network resource usage. However, this comes at the cost of a service disruption, the time the re-embedding of the virtual link is accomplished. Obviously, this may be painful, if not acceptable, for some applications. So another research problem addressed in this PhD thesis is to investigate a re-embedding strategy that reduces service disruptions.

1.2 Thesis contribution

This PhD work contributes to the support of point-to-multipoint end-to-end virtual links, potentially dynamic, and with QoS requirements on a multi-domain SDN based network, with some of the domains being wireless multi-hop networks. The proposed contributions are possible answers and solutions to the research problems described above and covers complementary aspects of the considered problem.

First, a Steiner tree based approach is proposed to effectively support point-to-

multipoint virtual links with and without QoS requirements on multi-domain SDN networks. Steiner trees are used for network domain topology aggregation, in other words, each SDN domain discloses a Steiner tree that includes its border nodes but also, depending on its traffic engineering policy, some other internal nodes. In comparison to the de facto full mesh topology aggregation, our proposed Steiner tree based topology aggregation leads to a more compact representation of the aggregated multi-domain network while being suited for computing Steiner/routing trees in a reduced time. We also propose a shortest path heuristic for the computation of approximate Steiner trees for topology aggregation but also for the computation of the global multi-domain Steiner tree on which the point-to-multipoint virtual link is established. Our experimental results show that the computational complexity gains and scalability gains brought by our approach comes with a limited loss of accuracy (optimality) in comparison to the optimal Steiner trees.

The de facto topology discovery service implemented on existing SDN controllers was designed for wired networks. As a consequence, it lacks many features required for the wireless context. So, the second contribution of this PhD work defines a generic topology discovery service for wireless multi-hop networks which relies on a comprehensive network topology representation with a rich set of links, ports and nodes attributes.This representation allows the controller to expose customized network views that meet the expectations of various SDN network control applications. This contribution defines the network representation built and maintained at the controller as well as the procedures and mechanisms used to establish and maintain the representation with reduced overhead.

Third, we propose two resource allocation algorithms for the provision of an overlay network (composed of point-to-point and point-to-multipoint end-to-end virtual links with bandwidth requirements) in software-defined multi-radio multi-channel wireless multi-hop networks : an Integer Linear Programming (ILP) based algorithm and a genetic algorithm. Both consider the specificities of wireless transmissions, namely the multipoint nature of wireless links which can be leveraged for point-to-multipoint links resource allocations, and, the interference between surrounding wireless links. They also consider switching resource consumption.

Last, we propose an ILP-based re-embedding algorithm to cope with dynamic overlay networks. In contrast to embedding algorithms whose essential focus is on reducing network resource consumption, the proposed re-embedding algorithms also considers and minimize the impact or the cost of a redeployment that we express as a function of the amount of node configurations that are needed to cancel the old data paths and set up the new ones. Reducing the number of configurations, indirectly reduces the number of disrupted VLs and their disruption delays.

1.3 Structure of the thesis

This dissertation is organized as follows. Chapter 2 presents the topology discovery service that we propose for SDN based wireless multi-hop network. Then,

Chapter 3 describes the ILP and genetic algorithms that we propose for embedding virtual links on such networks. Chapter 4 presents the re-embedding of virtual links when network requirements change, always using ILP and genetic algorithms. In Chapter 5, we propose an approach to efficiently support multipoint communications in a multi-domain context, where each domain exposes a synthetic and aggregated view of its network with Steiner-tree based topology aggregation. Finally, the general conclusion of this thesis is presented and research perspectives are proposed in Chapter 6.

Chapitre 2

A Topology Discovery Service For Software Defined Wireless Multi-Hop Networks

In this chapter, we present a topology discovery service for software-defined wireless multi-hop network, which is capable of capturing rich information about the network topology, such as link quality, interference and node characteristics etc., to effectively support the requirements of higher-layer SDN applications such as routing, channel allocation and transmission power control with enhanced flexibility. We analyse the general topology representation required by such SDN applications and specify the procedures and mechanisms required at the controller and nodes to maintain this representation. A proof-of-concept implementation of our service on an SDN enabled multi-channel multi-interface wireless multi-hop network testbed shows the feasibility of our proposal.

2.1 Introduction

Software-Defined Networking (SDN) has recently emerged as a new paradigm with the idea of taking control plane functions out of network forwarding devices and relocating them on remote external computer machines. The SDN architecture introduces a new entity, called the controller, that interfaces between the network control functions (or network control applications) and the network devices. It typically exposes, on the one hand, network or topology discovery services which provide these latter applications with the appropriate view of the network, and, on the other hand, some interfaces that let network control applications program remotely the forwarding devices.

Network and topology discovery services are essential to SDN applications. They require that the controller maintains a comprehensive view of the network (a network-wide view of forwarding devices, their intrinsic properties, and their status) from which it can derive different views, customized to each control application. The OpenFlow protocol, the main SDN enabler, provides some procedures to discover some of the nodes' attributes, but it doesn't specify how to proceed with link discovery and follow-up. To fill this gap, current SDN controllers such as Ryu[Ryu 2017], OpenDaylight[OpenDaylight 2017] etc. implement OFDP (OpenFlow Discovery Protocol) [OFDP 2017], which leverage LLDP (Layer Discovery Protocol)[LLDP 2009] with subtle modifications to perform topology discovery in

an OpenFlow network, with wired networks as the main target. The resulting network representation that current SDN controllers typically build is limited to the network topology with few additional information related to nodes and their network interfaces. This is clearly not sufficient in a wireless SDN context since it fails in capturing crucial characteristics and specificities of wireless links (e.g. broadcast nature of wireless links, their intermediate and varying performance, etc.). For instance, such a network representation assumes that links either work well or do not work at all and, hence, fails in capturing the *intermediate* and *varying* performance of wireless links (as opposed to the 0-1 binary representation of connectivity between wireless nodes).

In this chapter, we propose a new topology discovery scheme that is able to support effectively the various needs of the network control applications that may be used in a multi-hop wireless SDN context. The proposed scheme allows the controller to keep a comprehensive, updated and configurable view of the wireless multi-hop network with reduced overhead. The proposed service was implemented on Open vSwitch (OVS) enabled nodes and the Ryu controller and experimented on a wireless multi-interface multi-hop network testbed.

This chapter is organised as follows. Section 2.2 presents existing work related to the topology discovery in a wired and wireless SDN context. Section 2.3 presents legacy topology discovery service for wired SDN networks. Section 2.4 presents the limitations of existing topology discovery services for wireless SDN multi-hop networks. Sections 2.5 and 2.6 respectively present the proposed scheme and a prototype set-up with some experimental results. Section 2.7 concludes the chapter.

2.2 Related works

The network representation built by a typical network topology discovery service gathers information on nodes and links characteristics. Some of these are rather static, i.e. either they do not change over time (e.g. the number of network interfaces, the management address). Some other are dynamic meaning that they may be changing over time, this is for example the case of the wireless link state, the state of network interfaces (active/inactive), etc.. Existing topology discovery services implemented in many SDN/OpenFlow controllers (Ryu, Opendaylight, etc.) rely on many procedures to collect these informations. Nodes related attributes are collected using the OpenFlow protocol : the OpenFlow *Feature Request/Reply* and *port description* for static attributes and the OpenFlow *EchoRequest / EchoReply* and *Port_Status* for the dynamic attributes. For link discovery and tracking, the OFDP (OpenFlow Discovery Protocol) [OFDP 2017] is used by current SDN controllers with wired networks as the implicitly targeted network.

OFDP leverages the Link Layer Discovery Protocol (LLDP) with subtle modifications to perform link discovery in an OpenFlow network. As an OpenFlow switch cannot by itself send, receive and process LLDP messages, the controller sends periodically a separate *packet-out* message with a dedicated LLDP packet for

each port of each switch to discover their one-hop-away neighbours. By doing so, the controller discovers all available links in the network.

Adapting OFDP directly for a wireless SDN context, however, has several limitations. First of all, a wireless interface in one node may be connected to several interfaces of other nodes, unlike in a wired network. With the legacy OFDP based topology discovery service of common controllers, physical multipoint links are not considered. Secondly, adapting OFDP directly results in a 0-1 binary representation of links which is far from enough in a wireless context.

In fact, state-of-art work on multi-hop wireless SDN discovers the topology by exchanging broadcast packets between nodes to learn their one-hop neighbours, then upload such information to the SDN controller [Ku 2014] [Dely 2011] [Peng 2015]. Most, implicitly assume a 0-1 binary representation of wireless links.

More interestingly, in [Galluccio 2015], a topology management layer is proposed which abstracts the network resources. It can gather local information from nodes RSSI (Receiver Signal Strength Indicator), distance to the nearest sink node and battery level) and control their behaviour at all layers, according to the indications provided by the controllers. In [De Oliveira 2015], a TinyOS-based SDN framework that enables multiple controllers within the WSN, is proposed to use beacon packets to measure link quality and build neighbourhood table, which is then sent to the SDN controller. Two methods for link quality measurement are implemented : Link Estimator[Fonseca 2007] and RSSI. However, even with such information collected, much knowledge about the network is still missing, as more sophisticated link metrics are needed for various network applications such as QoS routing, channel assignment and topology control etc.

2.3 Legacy topology discovery service for wired SDN networks

The network representation built by existing network topology discovery services gathers information on nodes, ports and links characteristics. Some are static, i.e. they do not change during operation (e.g. the number of network interfaces, the management address, etc.) and are discovered when the node joins the network. Some others are dynamic (i.e. the port state : active/inactive, etc.) and hence need to be discovered and then monitored. Existing topology discovery services implemented in many SDN/OpenFlow controllers rely on complementary procedures to collect such information. Nodes and port related attributes discovery is performed by the OpenFlow protocol : during OpenFlow node-to-controller session initiation for attribute discovery, and then continuously during the session lifetime for dynamic attributes monitoring. In existing SDN controllers, link discovery and monitoring is ensured by the OFDP which, as explained below, is an adaptation of the Link Layer Discovery Protocol (LLDP) (used by LAN devices to advertise their identity and capabilities to their surrounding neighbours) to the SDN context.

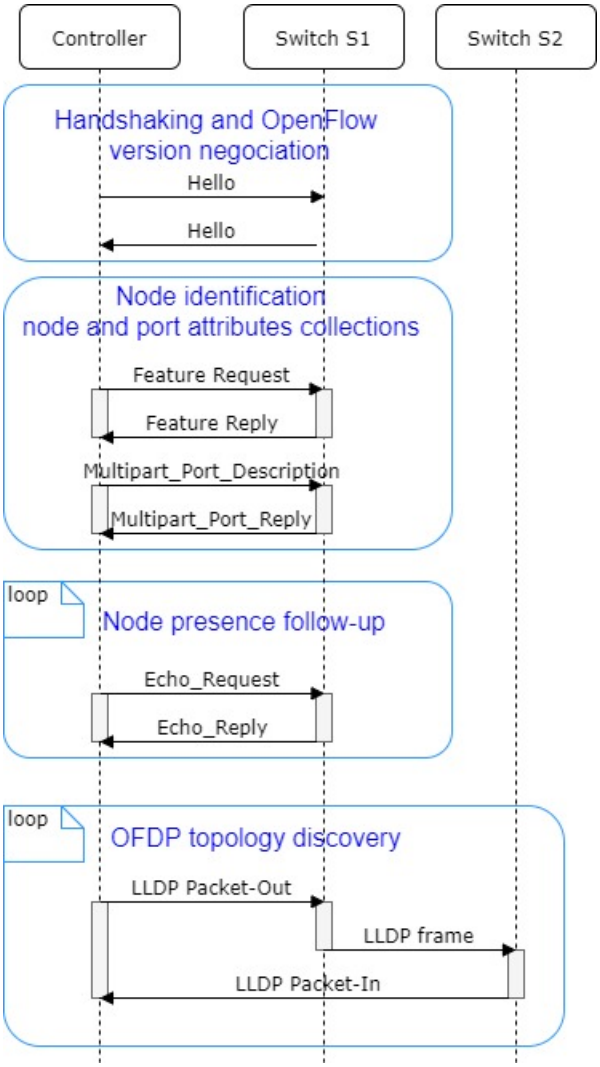

FIGURE 2.1 – Main procedures for network topology construction in legacy SDN

Figure 2.1 summarizes the OpenFlow and OFDP procedures that take place during a node-to-controller OpenFlow session and that contribute to the construction of the network representation at the controller. Notably,

— The OpenFlow "Feature Request/Reply" procedure initiated by the controller to identify the switch and collect its basic capabilities, complemented with the OpenFlow "Multipart_Port_Description Request/Reply" procedure which allows the controller to get the attributes of the ports belonging to the identified switch.

— The OpenFlow "Echo Request/Reply" procedure initiated by the controller or the node is used to let the controller follow up the presence of nodes. The OpenFlow "Port Status" notification mechanism is used by the node to inform the controller about changes in some dynamic port attributes.

— OFDP is used by existing controllers for link discovery and link state monitoring. OFDP is based on LLDP so far as OFDP packets are in fact LLDP frames encapsulated in OpenFlow messages, namely OpenFlow *packet-out* messages when flowing from the controller to the node and *packet-in* messages in the reverse direction. Periodically, the controller issues an OFDP packet destined to a node with the instruction of relaying the encapsulated LLDP packets on one of its active ports. Based on pre-installed OpenFlow rules that instruct nodes to forward LLDP packets to the controller, the receiving neighbor encapsulates the LLDP frame in an OpenFlow *packet-in*, saving the port on which the frame was received. Upon the reception of the *packet-in* message, the controller confirms the presence of a link connecting the above-cited nodes via the sending and receiving ports. A link is considered down, when no *packet-in* messages are received for some period of time or when a new *packet-in* message is received, meaning that the LLDP frame was received on a new port (that could belong to a new node).

2.4 Limitation of existing topology discovery services for wireless SDN multi-hop networks

Existing topology discovery services suffer from several limitations when applied to a wireless context. This is mainly due to the fact that they have been designed for wired networks. More precisely, many wireless node and wireless port attributes are missing. For instance, this is the case for node position, velocity and remaining energy, which could be useful for mobility management, geocasting or geographic routing. Also, some wireless ports attributes such as the operating channel, transmission power, supported physical modes, etc. are lacking and would help for channel assignment, routing, topology control network functions.

Moreover, OFDP provides an On-Off binary representation of links which is far from sufficient to capture the intermediate and varying performance of wireless links as required by many network functions and especially routing to select the best paths.

Finally, the topology discovery service implemented in common OpenFlow controllers assumes that links are point-to-point, which do not hold for wireless links where a transmission on an interface is received by all the nodes that sit in the radio coverage of the transmitting node. Referring back to the OFDP behavior from Section 2.3 and Figure 2.2, a LLDP frame leaving a wireless interface may be received by many nodes. As a consequence, many *packet-in* messages linked to the same originating LLDP frame are successively delivered to the controller. Each delivery is interpreted by the controller as a connectivity change with a newly valid link, which replaces the link discovered thanks to the previously received *packet-in* message. Hence, all the wireless links to which a wireless port belongs are not discovered. Instead, the controller notifies a succession of link changes when it should have discovered the presence of several links between each pair of nodes.

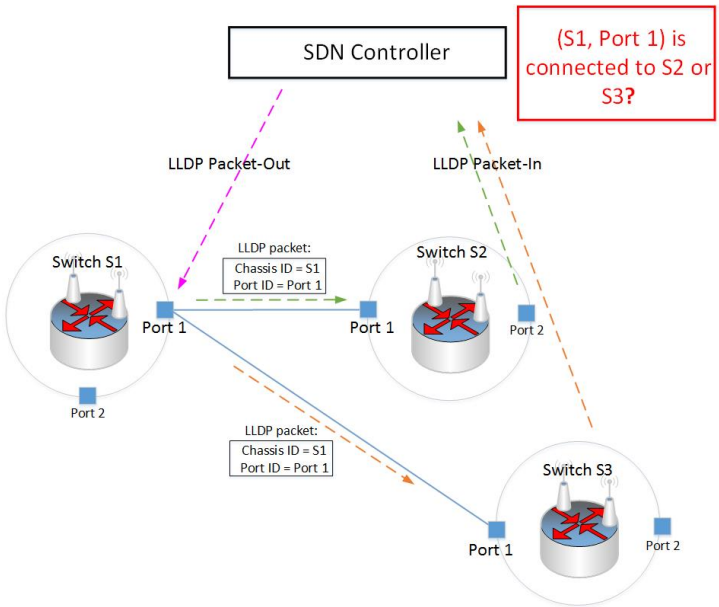

FIGURE 2.2 – Illustration of the problem of multipoint links in wireless SDN

2.5 The proposed discovery scheme

2.5.1 Network representation at the controller

Many network control functions will be using the network/topology discovery service, notably routing, channel assignment, topology control for interference mitigation or capacity improvement and mobility management. But, as stated above, these functions have different expectations on the network view that they get from the service. As a consequence, the network representation that the controller has to set up and maintain must cope with all these expectations. Below, we briefly and broadly describe the main requirements that has led to the network representation that we propose.

2.5.1.1 QoS routing

In a wireless multi-hop network, one of the most important and challenging functionality is routing. During the last decade, a plenty of routing metrics were proposed to capture the varying performance/quality of wireless links or paths in order to guide the selection of routes. Despite all the efforts, no general purpose link or path metric has gained complete adhesion from the research community. Instead, it was established that metrics should be chosen according to application requirements, network characteristics (mono or multi-interface, node mobility, energy constrained nodes, etc.) and the environment in which the network operates (Table 2.1 non exhaustively lists some of the most known metrics with their respective recommended usage scenario). As a consequence and as presented below,

TABLE 2.1 – Metrics classifications

Category	Consideration Dimensions	Metrics	Packet Delivery	PHY Info	Chanel Usage	Other Elements for Computation
Hardware-based	Radio strength	RSSI		RSSI		
	Radio strength, Noise floor	SINR[Neishaboori 2008]		SINR		
	Radio strength, Noise floor	SNR[Tan 2012]		SNR		
ETX-based	Radio strength, delivery ratio	rETT[Amusa 2011]	ETT	RSSI		
	Delivery ratio, intra-flow or inter-flow interference	EETT[Jiang 2007] WCETT[Draves 2004] MIC[Yang 2005]	ETT		Link Channel	
	Delivery ratio, interference-awareness	iAWARE[Subramanian 2006]	ETT	SNR, SINR		
Energy-based	Battery lifetime	MBCR[Toh 2001]				Remaining percentage battery capacity
	Battery lifetime, equalize the energy distribution	PEOF [Nurmio 2015]	ETX			Remaining energy
RTT-based	Delay	RTT [Adya 2004]				Unicast probe delay
	Delay	PktPair [Keshav 1991]				Delay difference of two unicast probe packets
Location-based	Delivery ratio, Interference	ALARM [Alotaibi 2008]	ETT		Link Chanel	Location of nodes
	Battery lifetime	ECGRID [Chao 2003]				Remaining energy, Location of nodes

our network representation supports multiple link metrics : (1) a set of elementary metrics that can be combined to derive most of state-of-the-art wireless link metrics (they are listed in Table 2.1 with the referring compound wireless link metric), and, (2) some of the most commonly used metrics, namely ETX [De Couto 2003] and ETT [Draves 2004]. When possible, it also includes some node attributes related to the available energy or the GPS position which is exploited by geographic routing algorithms.

2.5.1.2 Channel assignment

The problem of channel assignment is to assign a unique channel to each communication link in the network such that the interference is minimized [Qu 2016]. Channel assignment algorithms usually rely on conflict graphs [Jain 2003] to assign channels to wireless nodes. In fact, a conflict graph is a representation of the set of communication links that operate on the same channel (or frequency band) and interfere with each other, given an interference model. The wireless links are represented as vertices and, in case of interference between a pair of links, and edge is drawn, potentially, with a weight that reflects or quantify the level of interference (e.g. channel usage).

To meet the requirements of channel assignment network control applications, the channel (or frequency band) on which each network interface operates should be made available and hence included in our network representation. Regarding, the assessment of the level interference that each link causes to the surrounding links, we argue that the diverse wireless links metrics proposed for routing are enough exhaustive to meet channel assignment expectations.

2.5.1.3 Transmission power control

Topology control is the adjustment of node parameters and modes of operation so as to modify the topology of the network with the goal of improving its overall performance, network connectivity, coverage or extending its lifetime [Labrador 2010]. Transmission power control is at the heart of topology control. The general idea is to vary the transmission power of some selected nodes' interfaces or to turn them on or off to change the network topology [Labrador 2010]. In fact, transmission power control is an important technique to reduce energy consumption in wireless multi-hop networks. Moreover, in some circumstances, reducing the transmission power can mitigate interference. Transmission power control requires usually the knowledge of the transmission power of each node in the network, and the connectivity between them (called maximum power graph) and the associated link channel/frequency. Also, the locations of wireless nodes may be needed as in [Ramanathan 2000], as well as the received SINR [Qian 2009] [Chiang 2007]. More generally, link quality estimation could also be of help so as to preserve high-quality links and limit low-quality links.

2.5.1.4 Mobility management

The objective of mobility management is to maintain the connectivity of mobile nodes with a given level of performance and little if no interruption despite node movement. One challenge is to decide when a hand-off should be triggered and to select the appropriate forwarding nodes that will keep moving nodes connected. Signal strength measurements and their rate of variation are largely used to infer the stability of links and routes, such as associativity-based routing (ABR) [Toh 1997] and signal stability based adaptive routing (SSA) [Dube 1997]. GPS position, speed, direction and wireless link performance metrics are used to guide the selection of the forwarding nodes.

2.5.1.5 Summary of main identified requirements on topology representation

The requirements of the four presented network applications are quite representative of what network control applications may expect from a network discovery service. Table 2.2 lists the required topology representation for those four network applications. We show that these are expressed as nodes and link attributes to be collected by the SDN controller's topology discovery manager. Moreover, Figure 2.3 proposes one possible system architecture in which such nodes and link attributes are incorporated into the SDN topology discovery service.

TABLE 2.2 – Summary for Topology Representations and Required Node & Link Attributes

Wireless SDN Application	Typical Topology Representation	Required Dynamic Node Attributes	Required Dynamic Link Attributes
QoS Routing	Link Metric Graph	Remaining energy, localization, interference level	Link connectivity, packet delivery ratio, channel usage, RSSI, SINR, SNR
Channel Assignment	Conflict Graph		Link connectivity, channel usage of each link, link metric
Transmission Power Control	Maximum Power Graph	Localization, transmission power,	Link connectivity, channel usage of each link, SINR, link metric
Mobility Management	Connectivity Graph and localization	Localization (GPS), speed, direction	Link connectivity, RSSI

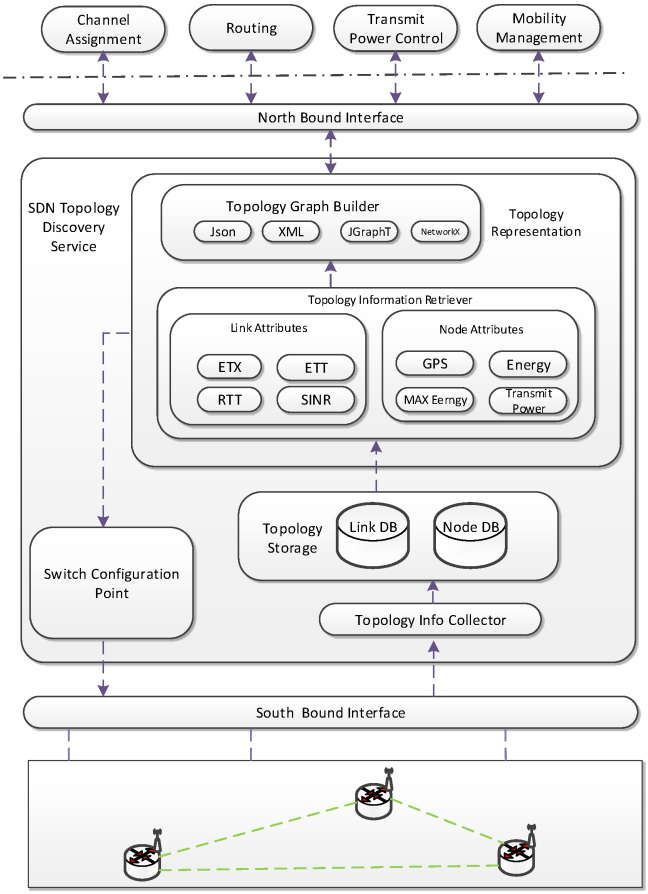

FIGURE 2.3 – The proposed system architecture

2.5.2 Topology construction and maintenance

Three design principles are at the heart of our proposed topology discovery service : (1) genericity in the sense that a plenty of nodes and link attributes can be made available to a variety of network control applications ; (2) configurability in the sense that the network topology that is built at the controller is configured and set according to the needs of network control applications that are running. From these needs, the network topology discovery service of the controller derives the attributes to collect. It consequently configures application agents running on the wireless nodes to proceed with local configurations as well as some measurements at nodes or wireless links to which nodes belong. The collected values are then sent to the controller following some procedures. (3) efficiency, in that the traffic overhead induced by the topology discovery is reduced.

The general behaviour of the proposed topology discovery service relies on and extends the *de facto* service implemented by many common SDN controllers for wired networks. It mainly proposes some modifications and extensions to make it work effectively for wireless multi-hop networks, namely the OpenFlow protocol

and OFDP.

2.5.2.1 Wireless node and port related attributes integration

The OpenFlow session establishment, which is initiated by the OpenFlow wireless node, goes through a sequence of steps : OpenFlow version negotiation, node and then port features discovery.
Most of the missing nodes attributes are dynamic and requires node to controller exchanges to keep these attributes up-to-date. Our proposal is to use the OpenFlow *EchoRequest / EchoReply* procedure to convey these node and port related attributes that change over time. These are symmetric messages and can be initiated by a node at will.

Other supplementary port attributes are discovered with the *OFPT_MULTIPART_REQUEST/REPLY* procedure which includes a *OFPMP_PORT_DESC* structure that gathers port characteristics. Since wireless ports are not yet considered by the OpenFlow 1.5.1 specification [OpenFlow 2015], following the same logic as for Ethernet ports, without loss of generality, we propose the WLAN port description properties data structure as depicted on Listing 2.1 to gather the missing port attributes presented in Table 2.2. The static attributes (i.e. not changing over time) are discovered following the procedure described above while the dynamic ones (i.e. changing over time) are maintained using the *OFPT_PORT_STATUS* procedure of the OpenFlow protocol.

,

Listing 2.1 – WiFi port description property

```
/* wifi port description property. */
struct ofp_port_desc_prop_wifi {
  uint16_t type;   /* OFPPDPT_WIFI. */
  uint16_t length; /* Length in bytes of the property. */
  uint8_t  pad[4]; /* Align to 64 bits. */

/* Bitmaps of OFPPF_* that describe features. All bits zeroed if
    unsupported or unavailable. */
  uint32_t channels_curr;     /* Current used channels */
  uint32_t channels_supported;  /* Supported channels */
  uint32_t phy_modes_curr;    /* Current used PHY modes*/
  uint32_t phy_modes_supported; /* Supported PHY modes */
  uint16_t antenna_sensitivity; /* antenna sensitivity*/
  uint16_t tx_pwr_min;          /* Minimum TX power */
  uint16_t tx_pwr_max;          /* Maximum TX power */
  uint16_t tx_pwr_curr;         /* Current TX power */
}
```

2.5.2.2 Wireless link metrics integration

OFDP messages are actually LLDP frames encapsulated in an OpenFlow *packet-out* message, generated by SDN controller. As shown in Figure 2.4, each LLDP frame

starts with the following mandatory TLVs : Chassis ID, Port ID, and Time-to-Live. The mandatory TLVs are followed by optional TLVs. The frame ends with a special TLV named end of LLDPDU.

As explained above, agents are configured and deployed on wireless nodes in order to measure the required wireless link performance metrics. The values collected at the nodes are then delivered to the controller. Our approach is to take advantage of the OFDP packets that are sent periodically for link discovery and maintenance by exploiting the optional TLVs of LLDP packets (one TLV per performance metric). When sending an OFDP message, the controller adds the optional TLVs related to the performance metrics to be updated. When receiving the OFDP message and parsing the optional TLVs, a node deduces the performance metrics that are actually needed by the controller and updates the TLVs with the last measured value and then transmits the OFDP packet back to the controller.

This procedure requires from the wireless nodes to process LLDP messages (parse and modify its fields). In practice, a standard Open vSwitch enabled node do not have this capability since only two operation modes with respect to LLDP messages have been implemented : *"LLDP disabled mode"* which is the default mode when OpenFlow is enabled, in which LLDP are simply forwarded by the node to the controller without any processing, and the *"LLDP enabled mode"* which corresponds to the standard use of LLDP where LLDP fields are generated by the switching nodes and processed by the receiving neighbouring node but never forwarded. For our proposed topology discovery service, we have defined a new intermediate operation mode, where switching nodes are not allowed to generate LLDP packets (this is the responsibility of the controller), but are allowed to process and forward LLDP packets.

FIGURE 2.4 – LLDPDU content

2.5.2.3 Multipoint wireless links

Concerning the multipoint wireless link issue, the problem comes from the controller logic. Indeed, when an OFDP *packet-in* is received, the controller extracts from the packet the source node, destination node and the sending and receiving ports. Then, it fetches from the list of existing links an entry that matches the source address and its outgoing port (regardless of the destination address and its port). In case of no match, a new link is added. If an entry is found, the destination address and port number are checked and if they do not coincide, the link is considered as obsolete and replaced with the new one. So our proposal is to include the destination address and receiving port in the match process and consider a link as obsolete in case of a predefined number of missing updates.

2.5.2.4 Configurability

Each running network control applications expresses the network view it needs with the associated attributes. For some dynamic attributes, it may also specify a refresh period or a condition that triggers the update. From all the needs of running network applications, the controller derives and updates the set of attributes to collect with their refresh periods or conditions. It consequently configures topology discovery agents deployed in wireless nodes to proceed with local configurations as well as node, port and link attributes measurements. These configurations can be performed by the OF-CONFIG protocol [OF-Config 2010] based on an attributes configuration data model in the YANG language. For example, the following three YANG data models can be used for modelling node attributes, link attributes and the configuration.

```
typedef node_attributes {
    description "enumeration of node attributes that could be
        collected"
    type enumeration {
        enum GPS;
        enum battery;
        enum velocity;
        enum transmission_power;
        enum CPU;
        enum memory;
    }
}
```

```
typedef link_attributes{
    description "enumeration of link attributes that could be
        collected"
    type enumeration{
        enum ETX;
        enum ETT;
        enum RSSI;
        enum SNR;
        enum SINR;
        enum channel;
        enum EAR;
        enum RTT;
        enum PktPair;
    }
}
```

```
container information_collection_configuration {
    config true;
    list collect_node_attributes{
        type node_attributes; }
    list collect_link_attributes {
        type link_attributes; }
    leaf report_interval_node_attributes { type int32; }
    leaf report_interval_link_attributes { type int32; }
```

```
}
```

2.5.2.5 Efficiency

It is crucial to reduce the control traffic in wireless networks in order to limit its impact on the performance experienced by user traffic. The global view of the network maintained at the controller sheds the light on topological properties of the network such as the level of importance of nodes or links with respect to the active routing paths and, hence, how severe the consequence is, if they fail or if some of their attributes are not fresh.

The main idea to reduce the control traffic is to adjust the distribution frequency of node/port (respectively link) related attributes according to the level of importance of the node (resp. the link). Depending on the SDN application and traffic transmission patterns, different centrality metrics (betweenness centrality, eigenvector centrality, PageRank) [Centrality 2018] can be used to measure the level of importance.

2.6 Implementation and experimental results

2.6.1 Implementation on Open vSwitch and Ryu controller

Our implementation applies to Open vSwitch (OVS) enabled wireless nodes and to the Ryu controller. Extensions to the source code of OVS aim at adding the new operation mode described in Section 2.5, which allows OVS to process and modify LLDP packets before sending them back to the controller. In the kernel space of OVS, each packet goes through the vport, datapath and then OpenFlow table. In the legacy "LLDP disabled mode" of OVS, if the packet is detected as an LLDP packet, it is sent to the controller via a *packet-in* message (shown by the path ①②③ in Figure 2.5). With our implementation, instead of sending the packet to the SDN controller, it is sent via netlink (which is used for upcall between OVS kernel and ovswitchd) to the userspace LLDP Agent (shown by the path ①②④). In the user space of OVS, the LLDP Agent, retrieves link metrics measurements from the “LinkMetric” Agent. The list of link metrics measured by the “LinkMetric” Agent is retrieved from OVSDB-server, which, as explained above, receives configuration from the Ryu controller via OFCONFIG. Link metrics measurements are added to the appropriate TLVs fields of the LLDP frame. Afterwards, this latter is encapsulated into an OpenFlow *packet-in* and transmitted to the Ryu controller (shown by ⑤). Ryu controller is modified in such a way that LLDP TLVs subtypes and field descriptions are added in its LLDP parser. On a LLDP *packet-in* reception, TLVs are parsed and link metrics are extracted and made available to the network representation maintained by the controller.

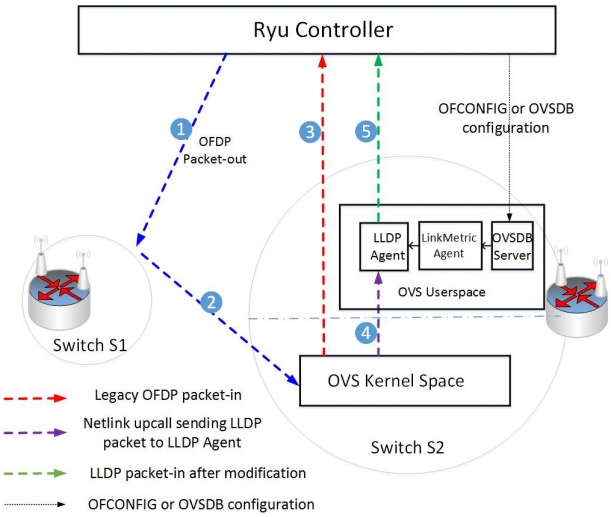

FIGURE 2.5 – Illustration of our implementation on OVS and Ryu

2.6.2 Application and validation on an experimental testbed

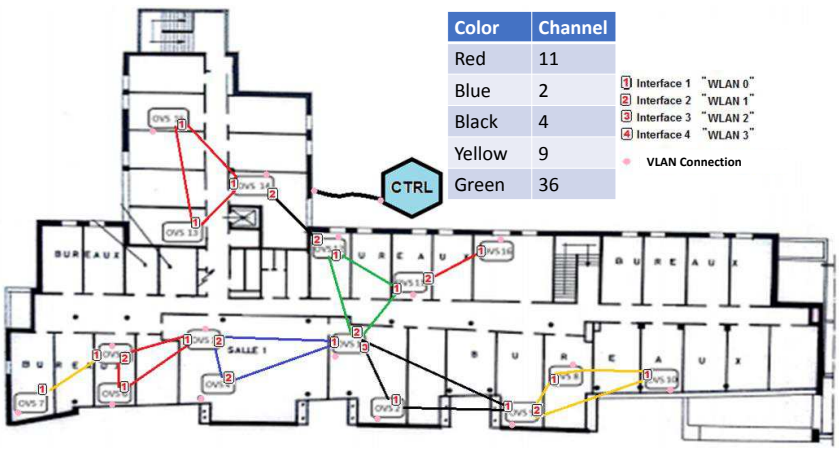

FIGURE 2.6 – Experimental testbed

A testbed has been setup to validate our proposals. It consists of 17 OpenWRT nodes (Bannana Pi and Avila gateworks) with one to three WiFi interfaces. The OpenFlow support has been added by embedding an OVS switch on each node. As shown in the Figure. 2.6, different WiFi channels are used to build a complex wireless ad'hoc network. Ryu and OVS are modified to support the LLDP-based topology discovery presented in the previous subsection. We consider two options to support the OpenFlow control traffic that is exchanged between wireless nodes and the OpenFlow controller.

2.6.2.1 Out-of-band control

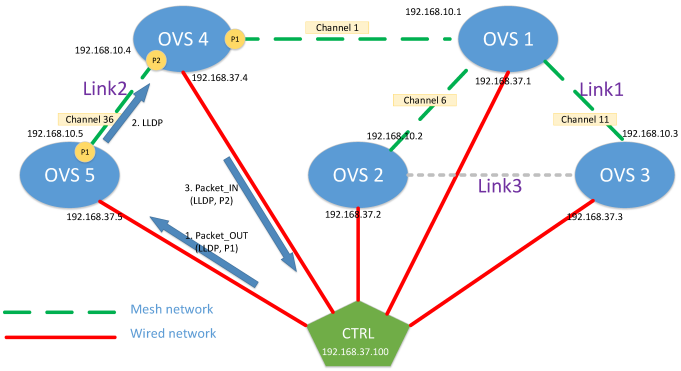

FIGURE 2.7 – Out-of-band control

When configured with out-of-band control, the testbed has two separate networks, one for OpenFlow control traffic, as it is shown in Figure 2.7 with the plain links, and the other for the multi-hop data network (dashed line links) that carries data traffic. The advantage of such a configuration is that the data traffic will not compete with the control traffic and will have no influence on the operation of the OpenFlow protocol and vice versa.

2.6.2.2 In-band control

In this configuration (as is shown in Figure 2.8), the control traffic and the data traffic share the same wireless transmission medium. In order to provide TCP level connectivity between wireless nodes and the controller (which is a prerequisite for OpenFlow connection establishment). Routes have to be pre-installed on wireless nodes. This can be achieved either statically by human, or dynamically using wireless ad'hoc network routing protocols. For simplicity, we use a static tree with the controller as the root to support the communication between the controller and wireless nodes.

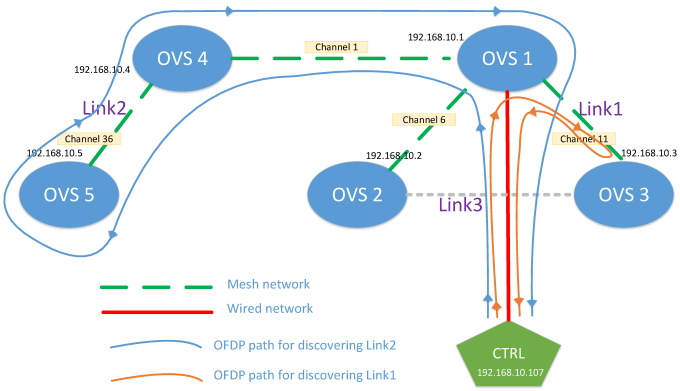

FIGURE 2.8 – In-band control

2.6.2.3 Topology visualization

In coherence with our proposed discovery scheme, the Topology Viewer of Ryu controller is also modified, in that link or node attributes collected via our LLDP based solution are also displayed. Figure 2.9 shows an example where RSSI of links are displayed. For the sake of clarity, only 5 nodes are shown.

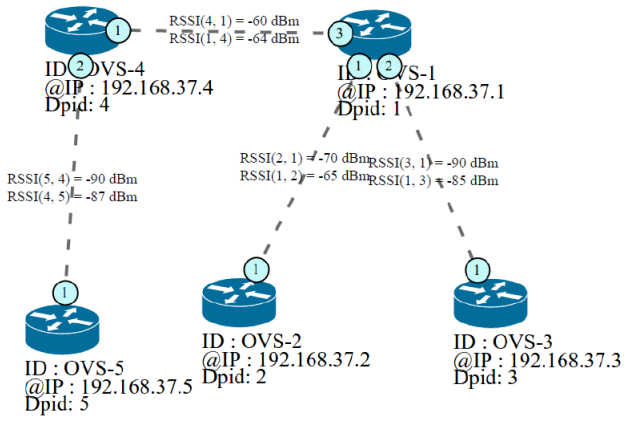

FIGURE 2.9 – Modified visualization with Ryu Topology Viewer showing RSSI information

2.6.3 Preliminary results

The main objective is to validate our implementation as well as to assess the performance of our wireless SDN testbed. Two experiments are carried out :

— The first experiment validates the expected difference in the link discovery time between an out-of-band setup and an in-band setup.

— The second experiment assesses the traffic overhead of OpenFlow messages brought in by topology discovery.

2.6.3.1 Link discovery time

In this experiment, we measure the delay between the *packet-out* and *packet-in* of an LLDP packet. For the sake of simplicity and easier understanding, only 5 nodes of the testbed are used, as is illustrated in in Figure 2.7 and Figure 2.8. The measurement is done at the Ryu controller, for both Link1 and Link2. Both out-of-band and in-band setup are considered and contrasted. Also, to show that this delay of link discovery could be impacted by data plane transmission, a 1 Mbps data traffic is sent on Link1 at the same time, which is used to compare with the case where there is no interfering data plane transmission.

The results are shown in Figure 2.10. We see that

— When there is interference caused by data traffic, the link discovery time is seriously impacted, especially in an in-band setup. This is logical since wireless transmissions share the same transmission channels.

— Two-hop-away links (Link2 in the in-band setup) takes more time to discover than one-hop away links. This is logic, since to discover a two-hop-away link, an LLDP packet traverses two nodes before going back to the SDN controller, hence takes approximately twice the time than discovering a one-hop-away link.

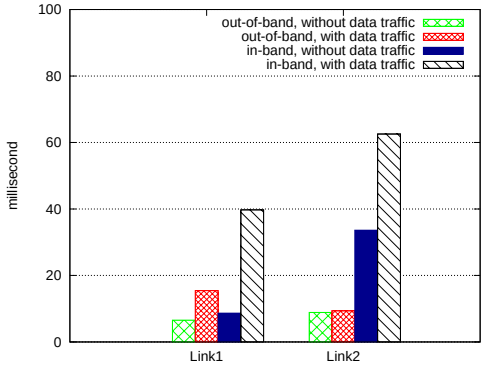

FIGURE 2.10 – Delay of OFDP messages from *packet-out* to *packet-in* for discovering two different links

2.6.3.2 Throughput of OpenFlow control traffic

It's important to have an idea on how much traffic the control plane is producing in a wireless SDN testbed, especially in a in-band setup. If the control plane takes two much bandwidth, it would get into contention with data plane transmissions, which could impact greatly the overall performance of the testbed.

In this experiment, we vary the number of nodes used in the testbed and measure the throughput of the OpenFlow control traffic. The results are shown in Figure 2.11. The throughput of OpenFlow control traffic increases when number of nodes and links increases. Also, we note that the traffic throughput experiences a peak during the initial phase, then it is stable until the end. The volume remains under 10 Kbps for a small topology of less than 12 nodes.

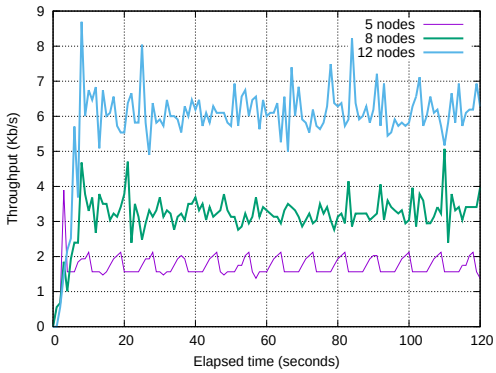

FIGURE 2.11 – OpenFlow control traffic throughput

2.7 Conclusions

In this chapter, we have proposed a topology discovery service for SDN based wireless multi-hop networks. The proposed service can be used by common network control applications and opens the way to the design of novel algorithms that take benefit from : (1) the centralized view and the rich information exposed by our proposed service, (2) the fine grained programming capabilities of SDN. The proposed service extends and adapts the procedures of the *de facto* topology discovery service implemented in most SDN/OpenFlow controllers for wired networks, more precisely OFDP/LLDP and the OpenFlow protocol. The service was implemented on the Ryu controller with Open vSwitch enabled wireless nodes.

Chapitre 3

Resource Allocation for Virtual Link Embedding in Multi-hop Multi-radio Multi-channel Wireless SDN

There is rising interest in applying SDN principles to wireless multi-hop networks, as this paves the way towards bringing the programmability and flexibility that is lacking in today's distributed wireless networks (ad-hoc, mesh or sensor networks) with the promising perspectives of better mitigating issues as scalability, mobility and interference management and supporting improved controlled QoS services.

This chapter investigates this latter aspect and proposes an Integer Linear Programming (ILP) based wireless resource allocation scheme for the provision of point-to-point and point-to-multipoint end-to-end virtual links with bandwidth requirements in software-defined multi-radio multi-channel wireless multi-hop networks. The proposed scheme considers the specificities of wireless communications : the multipoint nature of wireless links which can be leveraged for point-to-multipoint links resource allocations, and, the interference between surrounding wireless links. It also considers switching resource consumption of wireless nodes since, for the time being, the size of SDN forwarding tables remains quite limited. A Genetic Algorithm derived from the ILP formulation is also proposed to address the case of large wireless networks. Our simulation results show that both methods work effectively.

3.1 Introduction

Applying Software Defined Networking (SDN) design principles to wireless networks can pave the way to the emergence of novel and effective wireless network control applications (routing, network resource allocation, mobility management, energy management, etc.) with diverse expected benefits [Haque 2016], notably, an improved global network performance, end-to-end network services with enhanced Quality of Service (QoS), etc.

Indeed, under the assumption of an effective topology discovery service (see Chapter 2) that allows SDN controllers to build an updated global and comprehen-

sive view of the network (despite node mobility and link performance variability and instability) network control algorithms can leverage on this global and detailed view to derive informed and wise control decisions that are able to accommodate with the dynamicity of the network and flows' QoS requirements. Moreover, the flow level forwarding capability of SDN allows unprecedented fine-grained control on the traffic that is flowing in the network. Some of the prominent works from the literature that attempt to apply SDN to wireless networks in order to dynamically control the traffic for an improved provided QoS are : [Yu 2017], [Gante 2014] and [Lee 2016] respectively in the context of wireless ad-hoc, wireless sensor and wireless mesh networks.

The focus of this chapter is on the design of resource allocation methods that enable the on-demand provision of network services with QoS requirements in an SDN enabled multi-radio multi-channel multi-hop physical network. The network service is expressed as a set of point-to-point and point-to-multipoint unidirectional end-to-end virtual links (VLs), each with its own bandwidth requirement. In a network virtualization application context, they can represent the set of virtual links that connect the virtual nodes. In a more general application context, they can be seen as an overlay network service provisioned for a given application.
Two methods are proposed in this chapter. Both aim at mapping the requested virtual links on the substrate wireless network by computing the data paths that minimize link and switching resource consumption as well as interference between wireless links while satisfying the QoS requirements. They also consider and account for some of the specificities of wireless communications, namely the multipoint nature of wireless links which can be leveraged for point-to-multipoint virtual links resource allocations, and the mutual interference caused by transmissions on neighboring links. An Integer Linear Programming based formulation method is proposed to compute the optimal allocations for small and moderate size networks as well as an accompanying genetic algorithm for large networks. A Performance Evaluation is conducted for both methods.

The chapter is organized as follows. Section 3.2 reviews previous work from the literature that are related to virtual network resource allocation in wireless networks. Then, Section 3.3 gives some prerequisites before introducing the system model used in our formulations. Section 3.4 describes the ILP formulation and Section 3.5 describes the genetic algorithm formulation. Section 3.6 presents the performance analysis of the proposed methods. Finally, Section 3.7 concludes the chapter.

3.2 Related works

Virtual network embedding has attracted lots of attention in recent years. While most embedding schemes are conceived for wired substrate network, existing works also considered the case where the substrate network is a wireless one. Table 3.1 summarizes existing works in the field of virtual link resource allocation, classified

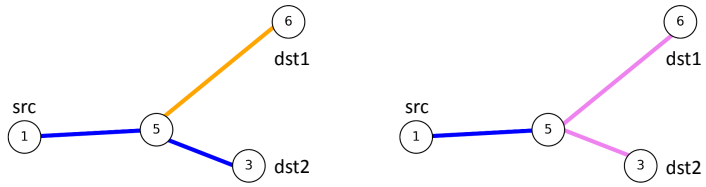

FIGURE 3.1 – An illustration of multicast advantage

according to the virtual link types, QoS support, node resource consideration etc. (in the table, VL stands for virtual link, P2P and P2MP represents Point-to-Point and Point-to-Multi-Point, ILP for Integer Linear Programming and GA for Genetic Algorithm).

As shown in Table 3.1, to the best of our knowledge, our work stands out as the first to address the issue of virtual link embedding in software-defined multi-radio multi-channel multi-hop wireless networks. Moreover, our work do not only consider point-to-point virtual links, but also addresses the case of point-to-multi-point virtual links. It also takes into account switching resources which is crucial especially when dealing with SDN based substrate network. A last characteristic of our method is the optional support of path splitting which allows a virtual link to be established on multiple paths.

3.3 Prerequisites and system model

3.3.1 Prerequisites

3.3.1.1 Prerequisites on multicast advantage

In wireless networks, multicast advantage refers to the fact that one transmission can be received by multiple nodes via the same channel. It's possible to benefit from this for point-to-multipoint communications. For example, in Figure 3.1, to relay the communication between source and destination nodes, node 5 could well use the violet channel to reach *dst1* and *dst2*, instead of using the blue and orange channel. When mapping a virtual link, such a choice should be favoured in order to efficiently use radio resources.

3.3.1.2 Prerequisites on OpenFlow group table

An OpenFlow switch embeds at least one flow table, a group table and a meter table. Each table has a limited number of entries. While forwarding rule allocation in an OpenFlow context has been extensively researched, existing works mainly concern flow table entries, and only a few consider the case of OpenFlow group table entries [Tegueu 2017]. Typically, one group entry is consumed when a flow is split or duplicated. However, in a wireless context, one group table is consumed

TABLE 3.1 – Classification of virtual link resource allocation schemes for wireless multi-hop networks

	VL type	VL QoS	multi-radio	multi-channel	Network model specificity	Method	Node resources	Support path-split
[Li 2017]	P2P (Anypath)	packet loss delay	no	no	link metrics (EATT and EATX)	Incremental virtual network embedding	None	No
[Abdelwahab 2016]	P2P	bandwidth	no	no	mobility	backtracking based heuristic	CPU, storage etc.	Not supported but discussed
[Yun 2013]	P2P	bandwidth	no	no	interference matrix	heuristic	CPU	No
[Lv 2012]	P2P	bandwidth	yes	yes	distance based interference model	greedy algorithm and GA	None	No
[Stasi 2013]	P2P	bandwidth	yes	yes	SINR-based interference model	ILP and heuristic	CPU	yes
our work	P2P P2MP	bandwidth	yes	yes	conflict graph based interference model	ILP and GA	flow table entries and group entries	Yes for ILP, No for GA

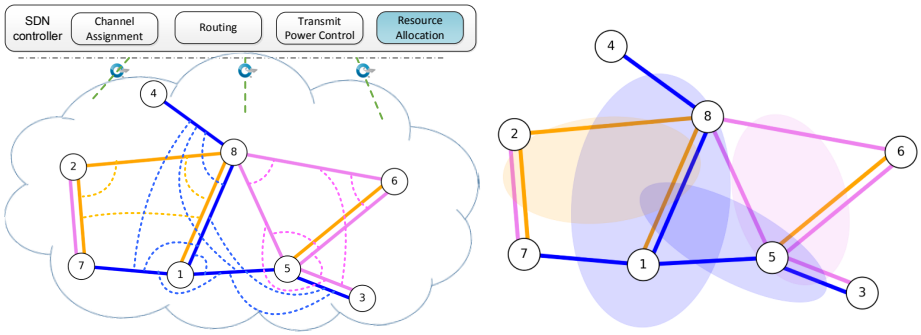

FIGURE 3.2 – An illustration of the substrate wireless SDN network, the conflict pairs between links and the “cliques" that are here represented by minimum ellipsoids covering midpoints of links.

only when the flow is duplicated or split, and then transmitted via at least two distinct channels.

3.3.2 Network model

Each node in a wireless multi-hop multi-radio multi-channel network is equipped with one or multiple Network Interface Cards (NICs). Each NIC is tuned to a channel and, any two NICs at the same node are tuned to different channels, in order to efficiently and fully make use of radio resources. We assume that the channel assignment is given and static. There are in total $|\Lambda|$ non-overlapping frequency channels in the system and each node is equipped with q NICs where $q \leq |\Lambda|$. The channel capacity of λ is noted as B_λ.

We model the multi-hop multi-radio multi-channel backbone network as a directed graph $G = (V, E)$ where V is the set of SDN nodes and $E \subseteq V \times V$ the set of bidirectionnal physical links which operate in half-duplex mode. However, while one NIC at a node v is transmitting/receiving data on one channel, another NIC at node v can simultaneously transmit/receive data on a different channel.

As we consider an OpenFlow-enabled infrastructure, to each node $v \in V$, is associated a switching capacity L_v, which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node v flow table is denoted by L'_v. Similarly, M_v and M'_v denote respectively the maximum and current size of the group table of node v. We assume that we have already obtained a good channel assignment ζ. ζ assigns to each node $v \in V$ a set of $\zeta(v)$ of $|\Lambda|$ different channels : $\zeta(v) \subseteq \Lambda$.

A pair of NICs can communicate with each other if they are on the same channel and are within the transmission range of each other. In other words, the wireless link $e = ((u, v), \lambda)$, $u, v \in V$ and $\lambda \in |\Lambda|$ belongs to the substrate network, if channel $\lambda \in \zeta(u) \cap \zeta(v)$ and on this latter channel, nodes u and v are within the transmission range of each other. The set of neighbours of v (via any channel) is noted as N_v.

3.3.3 Interference model

Our interference model is based on the concept of conflict graphs [Jain 2003]. A conflict graph is related to a channel. It describes the presence of interference (represented as edges in the graph) between pairs of links (represented as vertices) if both links are active simultaneously. Following the logic of some previous works [Gupta 2007], from the conflict graph, we derive maximal cliques [Cazals 2008]. A clique is a subgraph of the conflict graph where all nodes interfere with each other. Maximal cliques are jointly or individually used in different ways to derive different kinds of constraints on the transmission rates of the wireless links that form the clique. For instance, since links belonging to a maximal clique cannot be active simultaneously, their aggregate transmission rate must be lower than the channel capacity.

For example, in Figure 3.2, we show a network substrate composed of eight nodes and using three channels (in three different colors). Solid links represent transmission links and dashed lines represent interference relationship between transmission links. In this example, we can see that the interference set of the orange link (1, 8) is {(2, 8) and (2, 7)}, indicated by the dashed link between them. Instead of using the usual way of presenting cliques with conflicting links as "node", in this chapter cliques are directly draw on the illustrating graph, with ellipsoids covering the midpoint of each link in the clique. For example, the orange clique covers the midpoints of link (2, 7), (2, 8) and (1, 8), meaning that those three links belongs to the same clique.

There are different ways to build the conflict graphs with different levels of accuracy in capturing the interference. In this work, we do not promote any method and assume that we have the maximal conflict graphs as input. Thanks to the centralized nature of SDN and the adoption of an effective topology discovery service at the controller, the appropriate conflict graphs that meet the needs of network control applications can be made available by the controller. Also, we do not stick to any method to exploit the maximal cliques even if we have chosen one, for illustration, in our formulation of the resource allocation problem.

In the following, we denote the set of maximal cliques as C. We also denote the set of wireless links that form a clique c as E_c, and the nodes of the substrate network in the clique as S_c.

3.3.4 Virtual links request model

A virtual links request consists of a set of $|K|$ virtual links. Each virtual link $k \in K$ is characterized by :

— a source node $s_k \in V$, and a set of destination nodes $T_k \in V \backslash \{s_k\}$ (when $|T_k| = 1$, the VL is point-to-point, otherwise it is point-to-multipoint) ;

— a bandwidth requirement of b_k ;

The sequence of virtual links requests is noted as $\vec{\mathbf{K}} = [K_1, K_2, ...,]$.

3.4 ILP problem formulation

Based on the work [Capelle 2015] whose focus was on wired SDN networks, this section describes our ILP formulation of the online virtual links resource allocation on an SDN based wireless multi-hop substrate network. In comparison to the previous work, this formulation adds in many aspects by taking into consideration (1) the broadcast nature of wireless links, which we use it as a leverage to efficiently support point-to-multipoint virtual links, as well as, (2) the interferences between surrounding links which is minimized and distributed on different cliques (regions) to improve the admissibility of forthcoming virtual links requests.

Below, the variables and problem constraints are listed ; Then, the considered objective function is defined.

3.4.1 Resource-related assignment variables

The output of our assignment problem is the set of routes (with the bandwidth allocations at each supporting physical link and the number of flow and group table entries at each traversed node) that support each of the virtual links that composes a request. It is worth noting that since VLs may be point-to-multipoint, it is likely that resource allocations will be mutualized close to the source and as we get closer to destinations, they will tend to be more and more dedicated to specific destinations. As a consequence, basic assignment variables are related to a specific destination of a VL. In our model, we distinguish the following variables :

— $f_k^t((v,u),\lambda)$ is an integer variable that represents the bandwidth allocated at link $((v,u),\lambda)$ to the packets of VL k that are flowing from the origin node s_k to a destination node t. More generally, $f_k((v,u),\lambda)$ refers to the amount of bandwidth used on link $((v,u),\lambda)$ by the VL k, whatever the destination. It is set to the maximum of $f_k^t((v,u),\lambda)$ for all $t \in T_k$. Specific to the broadcast nature of wireless medium transmissions, in which one node can deliver a paket to multiple neighbors from one transmission, we also introduce an integer variable denoted as $f_k(v,\lambda)$ that refers to the amount of bandwidth used on channel $\lambda \in \zeta(v)$ by node v to support the VL k. It is set to the maximum of $f_k((v,u),\lambda)$ for all $u \in N(v)$ such as $((v,u),\lambda) \in E$.

— $l_k(v)$ is a binary variable that indicates the switching resources consumed by VL k at node v. It is expressed as the number of entries that are installed in node v flow table to support VL k with the assumption that all entries consume the same amount of resources regardless of the complexity of the *match* operation and the related *instructions* to perform. In this work, we assume that each VL consumes 1 flow table entry at each traversed node. A node is traversed by a VL if at least one of its adjacent physical links supports the VL. Formally :

$$\forall k \in K, \forall v \in V, \forall u \in N(v),$$
$$\forall \lambda \in \zeta(v) \cap \zeta(u) : g_k((v,u),\lambda) \leq l_k(v) \tag{3.1}$$

$$\forall k \in K, \forall v \in V, \forall u \in N(v) :$$
$$l_k(v) \leq \sum_{\lambda \in \zeta(v) \cap \zeta(u)} (g_k((v,u),\lambda) + g_k((u,v),\lambda)) \tag{3.2}$$

where $g_k((v,u),\lambda)$ is an intermediate binary variable that indicates if some bandwidth at link $((v,u),\lambda)$ is assigned to VL k or not. It is derived from another set of more focused intermediate variables $g_k^t((v,u),\lambda)$ that reflects some part of flow of packets of VL k destined to $t \in T_k$ traverses the link $((v,u),\lambda)$ (i.e. $g_k^t((v,u),\lambda) = 0$ if $f_k^t((v,u),\lambda) = 0$ and $g_k^t((v,u),\lambda) = 1$ otherwise).

— $m_k(v)$ is a binary variable indicating if a group table entry is assigned to VL k at node v. A group table entry is consumed by a VL at a node, if it is split (for multipath) or duplicated (for multicast) and supported by two ongoing links operating on two distinct channels. We see that for any VL k, as long as there are no less than two out-going flows from a node via no less than two different channels, a group table entry is consumed, expressed as :

$$\forall k \in K, \forall v \in V :$$
$$m_k(v) = \begin{cases} 0 & if \ \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 \\ 1 & otherwise \end{cases}$$

where $g_k(v,\lambda)$ is an intermediate boolean variable that indicates if node v relays packets from VL k on channel λ, whatever its neighbors on this channel. It is derived from set of previous variables $g_k((v,u),\lambda)$ with $u \in N(v)$ and $\lambda \in \zeta(v) \cap \zeta(u)$. This equation could be easily linearized as follows :

$$\forall k \in K, \forall v \in V :$$
$$2m_k(v) \leq \sum_{\lambda \in \zeta(v)} g_k(v,\lambda) \leq 1 + |\Lambda| \, m_k(v) \tag{3.3}$$

— ξ_{max} and ξ_{min} which refer to the maximum and minimum clique utilization after request acceptance (i.e. by taking into account the bandwidth allocations consumed by the virtual links that compose the request).

— l_{max} and l_{min} which similarly refer to the maximum and minimum flow table utilization (when considering all network nodes) after request acceptance.

3.4.2 Problem constraints

The constraints on bandwidth allocations are described hereafter in equations 3.4 to 3.12. The constraints related to switching resources allocation is given by inequalities 3.4 and 3.5. They simply ensure that the total number of flow and group table entries assigned to VLs composing the request, does not exceed available nodes' flow and group tables entries. Equation 3.6 reflects the linearization of the *Max* and *Min* operator applied to the variables $l_k(v)$ to get l_{max} and l_{min}.

$$\forall v \in V : \sum_{k \in K} l_k(v) \leq L_v - L'_v \tag{3.4}$$

$$\forall v \in V : \sum_{k \in K} m_k(v) \leq M_v - M'_v \tag{3.5}$$

$$\forall v \in V : l_{min} \leq L_v - L'_v + \sum_{k \in K} l_k(v) \leq l_{max} \tag{3.6}$$

Constraint 3.7 reflects the linearization of the maximum bandwidth $f_k^t(e)$ allocated to VL k at link $e = ((v, u), \lambda)$, whatever the destination. It allows to avoid the link stress problem that can occur when allocating point-to-multipoint VLs. Equation 3.8 ensures that the total bandwidth assigned to each clique does not exceed the remaining bandwidth of the clique. In this equation, each clique with it's associated channel λ is noted as $(c, \lambda) \in C$, which is composed of E_c, and the residual capacity is $\xi(c) = B_\lambda - \xi'(c)$, with $\xi'(c)$ denoting the bandwidth allocations related to already admitted virtual links on all the physical links that compose the clique c. Equation 3.10 is the usual flow conservation constraints.

Equation 3.9 reflects the linearization of the *Max* and *Min* operator applied to the variables ξ_c (with $c \in C$) to get ξ_{max} and ξ_{min}. Equation 3.10 is the usual flow conservation constraints.

$$\forall k \in K, \forall e = ((v, u), \lambda) \in E, \forall t \in T_k : f_k^t(e) \leq f_k(e) \tag{3.7}$$

$$\forall (c, \lambda) \in C : \sum_{k \in K} \sum_{v \in S(c)} f_k(v, \lambda) \leq B_\lambda - \xi'(c) \tag{3.8}$$

$$\forall (c, \lambda) \in C : \xi_{min} \leq B_\lambda - \xi'(c) - \sum_{k \in K} \sum_{v \in S(c)} f_k(v, \lambda) \leq \xi_{max} \tag{3.9}$$

$$\forall k \in K, \forall t \in T_k, \forall v \in V :$$

$$\sum_{u \in N(v)} \sum_{\substack{\lambda \in \zeta(u) \cap \zeta(v) \\ e_1((v,u),\lambda) \\ e_2((u,v),\lambda)}} (f_k^t(e_1) - f_k^t(e_2)) = \begin{cases} b_k & if\ v = s_k \\ -b_k & if\ v = t \\ 0 & else \end{cases} \tag{3.10}$$

Equation 3.11 is a channeling constraint between integer and binary variables : $f_k((v,u),\lambda)$ and $g_k((v,u),\lambda)$. It also constrains the VL k's bandwidth assignment at a physical link to the requested bandwidth b_k. Equation 3.12 constrains the bandwidth that is assigned to the flow of packets destined to a specific VL's end-point (or destination) within a range of values, in addition to establishing a channeling constraints between binary and integer variables. The inequality on the right side ensures that the bandwidth requirement of the VL is never exceeded. The inequality on the left side directs path-splittting and avoids the multiplication of splits with low bandwidth allocations. Indeed, if active, path-splitting is feasible only if the bandwidth allocated to the splits respects a minimum threshold b_k^{min}. In practice, b_k^{min} is a ratio of b_k, $b_k^{min} = PS_{ratio} * b_k$ with $PS_{ratio} \in [0,1]$ (then, $PS_{ratio} \leq 0.5$ when the path-splitting is allowed, and $PS_{ratio} = 1.0$ when it is forbidden).

$$\begin{aligned} &\forall k \in K, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \\ &\quad g_k((v,u),\lambda) \leq f_k((v,u),\lambda) \leq b_k * g_k((v,u),\lambda) \end{aligned} \tag{3.11}$$

$$\begin{aligned} &\forall k \in K, \forall t \in T_k, \forall v \in V, \forall u \in N(v), \forall \lambda \in \zeta(v) \cap \zeta(u) : \\ &\quad b_k^{min} * g_k^t((v,u),\lambda) \leq f_k^t((v,u),\lambda) \leq b_k * g_k^t((v,u),\lambda) \end{aligned} \tag{3.12}$$

3.4.3 Objective function

The objective function :
Minimize

$$\begin{aligned} &\alpha_1 \sum_{k \in K} \sum_{v \in V} \sum_{\lambda \in \zeta(v)} f_k(v,\lambda) + \alpha_2 \sum_{k \in K} \sum_{v \in V} l_k(v) \\ &+ \alpha_3 \sum_{k \in K} \sum_{v \in V} m_k(v) + \beta \sum_{(c,\lambda) \in C} \sum_{v \in S(c)} \sum_{k \in K} |E(c)|\, f_k(v,\lambda) \\ &+ \beta_2(\xi_{max} - \xi_{min}) + \beta_3(l_{max} - l_{min}) \end{aligned} \tag{3.13}$$

The objective function is set to take into account both the resource consumption and the interference introduced by bandwidth allocations on surrounding links. For that, the main objective of our approach is to minimize the total resources required

to map virtual links, which is represented by the first three terms that cover respectively links bandwidth, flow tables and group tables resources. In addition, the objective function mitigates the interference between links by avoiding overloading links belonging to cliques with a high number of members. This favours radio resource spatial reuse, increasing the overall available network resources. Finally, the last two terms aim at reducing the disparities of cliques' bandwidth utilization and flow tables' utilization. They also contribute improving flow admissibility.

3.5 Genetic Algorithm

Exact solutions to the considered problem can be obtained by solving our previously presented ILP-based algorithm. However, the complexity of computation, which increases exponentially with the number of parameters (number of nodes, links, radios and channels in our case), might make it practically infeasible for large networks. Therefore, a practically feasible approach is to find a proficient near-optimal solution while sustaining realistic performance. In this section, we present a genetic algorithm based solution to address this aspect.

3.5.1 General background

GAs, similar to other evolutionary algorithms, are inspired from the natural evolution process (i.e. "survival of the fittest") to search for the optimal solution to a given problem. Each point χ in the search space is a potential solution or an *individual* that the GA will try to find by exploring the space in an iterative way. The process is simple. From a randomly generated set of a given number N_p of individuals, which is designated as the *initial population*, GA creates successive generations of N_p individuals by applying simple reproduction operations. The odds of the survival of an individual depend on its *fitness values*. The attempt is to produce a new generation or descendants with a better fitness value than their parents. Typically, parents are chosen to mate with probability proportional to their fitness called proportional selection to privilege the survival and reproduction of the fittest individuals. This process of selection and reproduction (through crossover and mutation) continues until, either GA reaches N_g (the maximum number of generations) or when an acceptable fitness level for the entire population is achieved. The individual with the best fitness value will be considered as the near optimal solution. Normally, a GA requires an encoding scheme to genetically represent the individuals, and an optimization function to evaluate their fitness. Also, by following the natural selection process, a GA includes three major steps : selection, crossover and mutation. Algorithm 1 presents our GA-based resource allocation scheme. Hereafter, we will describe its main components : encoding scheme, initial population computation, fitness evaluation, and the genetic operations selection, crossover and mutation.

Algorithm 1: GA-based Resource Allocation

Input : $G(V,E); K;\ W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]\ ;\ \alpha_1; \alpha_2; \alpha_3; \beta_1; \beta_2; \beta_3; N_p; N_g;$ $cxPB; mutPB\ ;$

Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]$ (i.e. the best individual)

1 **begin**
2 $\quad P_0 \longleftarrow$ `InitialPop`(G, K, N_p, W)
3 $\quad P \longleftarrow P_0$
4 $\quad$ **for** ($j = 0;\ j < N_g;\ j++$) {
5 $\quad\quad$ **for** ($c = 0;\ c < N_p;\ c++$) {
6 $\quad\quad\quad (\chi^a, \chi^b) \longleftarrow$ `TournamentSelection`(P)
7 $\quad\quad\quad \chi^c \longleftarrow$ `Crossover`$(G, K, (\chi^a, \chi^b), cxPB, W)$
8 $\quad\quad\quad P \longleftarrow P \cup$ `Mutation`$(\chi^c, mutPB)$
9 $\quad\quad P \longleftarrow$ `PopulationSelection`$(P, N_p, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3)$
10 $\quad \chi \longleftarrow$ `SelectBestIndividual`(P)

3.5.2 Encoding scheme

To use a GA, it is necessary to choose a representation that defines the genotype of an individual which is conceptually designated as a chromosome. In our case, an individual is a possible solution to a request for resource allocation. Recall that each request K consists of a set of point-to-point and/or point-to-multipoint virtual links. We naturally represent an individual i denoted by χi as a vector of genes where each gene τ_k maps resources assigned to a virtual link $k \in K$. In other words, $\chi i[k] = \tau_k^i$ refers to gene k of individual i. As our GA doesn't take into consideration the case of path splitting, a tree connecting the source s_k to the destination nodes $t \in T_k$, is sufficient to represent a gene. This tree is in fact a subgraph of the substrate network graph G. Each tree is associated with switching resources and links bandwidth respectively allocated at each substrate node and link belonging to this tree. Also, each link e in each tree is associated with a link cost w_e, and the set of list costs of all links in all trees in noted as W. This link cost is used in shortest-path-like algorithms that will be used in our GA.

3.5.3 Initial population

The first step in the functioning of a GA is the generation of an initial population (Algorithm 1 - line 2). It is computed by generating a given population size (N_p) with each member of this population encoding an individual representing a possible solution. One important objective is to have a reasonable diversity among the initial population, in order to avoid premature local convergence. In our case, as detailed in Algorithm 2, to generate an individual i (Algorithm 2 - line 4), we compute the minimum Steiner tree (constructed with shortest path heuristic) as a routine to build the tree representation of each genes k (Algorithm 2 - line 9). Note that in the case of multiple links between two nodes, the link with the minimum link

Algorithm 2: Initial Population Computation

Input : $G(V,E); K; N_p;\ W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$
Output: $P_0 = \{\chi^i, \forall i \in \{1, ..., N_p\}\ with\ \chi^i = [\tau_1^i, \tau_2^i, ..., \tau_{|K|}^i]\}$

```
1  begin
2      P_0 ⟵ ∅
3      G'(V', E') ⟵ Clone(G(V, E))
4      W' ⟵ Clone(W)
5      for ( i = 0; i < N_p; i++ ) {
6          foreach e ∈ E' do
7              W'[e] ⟵ W'[e] × Random(1, 1.5)
8          foreach k ∈ K do
9              τ_k^i ⟵ ComputeSteinerTree(G', W'[e], s_k, T_k)
10             χ^i[k] ⟵ τ_k^i
11         P_0 ⟵ P_0 ∪ χ^i
```

cost is sustained for Steiner tree construction. To bring diversity, at each Steiner tree construction, the cost of each link in the substrate network is multiplied by a random factor in the range of $[1, 1.5]$.

3.5.4 Fitness function

After creating the initial population, each individual is evaluated and assigned a fitness value according to a fitness function. The optimality of a solution is defined by its corresponding fitness value. Also, a common technique to solve constrained optimization problem using GA is to apply penalty functions related to the violation of constraints. As a result, the problem is converted from a constrained to an unconstrained optimization problem. Equation 3.14 defines our fitness function.

$$\begin{aligned} F(\chi) =& (F_{bw}(\chi) + F_{openflow}(\chi) + F_{group}(\chi) + F_{interf}(\chi) \\ &+ F_{bw_balance}(\chi) + F_{sw_balance}(\chi)) \\ &* \hat{F}_{cliques}(\chi) * \hat{F}_{sw}(\chi) \end{aligned} \tag{3.14}$$

where

$$F_{bw}(\chi) = \alpha_1 \sum_{\tau \in \chi} \sum_{e \in \tau} \phi(\tau, e) b_\tau$$
$$F_{openflow}(\chi) = \alpha_2 \sum_{\tau \in \chi} \sum_{v \in V} \sigma(\tau, v)$$
$$F_{group}(\chi) = \alpha_3 \sum_{\tau \in \chi} \sum_{v \in V} \eta(\tau, v)$$
$$F_{interf}(\chi) = \beta_1 \sum_{\tau \in \chi} \sum_{(c,\lambda) \in C} \sum_{v \in S(c) \cap S(\tau)} |S(c)| \, b_\tau$$
$$F_{bw_balance}(\chi) = \beta_2 * (max_bw(C, \chi) - min_bw(C, \chi))$$
$$F_{sw_balance}(\chi) = \beta_3 * (max_sw(V, \chi) - min_sw(V, \chi))$$
$$\hat{F}_{cliques}(\chi) = 1 + 100 \sum_{(c,\lambda) \in C} \delta(\chi, c)$$
$$\hat{F}_{sw}(\chi) = 1 + 100 \sum_{v \in V} \theta(\chi, v)$$

In the fitness function, $\phi(\tau, e)$, $\sigma(\tau, v)$, $\eta(\tau, v)$, $\delta(\chi, c)$ and $\theta(\chi, v)$ are all indicator functions that take value 0 or 1. Furthermore, the parameters α_1, α_2, α_3, β_1, β_2 and β_3 here are the same as in the objective function of the ILP.

$\phi(\tau, e)$ indicates if an edge e in a tree τ supporting a virtual link is transmitting or not. It takes value 1 for all edges in the tree τ except those who use the multicast advantage : in the latter case, $\phi(\tau, e)$ takes value 0. b_τ is the requested bandwidth of a virtual link, and corresponds to the b_k in the ILP formulation. Hence we have $F_{bw}(\chi)$ which is the sum of bandwidth consumed by transmitting links.

In the same manner, $\sigma(\tau, v)$ indicates if a node v is included in the tree τ or not, hence consuming one OpenFlow table entry. $\eta(\tau, v)$ indicates if a node v serves as a multicast node or not, hence consuming one group table entry. We have therefore $F_{openflow}(\chi)$ and $F_{group}(\chi)$ which represent the switching resources to be consumed by adopting χ as a solution.

$F_{interf}(\chi)$ reflects the total interference brought by the instantiated virtual links, defined in the same manner as in the ILP.

For clarity reasons, detailed explanations of $F_{bw_balance}(\chi)$ and $F_{sw_balance}(\chi)$ are not given here. They correspond to the maximum minus minimum clique bandwidth consumption among all cliques and maximum minus minimum flow table entries consumption among all nodes and can also easily be calculated with $\phi(\tau, e)$ and $\sigma(\tau, v)$.

Those six fitness terms correspond to the objective function in the ILP formulation, i.e. they give the same results when the virtual link embeddings are the same.

Unlike the ILP formulation, the constraints on cliques and switching resources are also included in the objective function in the GA fitness function, i.e. the $F_{cliques}(\chi)$ and $F_{sw}(\chi)$ multiplier. In a feasible solution, those two terms should

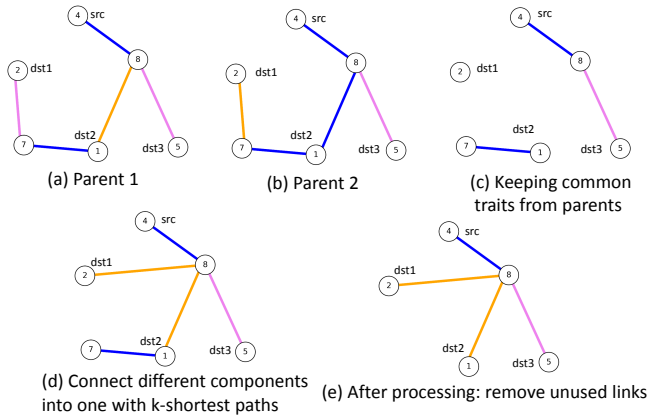

FIGURE 3.3 – Crossover of two parent trees to get a offspring

be at 1. However, in some cases due to the sparsity of feasible solutions, those infeasible solutions should not be removed from the population. In fact, some solutions are more infeasible than others, and should be reflected in our fitness function.

To reflect to which extent a solution χ is far from a feasible solution, we penalize those infeasible solutions according to how seriously they violate the clique bandwidth and the switching resource constraints. $\delta(\chi, c)$ indicates if the bandwidth allocations chosen in χ respect clique c bandwidth constraint, i.e. the total bandwidth of transmitting links in c which come from χ, doesn't violate the constraint delimited by the minimum remaining capacity of all links in c. The 100 here is a large number (compared to 1 as a multiplier) that penalizes violations of constraints. The more we have clique constraint violations for χ, the larger the fitness function $\hat{F}_{cliques}(\chi)$. In the same manner, $\theta(\chi, v)$ indicates if a node respects its switching resource constraint or not, and $\hat{F}_{sw}(\chi)$ reflects to which extent the violation is serious, i.e. the number of nodes that doesn't respect its switching resource constraint. Those two fitness terms correspond to the constraints of switching resources and of cliques in the ILP formulation.

3.5.5 Crossover schemes

In this stage (Algorithm 1 - Line 6), chromosomes from a population are selected for reproduction (crossover). The operation of selection aims at favoring reproduction and survival of the fittest individuals. Many schemes of selection have been proposed in the literature, and we choose to use tournament selection. Tournament selection executes roulette selection (in which the probability of a particular chromosome getting selected is directly proportional to its fitness) for a fixed number of times (depending on the tournament size) in order to generate a tournament subset of chromosomes. The best chromosome out of this set is then treated as the selected chromosome. In this way, fittest individuals have more chance to be reproduced and to survive.

We use tournament selection of size 3 to select a pair of chromosomes as the

Algorithm 3: Crossover Scheme

Input : $G(V,E); K; \chi^a; \chi^b; cxPB$; $W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
Output: $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_{|K|}^c]$

1 **begin**
2 $G'(V',E') \longleftarrow$ `Clone`$(G(V,E))$
3 $W' \longleftarrow$ `Clone`(W)
4 **if** (`Random`$(0,1) < cxPB$) **then**
5 **foreach** $k \in K$ **do**
6 $\tau_k^c \longleftarrow$ `Similitude`(τ_k^a, τ_k^b)
7 **while** `isNotConnected`(τ_k^c) **do**
8 $\tau_k^c \longleftarrow$ `randomKShortestPath`(G', W', τ_k^c)
9 $\chi^c[k] \longleftarrow \tau_k^c$
10 `updateProhibitiveLinkCost`(G', W', τ_k^c)

parents to produce a single offspring by applying crossover operator between them (Algorithm 1 - Line 7). Crossover is authorized with some probability $cxPB$. Its strategy is summarized in Algorithm 3. The idea is simple and consists to pass common genes or traits from parents to offspring according a specific logic called Similitude (Algorithm 3 - Line 4). In order to explain how this primitive works, let $\chi^a = [\tau_1^a, \tau_2^a, ..., \tau_K^a]$ and $\chi^b = [\tau_1^b, \tau_2^b, ..., \tau_K^b]$ be the selected parents. The crossover operator generates a child $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_K^c]$ by identifying the same links between τ_k^a and τ_k^b for each $k \in K$, and retaining these common links in τ_k^c (as in [Lu 2013]). According to the definition of the fitness function, the “better” individual has higher probability of being selected as a parent. Thus, the common links between two parents are more likely to represent the “good" traits. However, retaining these common links in τ_k^c may generate some separate sub-trees. Therefore, links are needed to be selected to connect these isolated sub-trees into a tree.

Figures 3.3 (a), (b) and (c) illustrate this situation. To maintain diversity among solutions, instead of connecting separated components each time with shortest path, we adopt a random of k-shortest paths (with k $\leq$ 5) (we use the method mentioned in [Parsa 1998] to construct k-shortest paths between two trees). This process is repeated until τ_k^c becomes connected (Algorithm 3 - Line 5 to 6). As shown in Figure 3.3 (d) and (e), a post-processing can be required to remove isolated branches of the tree that contain neither the source node nor destination nodes. As the cross-over is carried out from one VL to a next one that compose the request, the function updateProhibitiveLinkCost (Algorithm 3 - Line 10) assigns an infinity link cost to links that belong to cliques with no longer bandwidth left for future VLs, and to links with one end node with no flow table entries left. In this way, infeasible solutions are kept away when possible.

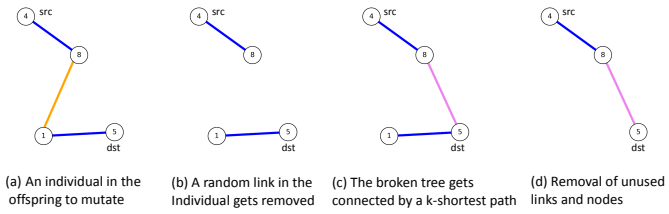

FIGURE 3.4 – Mutation scheme I : break-down and reconnection

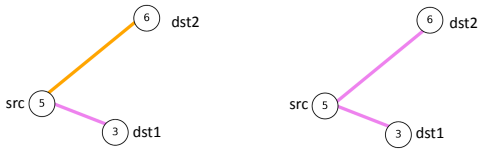

FIGURE 3.5 – Mutation scheme II : channel mutation

3.5.6 Mutation schemes

When a new offspring is produced, the mutation operation is performed according to the mutation probability $mutPB$ (Algorithm 1 - Line 8). We identify two types of mutations that could be very helpful, i.e. (1) mutation based on link breaking and reconnection, (2) mutation based on channel transition. The action of mutation gives more chances of getting rid of local sub-optimal solutions.

For the first type of mutation (as shown in Figure 3.4), the procedure randomly selects a link in the tree and remove it to create two separate sub-trees ; then, it re-connects these separate sub-trees using a k-shortest path.

The second type of mutation comes from the importance of channels in our problem (as shown in Figure 3.5). That is, when a selected link to mutate in the tree corresponds to a multi-link in the substrate network, it's possible to directly change the channel of this link. This could lead us to finding more opportunities of multicast advantage, or a better load balancing among cliques.

3.5.7 Balanced resource allocation with dynamic link cost

We set the normal link cost of $e = ((v, u), \lambda)$ as $w_e = \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$. If this static manner of defining the link cost is used across all requests in $\vec{\mathbf{K}} = [K_1, K_2, ...]$, cliques with low link costs are always favored in comparison to cliques with high link costs, regardless of their current load, leading to unbalanced cliques utilizations. Although in our fitness function the clique utilization balancing is taken into consideration, it's inefficient if most candidate solutions in GA lead to an unbalanced situation. To mitigate this, we give a dynamic version of link cost, as shown in Algorithm 4 and Algorithm 5. The idea of the algorithm is that if the clique utilization of a clique c is among the top N^c_{top} most loaded, the link cost of links that form c should be multiplied by a factor of 1.1. If the link $e \in c$ is yet being used in the current embedding of K (i.e. $e \in \chi^K$, as shown in Line-12 of Algorithm 5), then an even higher multiplying factor (i.e. 1.5) should be given to

Algorithm 4: Update Link Cost Dynamically

```
Input : G(V,E); C; N^c_top; N^s_top; W; χ;
1  begin
2      C_most, C_least, S_most, S_least ⟵ ∅
3      if cu_most − cu_least ≥ 30% then
4          C_most ⟵ mostUsedCliques(C, N^c_top)
5          C_least ⟵ leastUsedCliques(C, N^c_top)
6      if su_most − su_least ≥ 30% then
7          S_most ⟵ mostUsedSwitches(V, N^s_top)
8          S_least ⟵ leastUsedSwitches(V, N^s_top)
9      foreach e ∈ E do
10         flagNormal ⟵ true
11         if e ∈ C_most or e ∈ S_most then
12             if e ∈ χ^K then
13                 W[e] ⟵ W[e] × 1.5
14             else
15                 W[e] ⟵ W[e] × 1.1
16             flagNormal ⟵ false
17         if e ∈ C_least or e ∈ S_least then
18             if e ∈ χ^K then
19                 W[e] ⟵ W[e] ÷ 1.5
20             else
21                 W[e] ⟵ W[e] ÷ 1.1
22             flagNormal ⟵ false
23         if flagNormal = true then
24             W[e] ⟵ α_1 + α_2 + β_1 |E(c)|
```

the link e, before calculating the embedding solution of K_{next}. In this way, those most used cliques will be unfavored in the embedding of forthcoming requests, due to their high link costs. Note that the increase of link cost can be accumulated over time, i.e. if a clique stays always among the top most loaded from K_i to $K_{i+\Delta}$, its links costs will be increased $\Delta + 1$ times, until the clique is removed from the top most loaded list, at which point the normal link cost is given to the links in the clique. The same idea is incorporated into the algorithm for switch resources, i.e. links that have one end with switch resources among the N^s_{top} most used will be unfavored due to the increase of its link cost. Inverse actions are carried out on the N^c_{top} least used cliques and the N^s_{top} least used switches. At each iteration, links in other situations are given normal link cost.

Algorithm 5: Balanced Resource Allocation With Dynamic Link Cost

1 **Input** : $G(V,E); \vec{\mathbf{K}}; C; N_{top}; \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3,$
$N_p, N_g, cxPB, mutPB$
2 **begin**
3 **foreach** $e \in E$ **do**
4 $W[e] \longleftarrow \alpha_1 + \alpha_2 + \beta_1 \left|E(c)\right|$
5 **foreach** $K \in \vec{\boldsymbol{K}}$ **do**
6 $\chi^K \longleftarrow$ `geneticAlgorithm`$(G, K, W, \alpha_1,$
$\alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, N_p, N_g, cxPB, mutPB)$
7 `updateLinkCostDynamically`(G', W', χ^K)

3.6 Performance Evaluation

The objectives of this performance analysis is to show that our methods clearly succeed in capturing several essential aspects of wireless links : (1) their broadcast nature which should be exploited whenever possible when embedding point-to-multipoint virtual links ; and (2) interference between neighbouring links which should be avoided whenever possible ; and (3) to achieve a decent load-balancing of cliques and flow tables utilization to improve admissibility. It also compares both proposed methods and investigates the trade-off raised by these latter : accuracy versus computation time. Below, we describe our simulation model, the main performance metrics and some of the obtained results.

3.6.1 Heuristic algorithms for comparison

The considered algorithm is presented in Algorithm 6. It is a simple Steiner tree construction for each VL in a request sequentially, with the two complementary specificities : (1) For each VL k in a request K, a function called updateProhibitive-LinkCost (Algorithm 6 - Line 8) is called which gives infinity link cost to links that with one node what has no OpenFlow table entry left, or those links that has not enough bandwidth for the forthcoming VL k_{next} ; (2) Three different link metrics are used for comparison (Algorithm 6 - Line 9), which can contribute differently to the acceptance rate :

— Metric-1 : Dynamic Link Metric, as presented in Algorithm 4.

— Metric-2 : Link metric (associated to the clique c) defined as $\alpha_1+\alpha_2+\beta_1 \left|E(c)\right|$.

— Metric-3 : Link metric (associated to the clique c) defined as $\frac{\alpha_1+\alpha_2+\beta_1\left|E(c)\right|}{residual_capacity(c)}$.

We can see that the Metric-2 and Metric-3 don't take into consideration the switching resources. It is expected that Metric-1 and Metric-3 would lead to a decent load balancing between cliques, which should not be the case for Metric-2. It is also expected that Metric-1 would help in balancing switch resource consumption.

Algorithm 6: Shortest path heuristic for comparison

Input : $G(V,E); \vec{\mathbf{K}}$;
Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]\}$

```
1 begin
2     G'(V', E') ⟵ Clone(G(V, E))
3     W = InitiateLinkCosts()
4     foreach K ∈ K⃗ do
5         foreach k ∈ K do
6             τ_k ⟵ ComputeSteinerTree(G', s_k, T_k, W)
7             χ[k] ⟵ τ_k
8             updateProhibitiveLinkCost(G', W)
9         updateLinkCost(G', W, χ)
```

3.6.2 Network Model

For illustration purpose, one single network instance is considered in the presented results. It is composed of 20 nodes connected via 60 links. Nodes are equipped with up to 3 radio interfaces that operate on 6 disjoint frequency bands (channels). The capacity of each channel is set to 180 units of bandwidth (UB). The left side of Figure 3.6 depicts the network topology, each link color reflects a frequency band. It leads to 9 cliques (as depicted in right side of Figure 3.6) with a number of members ranging from 3 to 11 links. Unless specified, the flow table and group table maximum size are set to 1000 and 100, respectively.

3.6.3 Load Model

Virtual links requests are composed of a number of point-to-point and point-to-multipoint virtual links randomly chosen between 4 and 6. Each point-to-multipoint virtual link has a number of destinations randomly chosen between 2 and 6. The bandwidth requirement of each virtual link is also chosen randomly from 1 to 3 UB. Source and destination selection is performed on a random basis. The request arrivals follow a Poisson process with an arrival rate r of 0.01, 0.02, 0.03, 0.04, i.e. in average 1, 2, 3 or 4 requests each 100 units of time (UT). The request life-time conforms to an exponential distribution with an average of 1000 UT. For each arrival rate $r = 0.01, 0.02, 0.03, 0.04$, we generate 1 different request series and use those 1 request series across the whole chapter in different experiments (with replay).

3.6.4 Simulation settings

The Integer Linear model was implemented in Python with CPLEX 12.63 solver. The experiments were carried out on a virtual machine with 25 vCPU and 16GB of RAM and running Ubuntu 14.04. A gap of less than 1% to the optimal solution is considered satisfactory. Unless specified, path splitting is disabled for ILP. For GA,

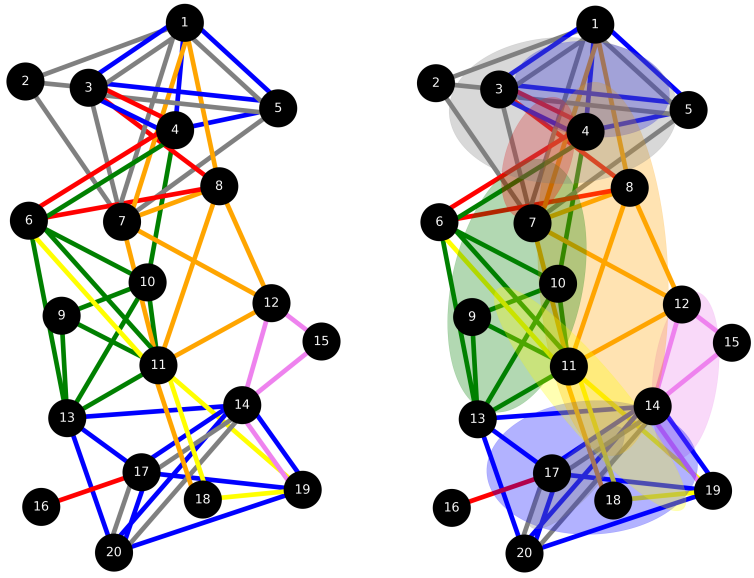

FIGURE 3.6 – Network model used in our performance evaluation

the implementation is in Python (running on pypy [1]) using deap [De Rainville 2012]. Unless specified, population size is set to 18 and number of generations at 18. Crossover probability is 0.9 and mutation probability is 0.05. N^c_{top} is set to 2, and N^s_{top} is set to 4. The simulation horizon is fixed to 10000 UT (this time period is sufficient to have our methods in the stationary regime). α_1, α_2 and α_3 are set to 1, 1 and 5 respectively throughout all evaluation experiments.

3.6.5 Performance metrics

The following performance metrics are computed during simulation for performance analysis purposes :

— Acceptance rate (ac, in %) : the percentage of successful virtual links requests out of all the requests that arrived during the simulation time or accumulatively with the time.

— Clique utilization (cu, in %) : bandwidth allocated at the links composing a clique divided by channel capacity, computed as $100 * \frac{\xi'(c)}{B_\lambda}$.

— Switch resource utilization regarding flow table utilization (su, in %) : flow table utilization at the nodes divided by the initial flow table size, computed as $100 * \frac{L'_v}{L_v}$.

— Switch resource utilization regarding group table utilization (gu, in %) : group table utilization at the nodes divided by the initial group table size, computed as $100 * \frac{M'_v}{M_v}$.

— Computation time (in second) : the average computation time for one request.

1. http ://pypy.org/

3.6.6 Performance results

3.6.6.1 Coping with wireless links Interference

The objective is to assess how efficient are our methods in reducing and avoiding wireless links interference. To this end, β_2 and β_3 are set to 0 in a first place. We compare the effect of setting β_1 to 1 versus to 0.

In fact, when embedding virtual links requests, our methods favor links belonging to cliques with limited number of members, introducing, by the way, in their surroundings less interference and, hence, preserving the overall available bandwidth. This is clearly shown in Figure 3.7 which focuses on the clique utilization of two groups of cliques : small cliques with a small number ($3 \sim 6$) of interfering links and large cliques with a high number ($8 \sim 11$) of links. When activating interference reduction, the bandwidth consumed by large cliques is decreased in contrary to small cliques. As expected a portion of the bandwidth consumed by a large-size clique is transferred to smaller-size cliques : small cliques experience an increase in clique utilization while large cliques get less loaded.

Favouring small-size cliques may lead to longer data paths and hence more resources are needed to support the virtual links being embedded. Since the acceptance rate is improved with such a strategy, this means that this latter increase is compensated by the resource that are preserved thanks to interference reduction. With the considered network and load models, our experiments show a slight increase around 1% on the average length of selected data paths.

3.6.6.2 Assessing the gain brought by the Multicast advantage

The objective is to quantify the gain in resource usage that our methods achieve by exploiting the multicast advantage when embedding point-to-multipoint virtual links. To this end, we compare our embedding methods to a variant of our methods where the multicast advantage is inhibited.

In fact, with multicast advantage, in the ILP formulation, we use $f(v, \lambda)$ to represent all flows that go from v and through λ, e.g. $f((v, u_1), \lambda)$ and $f((v, u_2), \lambda)$. The dissolving of this relation leads to the disabling of multicast advantage. For GA, cancelling the detection of multicast advantage opportunities and giving the same fitness values to a transmission via two or multiple distinct channels and a transmission via the same channel, the disabling of multicast advantage is achieved.

Again, β_2 and β_3 are set to 0 in a first place. β_1 is set to 1 as we have shown that interference should be taken into consideration for the modeling.

The clique utilization of the 9 cliques is presented in Figure 3.8. We see that disabling the multicast advantage induces extra bandwidth consumption that overloads all cliques and causes significantly more embedding failures. Therefore, in our following experiments, multicast advantage will be always enabled.

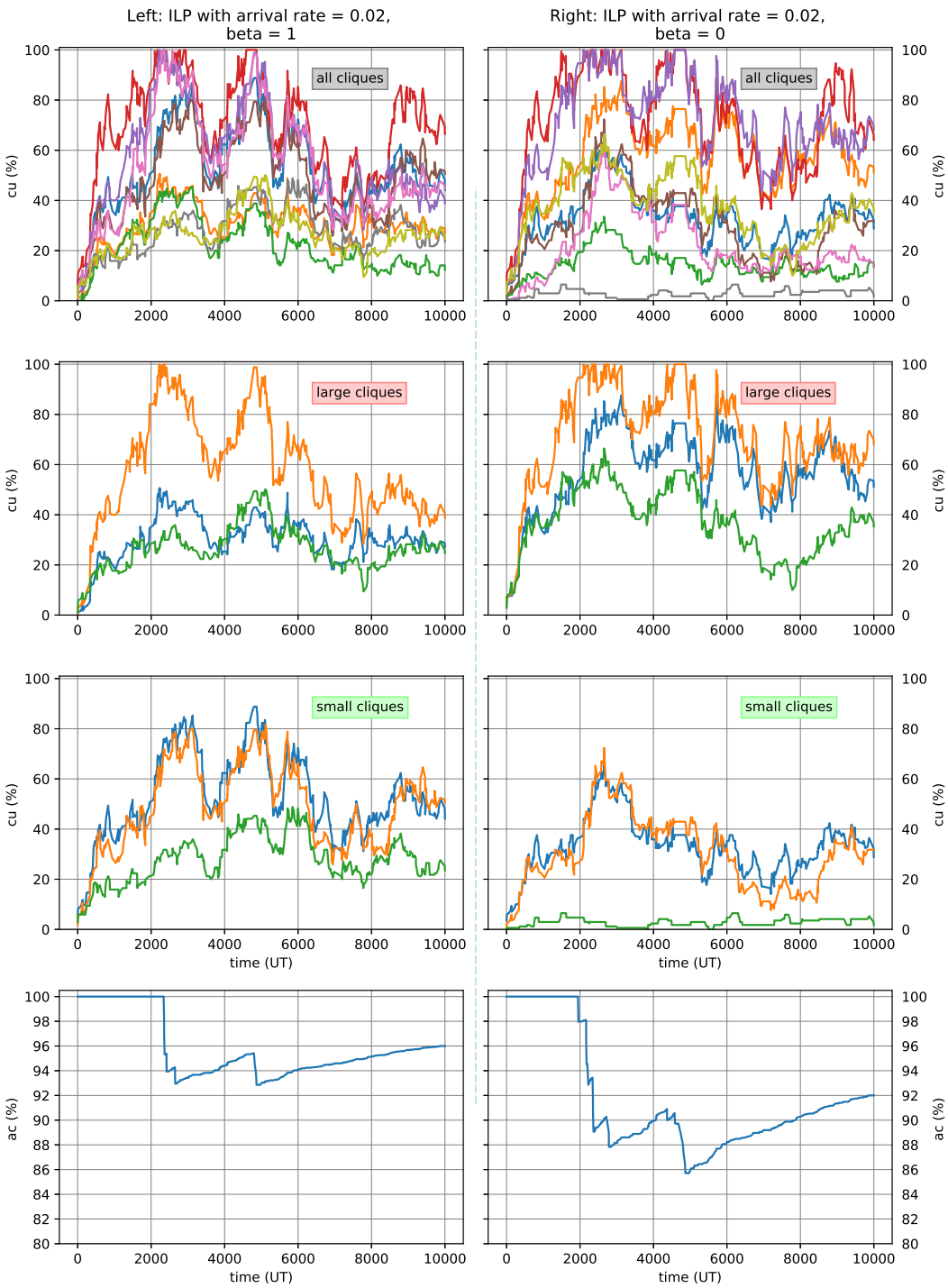

FIGURE 3.7 – Clique utilization (all cliques, large cliques and small cliques) and acceptance rate with $\beta_1 = 1$ v.s $\beta_1 = 0$. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA.

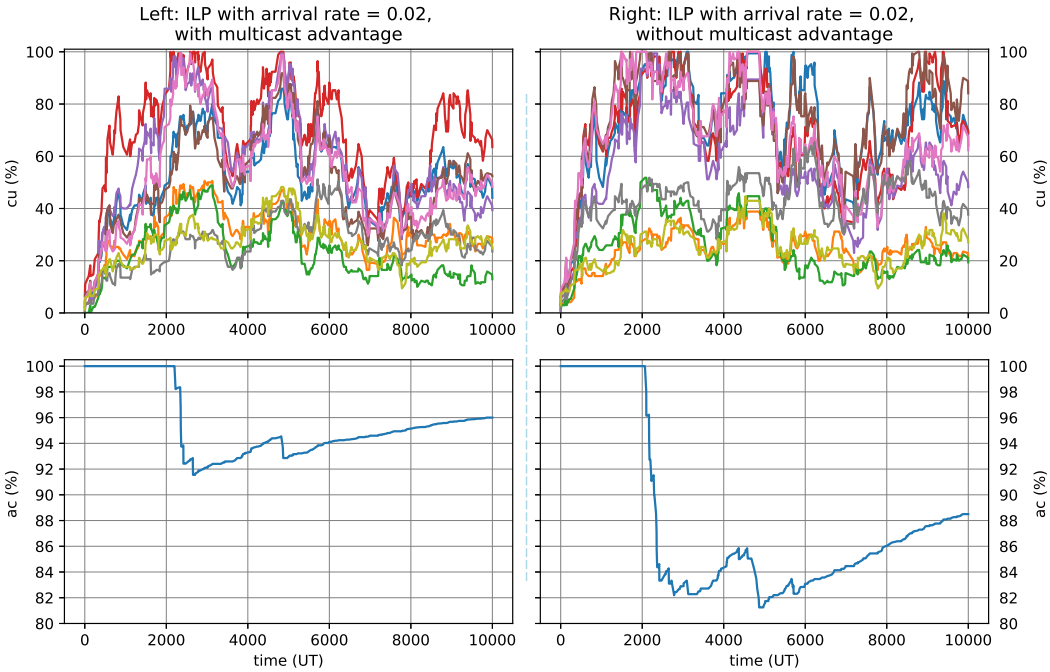

FIGURE 3.8 – Clique utilization and acceptance rate with and without multicast advantage. Computed with ILP. Arrival rate = 0.02. $\beta_2 = 0$, $\beta_3 = 15$. Similar results are obtained with GA.

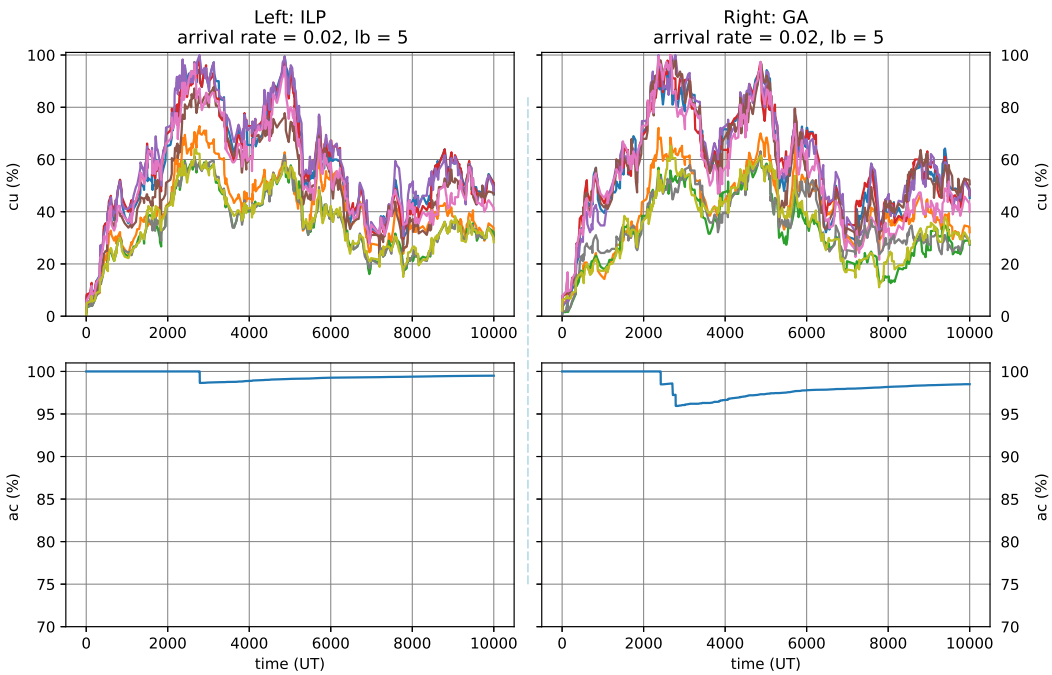

FIGURE 3.9 – Clique utilization balancing with ILP and GA. Arrival rate = 0.02. $\beta_3 = 15$.

3.6.6.3 Clique utilization balancing

By setting β_2 to 5, we activate clique load balancing. To show the effect of clique load balancing, Figure 3.9 shows that the clique utilizations have now much less disparity, with ILP as well as GA, compared to Figure 3.7 and 3.8 where $\beta_2 = 0$, leading to an improved acceptance rate (99.5% for ILP and 98.5% for GA) .

3.6.6.4 Switch resource consumption and balancing

To show how flow table resource is consumed and balanced and in which manner this might impact , we set the initial flow table size to 90, and compared the results of $\beta_3 = 0$ versus $\beta_3 = 15$ using ILP, as is shown in Figure 3.10. Apart from a much better balancing of flow table resource utilization, we also see an improved acceptance rate (99.5% v.s. 94.5%). Hence, flow table resource balancing should be activated. The effect of switch resource balancing of GA is shown in Figure 3.11. We can see that GA is less effective than ILP in switch resource balancing.

Figure 3.12 presents the group table utilization, computed with ILP and GA. We observe that with ILP and GA, thanks to the multicast advantage, the group table consumption remains very limited despite the successful mapping of point-to-multipoint virtual links. With ILP-PS, more group table entries are consumed, but always within a reasonable limit. As expected, for the considered simulation

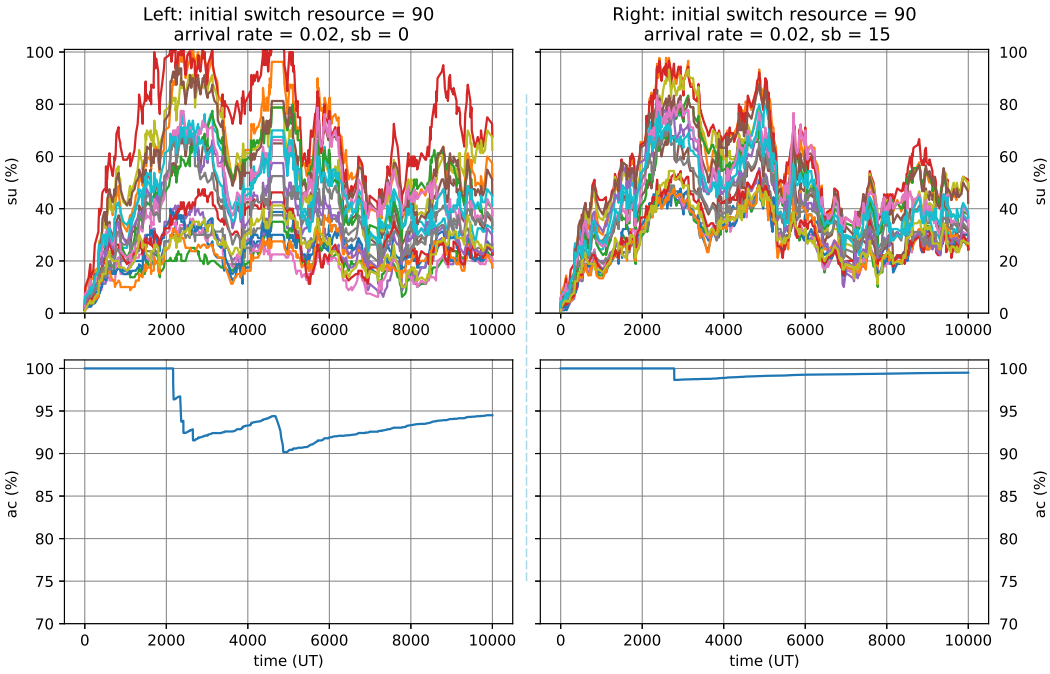

FIGURE 3.10 – Switch resource consumption of all nodes, computed with ILP, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$

model, group table entries are abundant in comparison to embedding needs. As a consequence, it does not play a decent role in the selection of the data paths. Hence, there is no need to balance its utilization in the formulation. Finally, if the path splitting of ILP is enabled, we observe a significant increase in switch resource consumption.

3.6.6.5 Path splitting

With path splitting, the acceptance rate improves slightly. However, if we set the initial switch resource to relatively scarce, then we observe that ILP-PS delivers much worse results than ILP and GA. Figure 3.13 depicts this with initial flow table size set at 90. We see that ILP-PS consumes more group table entries and flow table entries, and the latter will lead to more embedding failures of ILP-PS compared to ILP. Hence, path splitting should not be activated in a situation where switch resources are not abundant.

3.6.6.6 Comparison with heuristic algorithms

Here we present the results of SPH using different link metrics, as is shown in 3.14. We see that the heuristic with all 3 link metrics has a lower acceptance rate than ILP, ILP-PS and GA. However, Metric-1 has the best performance, while

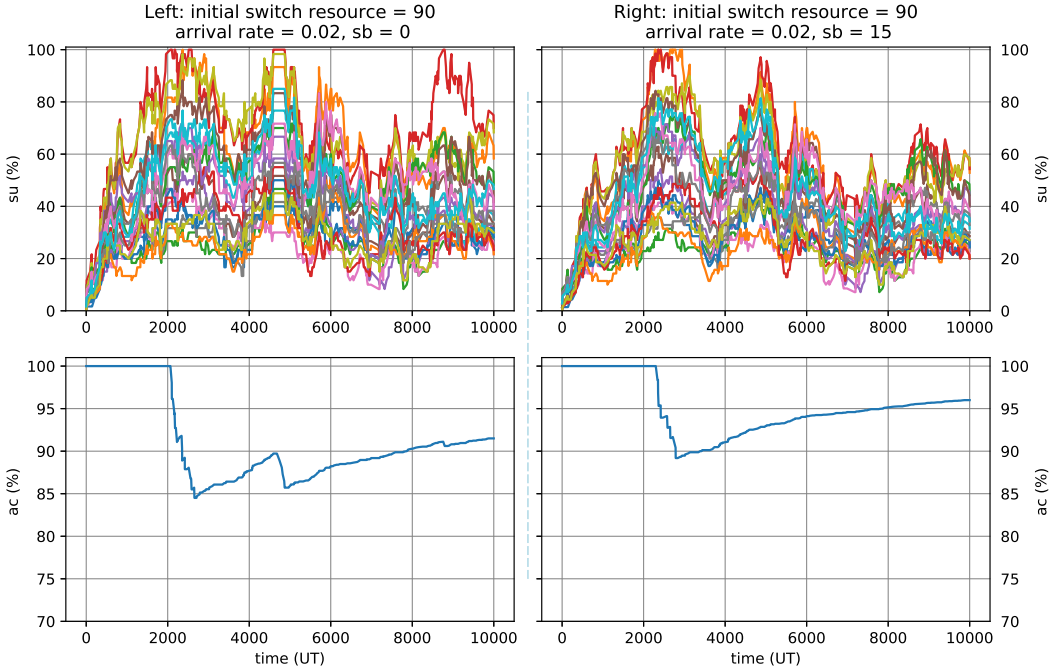

FIGURE 3.11 – Switch resource consumption of all nodes, computed with GA, with initial flow table size set at 90. Arrival rate = 0.02. $\beta_2 = 5$.

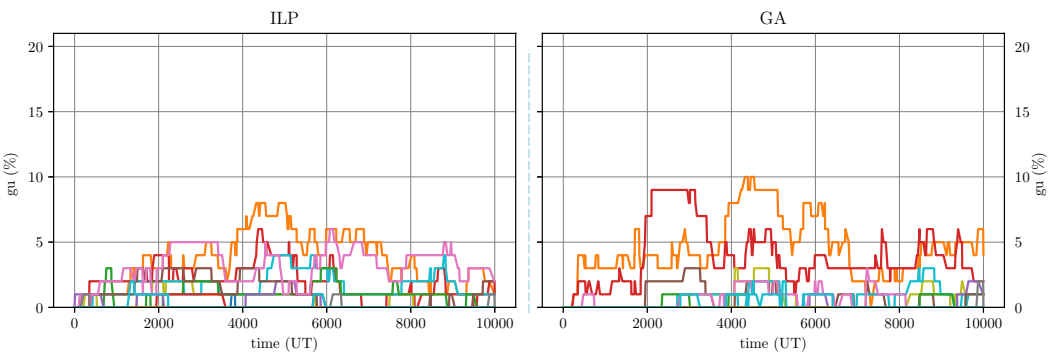

FIGURE 3.12 – Group table consumption of ILP and GA. Arrival rate = 0.02. $\beta_2 = 5$, $\beta_3 = 15$.

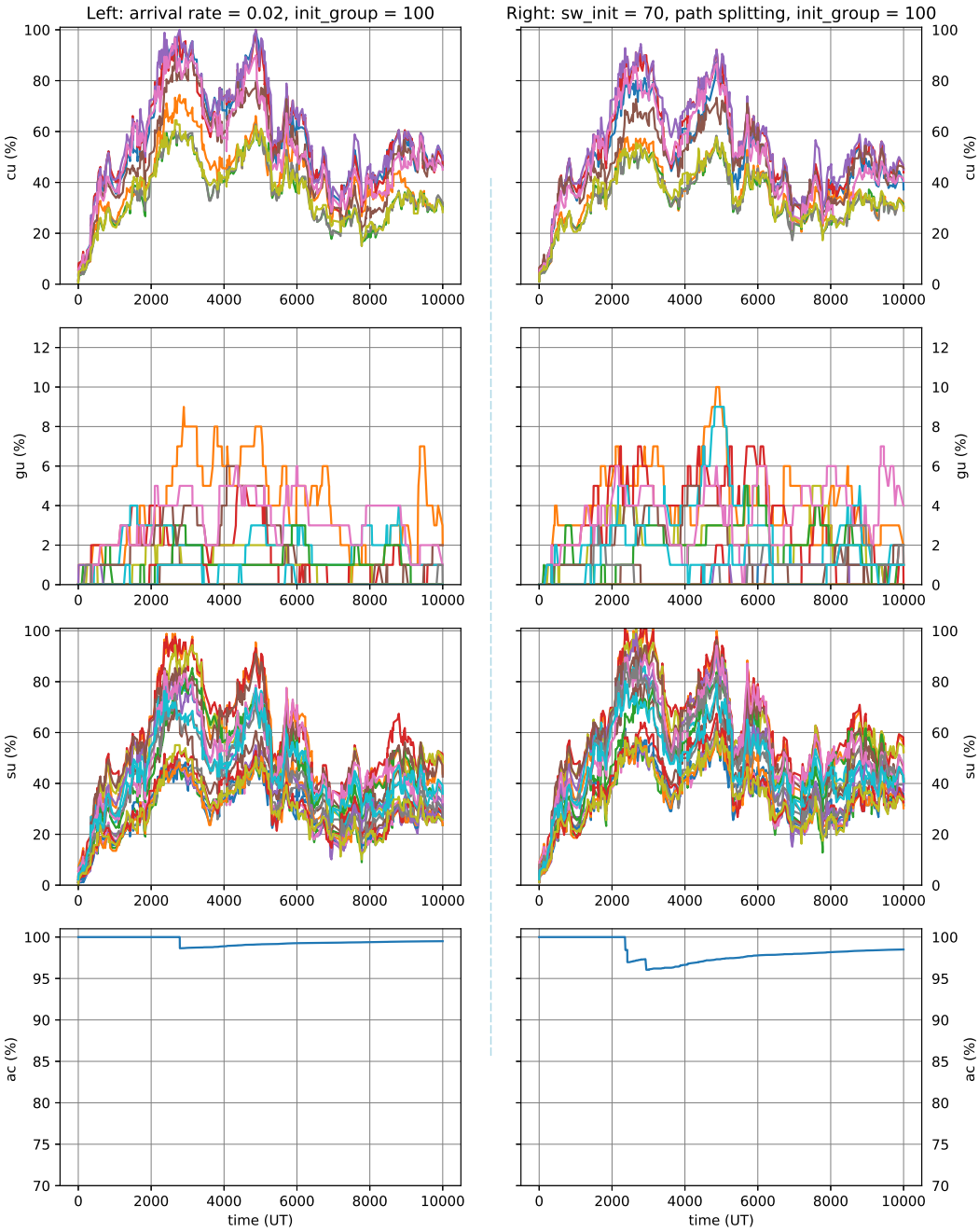

FIGURE 3.13 – Switch resource consumption with ILP and ILP-PS. The initial flow table size is set to 90. Arrival rate = 0.02. $\beta_2 = 5$, $\beta_3 = 15$.

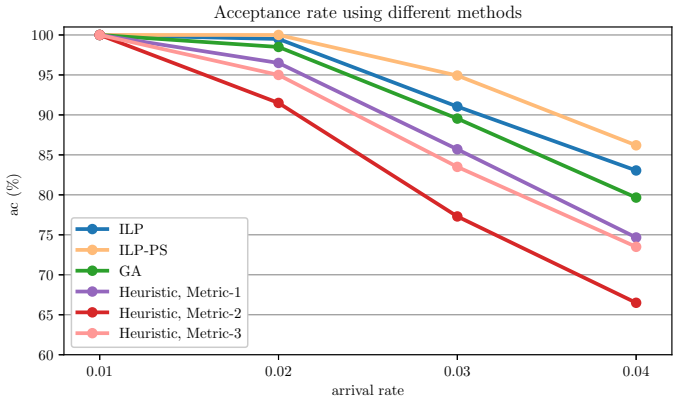

FIGURE 3.14 – Acceptance rate of the heuristic using 3 different link metrics as well as ILP, GA and ILP-PS, for different arrival rates. $\beta_2 = 5$, $\beta_3 = 15$.

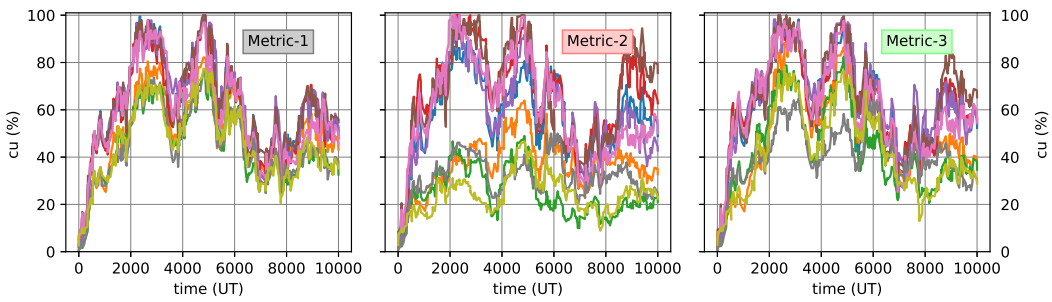

FIGURE 3.15 – Clique load balancing of heuristic using the 3 different link metrics

Metric-2 has the worst. Also, Figure 3.15 and Figure 3.16 show that Metric-1 leads to a relatively good balancing both in clique load and switch resource utilization, while Metric-3 leads only to a decent balancing in clique load. Metric-2, as expected, leads to an unbalanced situation, and hence gives the worst acceptance rate.

3.6.7 Method selection : ILP vs GA

Our chapter proposes two methods for solving the problem of virtual link resource allocation in software defined multi-radio multi-channel multi-hop wireless networks. Apart from the apparent support for path splitting (ILP yes and GA no), there are two important criteria to consider when choosing the method to be applied : (1) accuracy (leading to optimal acceptance rate) and (2) computation time. Figure 3.14 shows the acceptance rate using different methods, and we can see that ILP-PS (ILP with path splitting enabled) gives slightly better results than ILP and GA. Figure 3.17 and Figure 3.18 present the acceptance rate and the computation time obtained with GA when considering different population and generation sizes. For the considered network model, GA with a population of 18 individuals and 18

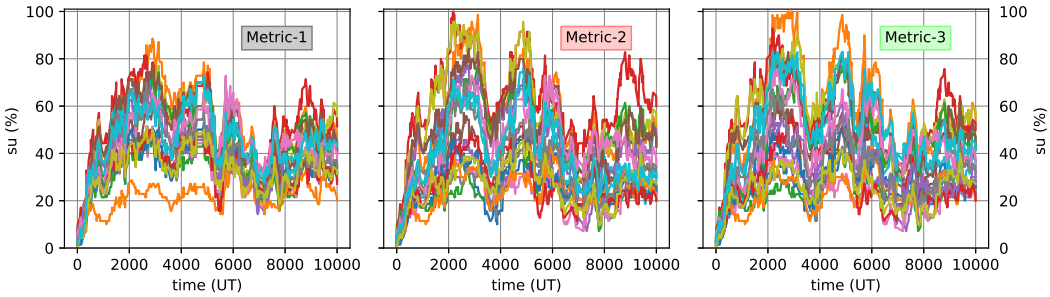

FIGURE 3.16 – Switch resource balancing of heuristic using the 3 different link metrics

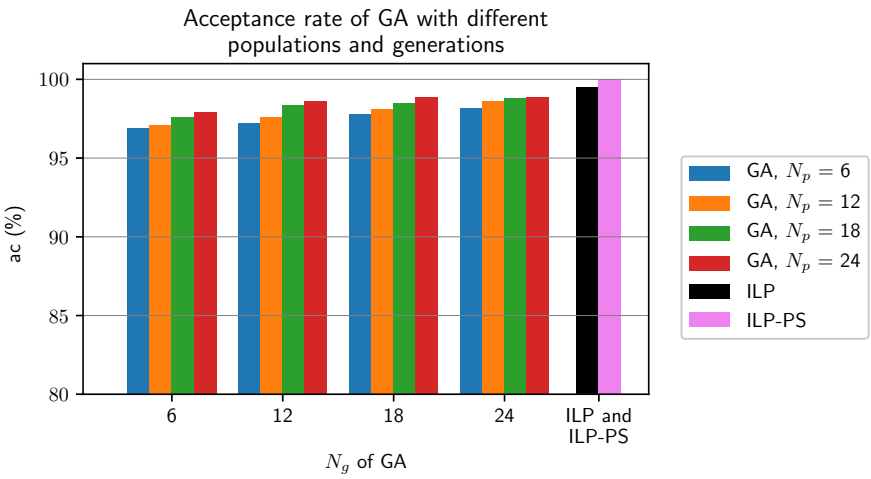

FIGURE 3.17 – Acceptance rate of GA by varying number of generation and size of population, compared with ILP and ILP-PS. Arrival rate = 0.02

generations lasts 80% of the computation time of ILP. With smaller population and generation size (e.g. 12 and 12), the computation time can be significantly reduced (less than 1/10 of ILP), bringing only minor degradation to the acceptance rate (∼1.5%). Our experiments show that for larger network models, GA shows a significant advantage in computation time compared to ILP. On the contrary, with ILP-PS, as the search space explodes, the computation time can be several folds of that of ILP and hence much more than GA. This is another major shortcoming of ILP-PS, besides more consumption of switch resources.

3.7 Conclusion

In this chapter, we developed an Integer-Linear programming method and a genetic algorithm method for the resource allocation of multiple virtual links in wireless software defined multi-radio multi-channel multi-hop networks. In comparison

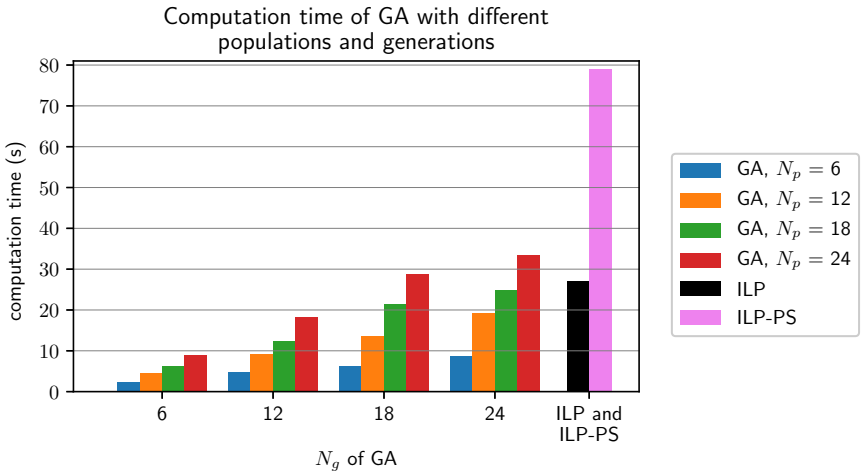

FIGURE 3.18 – Average Computation Time of each request of GA by varying number of generation and size of population, compared with ILP and ILP-PS.

to existing works, the main contribution of our proposals lies in the conjunction of the following features : (1) the support of point-to-multipoint virtual links in addition to point-to-point virtual links, and, in a wireless context, how to benefit from the multicast advantage to gain in bandwidth consumption, (2) the consideration of switching resources in the allocation of resources in addition to the bandwidth of channels. Through our evaluations, we show that both of our proposed methods work well. More interestingly, we investigated how the consideration of interference, multicast advantage as well as resource balancing could positively impact the embedding results, and, how the two proposed methods differ in performance and computation time.

Chapitre 4

Re-embedding Schemes for Evolving Virtual Links in SDN Networks

This chapter presents a re-embedding scheme for evolving virtual links in SDN networks. In the previous chapter, VL requests are assumed to be static and don't evolve over time. The requests are sent to the SDN controller and then VLs are instantiated until the end of their lifetime. In this chapter, however, we consider that VLs requests could evolve over time, e.g. (increasing or decreasing bandwidth requirement, destination nodes that join or leave). This makes total sense in that applications that run over SDN substrate can be in a quite dynamic environment, bringing this dynamicity to requests. To meet the evolving requests, reconfigurations with OpenFlow Modification messages are involved. However, frequent reconfigurations lead to more vulnerability to service disruption. Hence it might be desirable to preserve existing embedded VLs and only reconfigure extra paths that might influence.

Two algorithms are proposed to investigate the problem, one in ILP and another in GA. This chapter considers the wired case. However, wireless case can be easily derived, with interference and multicast advantage included in the modelling.

4.1 Introduction

For the virtual network (VN) embedding problem, most researches from the literature considers only static VN requests, i.e. the VN request and the demand for resources is fixed and does not evolve over time.

However, VN requirement may change during the VN lifetime. Figure 4.1 illustrates an already instantiated initial embedding, with node A as the source, and node G as destination. One possible change to the VN requirements is to add node B and E as new destinations. As is illustrated in Figure 4.2, one approach is to recompute from scratch a new tree. In this way, minimum resource consumption is guaranteed. However, this involves six OpenFlow Modification messages overhead : the more modification to the existing embedded VN, the more likely it is to experience service disruption. Another possible solution is illustrated in Figure 4.3, which is based on the existing embedding tree, and only adds one reconfiguration overhead (service disruption). However, this consumes more network resources.

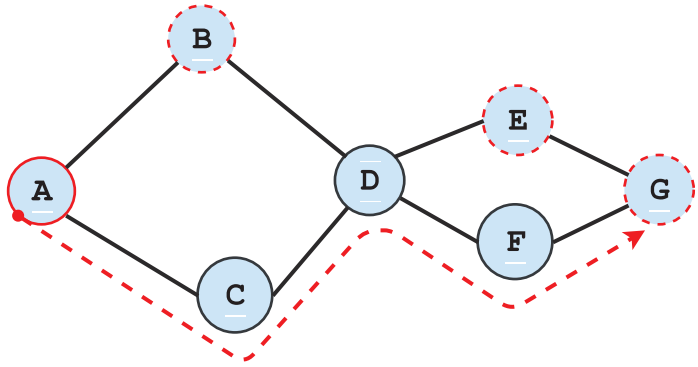

FIGURE 4.1 – Initial Embedding, with node A as the source, and node G as destination. The evolving of the request requires that node B and E should be added as destinations.

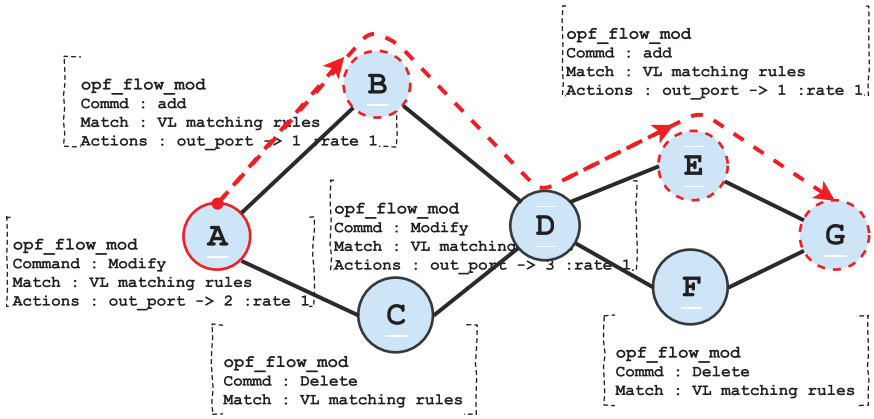

FIGURE 4.2 – One possible solution : recomputation of a new tree which minimizes resource consumption, but this would introduce service disruption

Excessive OpenFlow Modification messages from SDN controller might cause SDN controller overload, bandwidth overhead as well as service disruption. On the other hand, network resources are limited, and should be efficiently assigned. This chapter proposes an ILP and GA based formulation of the virtual link re-embedding problem that captures the incurred service disruption.

This chapter is organized as follows. Section 4.2 reviews previous work from the literature that are related to re-embedding of evolving virtual links. Then, Section 4.3 describes the system model and problem statement. Section 4.4 describes the ILP formulation and Section 4.5 describes the genetic algorithm formulation. Section 4.6 presents the performance analysis of the proposed methods. Finally, Section 4.7 concludes the chapter.

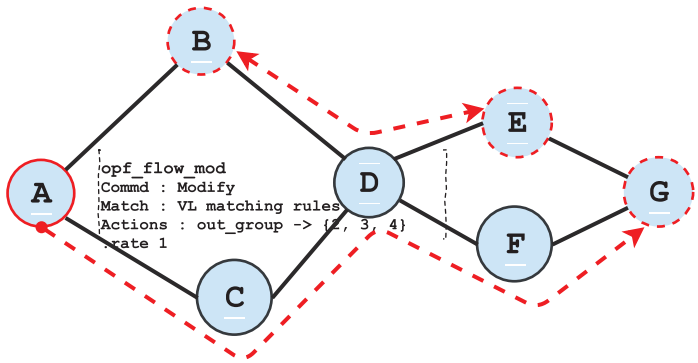

FIGURE 4.3 – Another possible solution : a solution that is based on the existing embedding tree, which brings minimum reconfiguration overhead and minimizes service disruptions

4.2 Related work

There are few works who consider the re-embedding of evolving virtual networks. Table 4.2 is an exhaustive list of comparison. Our work is the first to consider the SDN substrate network, with an emphasis on point-to-multi-point communications, as well as switch resources. Also, only in our work that GA is used to solve the problem.

4.3 System model and problem statement

4.3.1 Substrate network model

The SDN substrate network is modelled as a bidirectional graph $G = (V, E)$ where N is the set of SDN nodes or switches and $E(E \subseteq V \times V)$ the set of physical links which operate in full-duplex mode. To each node $i \in V$, is associated a switching capacity U_i, which is the maximum number of entries (i.e. size limit) of its flow table. The current size of node i flow table is denoted by U'_i. Similarly, M_i and M' denote respectively the maximum and current size of the group table of node i. Each link $(i, j) \in E$ is weighted by its bandwidth B_{ij}. Links are assumed to have the same characteristics in both directions, i.e. $B_{ij} = B_{ji}$. The bandwidth that is currently assigned at link (i, j) by already admitted VLs is denoted by B'_{ij}.

4.3.2 Virtual network request model

A VN request denoted as K is composed of a set of $|K|$ VLs. Each VL k is characterized by :

— a source node $s_k \in V$, and a set of destination nodes $T_k \in V \backslash \{s_k\}$ (when $|T_k| = 1$, the VL is point-to-point, otherwise it is point-to-multipoint) ;

— a bandwidth requirement of b_k ;

TABLE 4.1 – Classification of virtual link re-embedding schemes in the literature

Existing Works	VL type	SDN substrate	Evolving Resources	Proactive or Reactive	Solving Method
[Jmila 2015]	P2P	No	Nodes resource, Link bandwidth	Reactive	Heuristic
[P. N. 2013]	P2P	No	Nodes resource (CPU), Link bandwidth	Reactive	ILP
[Zhu 2006]	P2P	No	Node CPU, Link bandwidth	Proactive	Heuristic
[Cai 2010]	P2P	No	Node CPU, Link bandwidth	Reactive	ILP, Heuristic
Our work	P2P P2MP	Yes	Node switch resources, Link bandwidth	Reactive	ILP, Genetic Algorithm

VN requests are treated by the network resource allocation algorithm that computes physical paths as well as resources to assign along these paths.

4.3.3 Evolving virtual links request model

An evolving VLs request can be seen as a VN request that consists of a set of already embedded running VLs from the same or different VNs, each with new characteristics. We use K_o and K_e to represent the original and the evolved VLs request respectively. In other words, these two requests have the same VLs, but with different characteristics. Each VL $k \in K_o$ has initially been embedded by the resource allocation algorithm and eventually been re-embedded by the algorithm in charge of handling the evolved VLs request. It is currently mapped to one or multiple original substrate paths represented by F_{ok}, associated with the bandwidth allocation at each PL (physical link) $(i, j) \in F_{ok}$ denoted by $f_{ok}(i, j)$ as well as the number of flow and group table's entries consumed at each traversed node $i \in F_{ok}$, which are represented by $l_{ok}(i)$ and $n_{ok}(i)$ respectively. Similarly, as soon as an evolved VLs request arrives, the re-embedding process remaps each VL $k \in K_e$ to new substrate paths F_{ek} in such a manner that the bandwidth requirements are satisfied as well as substrate capacity constraints. Let $f_{ek}(i, j)$ refer to the amount of bandwidth used at PL $(i, j) \in F_{ek}$ by k. Likewise, we receptively denote by $l_{ek}(i)$ and $n_{ek}(i)$ the number of flow or group table entries that are allocated at each traversed node $i \in V$ to support VL k.

There are four basic different scenarios for an evolving VL : new destination node arrival or departure, bandwidth requirement increase or decrease. Since node departures and bandwidth requirement decrease are handled by releasing resources, next, we focus on the two other scenarios, namely the case of an increase of the VL's requirement bandwidth and the case of destination node addition.

4.3.4 Virtual links re-embedding objective

We use binary $g_{ok}(i, j)$ and $g_{ek}(i, j)$ to represent original and evolved mapping results of a VL k respectively, at each PL (i, j). In detail, $g_{ok}(i, j) = 1$ if k is originally embedded onto PL (i, j), otherwise $g_{ok}(i, j) = 0$. Likewise, $g_{ek}(i, j) = 1$ if VL k is re-embedded onto PL (i, j), otherwise $g_{ek}(i, j) = 0$.

In this chapter, the re-embedding cost denoted as C consists of two criteria : i) the embedding cost in terms of the amount of bandwidth and flow tabe entries that are consumed ; ii) the reconfiguration overhead (service disruption) which captures service disruption experienced by evolving VLs. Formally C is expressed as follow :

$$C = \alpha * \sum_{k \in K_e} \left(\sum_{i \in V} l_{ek}(i) + \sum_{(i,j) \in E} f_{ek}(i,j) \right) + \beta * \sum_{k \in K_o} \left(\sum_{(i,j) \in E} g_{ok}(i,j) * (1 - g_{ek}(i,j)) \right)$$

Finally, we re-embed VLs request when they evolve, with the objective of successfully re-embedding it while minimizing C. α and β are constants aimed at scaling the two terms to comparable magnitudes, or for giving one criterion more importance than the other.

4.4 ILP for evolving VLs problem

In previous section, we stated the evolving VLs problem and we also defined the considered system model. In this section, we rely on this system model to describe the ILP formulation based on multi-commodity flow that we propose to optimally solve the problem. Requests arrive and are treated in sequence with no information on future requests (i.e. online). Each request gathers multiple concurrent virtual link sub-requests, each concerning a point-to-point or point-to-multipoint virtual link with bandwidth requirements. Below, the variables are described ; then, the considered objective function is defined, finally the problem constraints are listed.

4.4.1 Resource-related assignment variables

The output of our re-embedding problem is the set of paths (with the bandwidth allocations at each supporting PL and the number of flow and group table entries consumed at each traversed node) that support each of the VLs that composes the evolving VLs request K_e. It is worth noting that since VLs may be point-to-multipoint, it is likely that resource allocations will be mutualised close to the source and as we get closer to destinations, they will tend to be more and more dedicated to specific destinations. As a consequence, basic assignment variables are related to a specific destination of a VL. In our model, we distinguish the following variables :

— $f^t_{ek}(i,j)$ is an integer variable that represents the bandwidth allocated at link (i,j) to the packets of VL k that are flowing from the origin node s_k to a destination node t. More generally, $f_{ek}(i,j)$ refers to the amount of bandwidth used on link (i,j) by the VL k, whatever the destination. It is set to the maximum of $f^t_{ek}(i,j)$ for all $k \in K_e$.

— $l_{ek}(i)$ is an binary variable that indicates the switching resources consumed by VL k at node i. It is expressed as the number of entries that are installed in node i flow table to support VL k with the assumption that all entries consume the same amount of resources regardless of the complexity of the

match operation and the related *instructions* to perform. In this work, we assume that each VL consumes 1 flow table entry at each traversed node. A node is traversed by a VL if at least one of its adjacent physical link supports VL. Formally :

$$\forall k \in K_e, \forall i \in V, \forall (i,j) \in E : g_{ek}(i,j) \leq l_{ek}(i) \tag{4.1}$$

$$\forall k \in K_e, \forall i \in V : l_{ek}(i) \leq \sum_{\substack{j \in V \\ (i,j) \in E}} g_{ek}(i,j) \tag{4.2}$$

where $g_{ek}(i,j)$ is an intermediate binary variable that indicates if some bandwidth from link (i,j) is assigned to VL k or not. It is derived from another set of more focused intermediate variables $g^t_{ek}(i,j)$ that reflects whether the flow of packets of VL k destined to t is supported by the physical link (i,j) (i.e. $g^t_{ek}(i,j) = 0$ if $f^t_{ek}(i,j) = 0$ and $g^t_{ek}(i,j) = 1$ otherwise).

— $n_{ek}(i)$ is a binary variable indicating if a group table entry is assigned to VL k at node i. A group table entry is consumed by a VL at a node, if it is split (for multipath) or duplicated (for multicast). In other words, if at least two of its adjacent physical links support the VL. Formally (equations 4.3 and 4.4) :

$\forall k \in K, \forall i \in V, \forall (i,j) \in E :$

$$n_{ek}(i) \leq \left(\sum_{(i,j) \in E} g_{ek}(i,j) \right) - g_{ek}(i,j) \tag{4.3}$$

$\forall k \in K, \forall i \in V, \forall (i,j_1) \neq (i,j_2) \in E :$

$$g_{ek}(i,j_1) + g_{ek}(i,j_2) - 1 \leq n_{ek}(i) \tag{4.4}$$

4.4.2 Objective function and problem constraints

The objective is :

$$\textit{Minimize } C \tag{4.5}$$

and the subjects are below :

$$\forall (i,j) \in E : \sum_{k \in K_e} f^t_{ek}(i,j) \leq B_{ij} - B'_{ij} \tag{4.6}$$

$$\forall k \in K_e, \forall t \in T_{ek}, \forall i \in V :$$

$$\sum_{(i,j)\in E} (f^t_{ek}(i,j) - f^t_{ek}(j,i)) = \begin{cases} b_{ek} & if\ i = s_k \\ -b_{ek} & if\ i = t \\ 0 & else \end{cases} \tag{4.7}$$

$$\forall i \in V : \sum_{k\in K_e} l_{ek}(i) \leq U_i - U'_i \tag{4.8}$$

$$\forall i \in V : \sum_{k\in K_e} n_{ek}(i) \leq M_i - N'_i \tag{4.9}$$

Equation (4.6) ensures that the total bandwidth of all virtual links routed through a physical link doesn't exceed its residual capacity. Equation (4.7) is the multi-comodity flow constraint, which guarantees that all virtual links are mapped on a physical path. According to this constraint, the net flow to a physical node i is zero if it accommodates neither the source node s_k nor the destination node t of the virtual link. Equations (4.8) and (4.9) ensure that the total entries allocated to all virtual links routed through a physical node doesn't exceed its residual flow and group table capacity respectively.

4.5 Genetic algorithm for evolving VLs problem

The ILP-based algorithm might entail significant computation time and can fail to deliver the re-embedding results in a timely manner. In this section, we present a genetic algorithm based solution to address this aspect, which is based on the GA presented in the previous chapter (Chapter 3) with some major differences. The motivations of the general framework of each step of the GA presented in this chapter is the same as presented in the previous section. For this reason, explanations are simplified and more attentions are given to the implementation aspects. The overall workflow of our GA scheme is described in Algorithm 7.

4.5.1 Encoding scheme

An individual is a possible solution to an evolving request. We represent an individual χ as a vector of genes where each gene τ_k refers to the solution to an evolving virtual link $k \in K$. A tree connecting the source s_k to the destination nodes $t \in T_k$, is sufficient to represent a gene, in the same manner as presented in Chapter 3. Each tree is associated with switching resources and links bandwidth respectively allocated at each substrate node and link belonging to this tree. Accordingly, χ_o refers to the original VLs embedding.

Algorithm 7: Overall workflow of GA-based re-embedding scheme

Input : $G(V,E); \chi_o; K_e; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
$\gamma_{comm}^{min}; \gamma_{comm}^{max}; \gamma_o^{min}; \gamma_o^{max}; \gamma_{normal}^{max}; N_p; N_g; cxPB; mutPB$;
Output: $\chi = [\tau_1, \tau_2, ..., \tau_{|K|}]$ (i.e. the best individual)

1 **begin**
2 $P_0 \longleftarrow$ `InitPop`$(\mathrm{G}, \chi_\mathrm{o}, \mathrm{K_e}, \mathrm{N_p}, \gamma_\mathrm{o}^{\mathrm{min}}, \gamma_\mathrm{o}^{\mathrm{max}}, \gamma_{normal}^{max}, W)$
3 $P \longleftarrow P_0$
4 **for** ($j_g = 0;\ j_g < N_g;\ j_g++$) {
5 **for** ($j_p = 0;\ j_p < N_p;\ j_p++$) {
6 $(\chi^a, \chi^b) \longleftarrow$ `TournamentSelection`(P)
7 $\chi^c \longleftarrow$ `Crossover`$(G, \chi_o, K_e, \chi^a, \chi^b,$
$cxPB, \gamma_{comm}^{min}, \gamma_{comm}^{max}, \gamma_o^{min}, \gamma_o^{max}, \gamma_{normal}^{max}, W)$
8 $\chi^d \longleftarrow$ `Mutation`$(G, K_e, \chi^c, mutPB, \gamma_{mut}, W)$
9 $P \longleftarrow P \cup \chi^d$
10 $P \longleftarrow$ `PopulationSelection`$(P, \mathrm{N_p})$
11 $\chi \longleftarrow$ `SelectBestIndividual`(P)

4.5.2 Initial population

To construct trees on graphs, it's necessary to define link costs so that graph algorithms could be employed to calculate shortest paths, Steiner trees etc. In our case, we define the link cost w_e of link e as :

— $w_e = \infty$(infinity), if the link has not enough bandwidth supporting evolving VLs (i.e. $residual_bw(e) < b_k$ for all $k \in K_e$), or if the link has one end node with no flow table entries left.

— $w_e = \frac{1}{residual_bw(e)}$ otherwise, i.e. the link cost is inversely proportional to its residual bandwidth.

With the link costs W defined, we come to the generation of an initial population (Algorithm 7 - line 2). It is computed by generating a given population size (N_p) with each member of this population encoding an individual representing a possible solution. Two objectives should be achieved for this task : (1) Diversity among populations, in order to avoid premature local convergence. (2) Similarity to original links χ_o.

To achieve the first objective, the cost of each link in the substrate network is multiplied by a random factor in the range of $[1, \gamma_{normal}^{max}]$. To achieve the second objective, for those links constituting the original VL χ_o, the cost of each of them is multiplied by a small random factor (e.g. around 0.2) in the range of $\left[\gamma_o^{min}, \gamma_o^{max}\right]$. Afterwards, we compute the minimum Steiner tree as a routine to build the tree representation of each gene k of an individual χ.

Algorithm 8: Initial population computation

Input : $G(V,E); \chi_o; K_e; N_p; \gamma_o^{min}; \gamma_o^{max}; \gamma_{normal}^{max}; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
Output: $P_0 = \{\chi^i, \forall i \in \{1, ..., N_p\}\ with\ \chi^i = [\tau_1^i, \tau_2^i, ..., \tau_{|K|}^i]\}$

```
1  begin
2      P_0 ⟵ ∅
3      G'(V', E') ⟵ Clone(G(V, E))
4      W' ⟵ Clone(W)
5      for ( i = 0; i < N_p; i++ ) {
6          foreach k ∈ K_e do
7              W' ⟵ Clone(W)
8              foreach e ∈ χ_o[k] do
9                  W'[e] ⟵ W'[e] × Random(γ_o^min, γ_o^max)
10             foreach e ∈ E' − χ_o[k] do
11                 W'[e] ⟵ W'[e] × Random(1, γ_normal^max)
12             τ_k^i ⟵ ComputeSteinerTree(G', W'[e], s_k, T_k)
13             χ^i[k] ⟵ τ_k^i
14         P_0 ⟵ P_0 ∪ χ^i
```

4.5.3 Fitness function

After creating the initial population, each individual is evaluated and assigned a fitness value according to a fitness function. The optimality of a solution is defined by its corresponding fitness value. Equations 4.10 define our fitness function.

$$F(\chi) = C(\chi) * \hat{F}_{bw}(\chi) * \hat{F}_{sw}(\chi) \tag{4.10}$$

where

$$\hat{F}_{bw}(\chi) = 1 + 100 \sum_{e \in E} \delta(\chi, e)$$
$$\hat{F}_{sw}(\chi) = 1 + 100 \sum_{v \in V} \theta(\chi, v)$$

In the fitness function, $C(\chi, \chi_o)$ is the objective function defined in Section 4.3-D, computed from χ and χ_o. The constraints on link bandwidth resources and switching resources are also included in the fitness function of GA, i.e the $F_{bw}(\chi)$ and $F_{sw}(\chi)$ multiplier, in the same manner as the GA presented in Chapter 3, so that solutions that are less infeasible are favoured for survival than more infeasible ones.

4.5.4 Selection of parents and crossover scheme

In this stage (Algorithm 7 - Line 6), chromosomes from a population are selected for reproduction (crossover), detailed in Algorithm 9. We use tournament selection of size 3 to select a pair of chromosomes as the parents to produce an offspring by applying crossover operator between them, with the crossover probability $cxPB$. The strategy is summarized in Algorithm 9.

According to the definition of the fitness function, the “better" individual has higher probability of being selected as a parent and survive. Thus, the common links between two parents are more likely to represent the “good" traits. In our case, this consists in passing common traits from parents to offspring according to a specific logic called *Similitude* (Algorithm 9 - Line 4). Let $\chi^a = [\tau_1^a, \tau_2^a, ..., \tau_K^a]$ and $\chi^b = [\tau_1^b, \tau_2^b, ..., \tau_K^b]$ be the selected parents. The crossover operator generates a child $\chi^c = [\tau_1^c, \tau_2^c, ..., \tau_K^c]$ by identifying the same links between τ_k^a and τ_k^b for each $k \in K$, and retaining these common links in τ_k^c with a high probability. To this end, in the substrate network, the link costs of those common links are multiplied by a random near-to-zero factor in the range of $\left[\gamma_{comm}^{min}, \gamma_{comm}^{max}\right]$ (e.g. around 0.1). At the same time, it is always important to guarantee the similarity to original links as well as the diversity of solutions. To this end, the link cost of original links are multiplied by a small random factor in the range of $\left[\gamma_o^{min}, \gamma_o^{max}\right]$. Other links are multiplied by a random factor in the range of $[1, \gamma_{normal}^{max}]$. Afterwards, we compute the minimum Steiner tree on the substrate network with adjusted link costs.

4.5.5 Mutation scheme

When a new offspring is produced, the mutation operation is performed according to the mutation probability $mutPB$ (Algorithm 7 - Line 8). It consists in mutating the produced offspring of the crossover stage, χ^c, into a slightly different one, χ^d, as in shown in Algorithm 10. The procedure randomly selects a link in the tree and gives a high multiplier factor γ_{mut} (with $\gamma_{mut} \gg 1$) to its link cost. Other unselected links in the offspring χ^c are given link cost of 0. Steiner tree is then computed on this substrate graph. In this way, all existing links in the offspring except the selected link are certainly included in the mutated offspring, and the selected link would be changed to another path if it exists.

4.6 Performance evaluation

We firstly introduce the simulation settings, before presenting our main results.

4.6.1 Simulation settings

Our algorithms are applied to a real network topology with different randomly generated VLs requests. The considered experimental set up is described hereafter.

Algorithm 9: Crossover scheme

Input : $G(V,E); \chi_o; K_e; \chi^a; \chi^b; cxPB;\ \gamma^{min}_{comm}; \gamma^{max}_{comm}; \gamma^{min}_{o}; \gamma^{max}_{o}; \gamma^{max}_{normal};$
$W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
Output: $\chi^c = [\tau^c_1, \tau^c_2, ..., \tau^c_{|K|}]$

1 **begin**
2 $G'(V',E') \longleftarrow$ `Clone`$(G(V,E))$
3 **if** $($`Random`$(0,1) < cxPB)$ **then**
4 **foreach** $k \in K$ **do**
5 $W' \longleftarrow$ `Clone`(W)
6 $\tau^c_k \longleftarrow$ `Similitude`(τ^a_k, τ^b_k)
7 **foreach** $e \in \tau^c_k$ **do**
8 $W'[e] \longleftarrow W'[e] \times$ `Random`$(\gamma^{min}_{comm}, \gamma^{max}_{comm})$
9 **foreach** $e \in \chi_o[k]$ **do**
10 $W'[e] \longleftarrow W'[e] \times$ `Random`$(\gamma^{min}_{o}, \gamma^{max}_{o})$
11 **foreach** $e \in E' - \chi_o[k] - \tau^c_k$ **do**
12 $W'[e] \longleftarrow W'[e] \times$ `Random`$(1, \gamma^{max}_{normal})$
13 $\chi^c[k] \longleftarrow$ `ComputeSteinerTree`(G', E', W', s_k, T_k)
14 **else**
15 $\chi^c \longleftarrow$ `RandomChoice`(χ^a, χ^b)

4.6.1.1 Network model

We consider in this work a real network topology taken from the European Research Network GEANT with 41 network nodes and 60 links that connect the main European cities. Like [Mijumbi 2014], we assume that for each switch there are 125 entries in flow table and 50 in group table. We consider that each VL requires 1 entry in flow table of each traversed nodes and consumes 1 group table entry when it is multiplied over.

4.6.1.2 Requests model

We classify requests into two categories : i) *VN request* that refers to a resource allocation demand for a new VN ; ii) evolving VLs request that refers to a demand for upgrading allocations currently assignated to each VL. In detail, we assume that about 60% of incoming requests are evolving VLs requests, and the rest close to 40% as VN requests. VN requests arrive following a Poisson distribution. The VN request life-time conforms to an exponential distribution with an average of 5000 UT (Unit of Time). The simulation horizon is fixed to 5000 UT. The number of VLs per VN request, as well as the number of destination node per VL are set according to a discrete uniform distribution, using the values given in $[1,4]$ and $[1,4]$ respectively. Their bandwidth demand is uniformly distributed between 50 and 100 Mbps. Source and destination node selection is performed on a random basis, the

Algorithm 10: Mutation Scheme

Input : $G(V,E); K_e; \chi^c; mutPB; \gamma_{mut}; W = [w_{e_1}, w_{e_2}, ..., w_{e_{|E|}}]$;
Output: $\chi^d = [\tau_1^d, \tau_2^d, ..., \tau_{|K|}^d]$

1 **begin**
2 $G'(V',E') \longleftarrow$ `Clone`$(G(V,E))$
3 **foreach** $k \in K_e$ **do**
4 **if** $($`Random`$(0,1) < mutPB)$ **then**
5 $W' \longleftarrow$ `Clone`(W)
6 $e_{mut} \longleftarrow$ `OneRandomLink`(τ_k^c)
7 $W'[e_{mut}] \longleftarrow W'[e_{mut}] \times \gamma_{mut}$
8 **foreach** $e \in \chi^c - e_{mut}$ **do**
9 $W'[e] = 0$
10 $\chi^d[k] \longleftarrow$ `ComputeSteinerTree`(G', E', W', s_k, T_k)
11 **else**
12 $\chi^d[k] \longleftarrow \chi^c[k]$

probability that a node is chosen is proportional to its overall bandwidth capacity (i.e. when considering all its incoming/outgoing links).

Similar to VN requests, evolving VLs requests are randomly generated. We assume that evolving VLs requests arrive following a Poisson distribution with an arrival rate $\lambda = 0.06$, i.e. average 6 requests each 100 units of time (UT). Each evolving VLs request consists to a set of VLs from distinct or same VN. The number of VLs is uniformly taken between 1 and 4. These VLs are uniformly chosen among the accepted ones and each of them can be uniformly concerned by a basic or mixed evolving VL scenario. Scenarios are set as follow : *bandwidth expansion* is uniformly distributed between $150Mbps$ and $180Mbps$; finally, for *adding new destination*, one or two nodes are uniformly selected according to the node selection model that is previously defined for new VN request.

4.6.1.3 Algorithms settings

The Resource Allocation Algorithm that handles VN requests is taken from [Tegueu 2016]. It is an ILP-based resource allocation method that jointly accommodates point-to-point and point-to-multipoint VLs, each with an associated bandwidth requirement. It also considers substrate link's bandwidth and node's switching resources. The ILP that we propose in this chapter for solving evolving VLs requests was implemented in C++ with CPLEX-12.06 solver.

For GA, the implementation is in C++ using OpenBeagle [Gagné 2006]. Population size is set to 20 and number of generations at 20. Cross-over probability is 0.9 and mutation probability is 0.05. $\gamma_o^{min}, \gamma_o^{max}, \gamma_{normal}^{max}, \gamma_{comm}^{min}, \gamma_{comm}^{max}$ and γ_{mut} are set to 0.1, 0.3, 1.5, 0.05, 0.15 and 100000 respectively.

We compare the two cases, where the parameters (α and β) of re-embedding

cost C are set to :

— $\alpha = 1$ and $\beta = 0$, hence the resource consumption (the amount of bandwidth and flow table entries) is taken into consideration, neglecting the reconfiguration overhead (service disruption). We note this experiment as **EXP-Minimize-Resource**.

— $\alpha = 0$ and $\beta = 1$, hence only the reconfiguration overhead (service disruption) is taken into consideration, regardless of the resource consumption. We note this experiment as **EXP-Minimize-Reconfiguration**.

4.6.2 Performance metrics

We use several performance metrics for evaluation purposes in our experiments. We measure the acceptance ratio, the reconfiguration overhead (service disruption) and resource consumption.

— Acceptance ratio (ac, in %) : the rate of successful evolving requests out of all the incoming evolving requests during the simulation time. The acceptance ratio translates into application performances and user satisfactory ;

— Reconfiguration overhead (service disruption) : the total number of OpenFlow Modification Message that are generated by the SDN controller to reconfigure forwarding devices, only considering evolving VLs requests. This metric indicates user experience penalty incurred due to service disruption when live updates feature is not supported ;

— Resource consumption : the total amount of bandwidth and flow table entries that are allocated at substrate links and nodes for embedding all running VLs. It measures how substrate resources are consumed and managed.

We plot these performance metrics to show how algorithms actually performs during simulation time.

4.6.3 Evaluation results

Figure 4.4 presents the results of acceptance rate. We see that with ILP, **EXP-Minimize-Resource** presents a higher acceptance rate than **EXP-Minimize-Reconfiguration**, i.e. 91.1%. v.s. 82.2% (computed with ILP). This is in coherence with the resource consumption result, as is presented in Figure 4.6, **EXP-Minimize-Resource** consumes less resources than **EXP-Minimize-Reconfiguration**, hence less refusal of requests.

However, this is in sacrifice of a higher service disruption. As is presented in Figure 4.5, **EXP-Minimize-Resource** brings more reconfiguration overhead (service disruption) than **EXP-Minimize-Reconfiguration** (with ILP).

Our experiments with GA show similar results as ILP, with slightly degraded acceptance rate, and more resource consumption as well as more reconfiguration overhead (service disruption), as is shown in Figure 4.4, Figure 4.6 and Figure 4.5. However, our empirical experiments show that our GA consumes only about 1/3

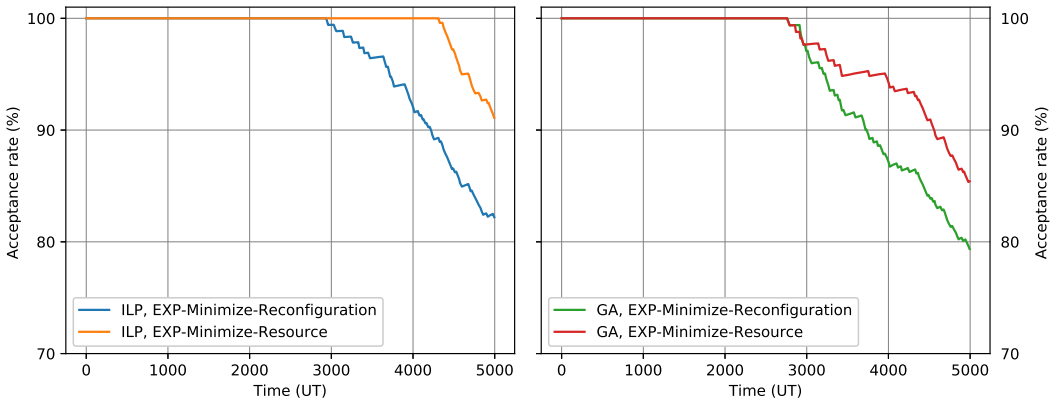

FIGURE 4.4 – Acceptance rate comparison

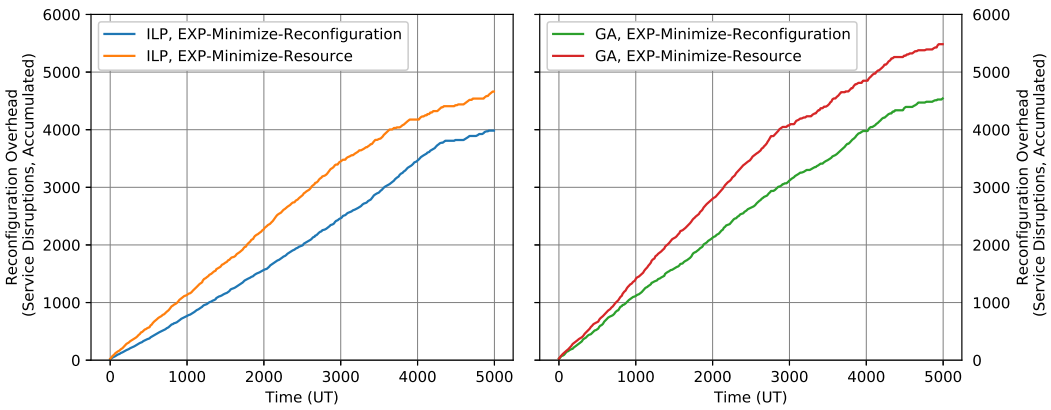

FIGURE 4.5 – Reconfiguration cost accumulated in the time comparison

the computation time of ILP. More experiments should be conducted to explore how GA is impacted by all its parameters.

4.7 Conclusion

In this chapter, we presented our re-embedding schemes for evolving virtual links in SDN networks where the resource requirements of existing virtual link might evolve over time. We consider the trade-off between resource minimization and the avoidance of service disruption by minimizing the reconfiguration overhead. Two algorithms are proposed, one based on an integer linear programming and another based on genetic algorithm. Experimental results show that both solutions work well.

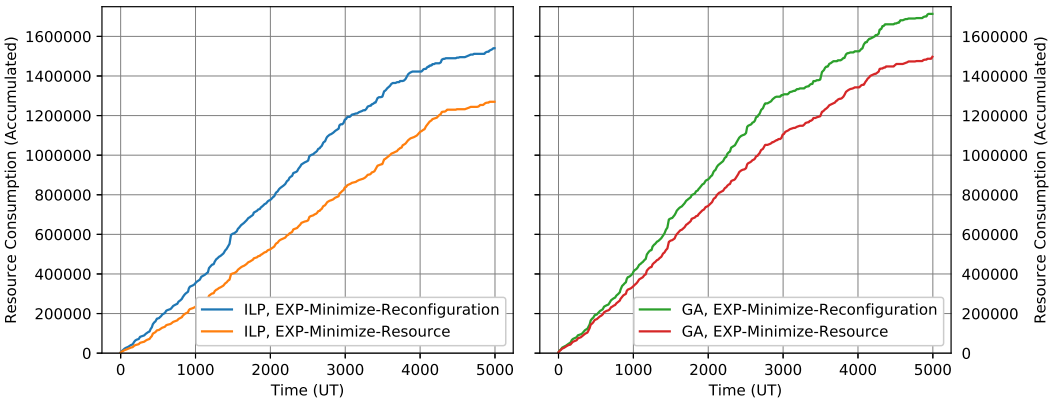

FIGURE 4.6 – Resource consumption accumulated in the time comparison

Chapitre 5

Topology Aggregation and Resource Allocation in Multi-Domain SDN Networks

This chapter proposes an approach based on Steiner trees to efficiently support multipoint communications in an SDN based multi-domain context, where each domain exposes a synthetic and aggregated view of its network. The approach that we propose is based on three pillars : The adoption of a topology aggregation of each domain's network as a Steiner tree, the adoption of global Steiner trees derived from the aggregated topologies as the embedding solution and the use of a shortest path heuristic for the computation of domains' aggregated topologies as well as the global Steiner trees.

Two resource allocation algorithms are proposed. The first addresses the case where network resources are abundant and, if expressed, the bandwidth requirement of the multipoint virtual link to embed can be easily supported by all links. In this scenario, the rationale of our algorithm is to minimize network resource consumption or a predefined network link related cost function. The second algorithm assumes that network resources are limited and for some embedding requests some links may not be able to support the required bandwidth or some nodes may not have enough available entries in their flow tables (switching resources). In this case, when possible, our algorithm computes a global tree that meets the requirements with the objective of minimizing the allocated resources as well as spreading network load (of already accepted multipoint links) at each domain. An experimental study on random and real network topologies assesses the performance of our approach in terms of both accuracy and computational complexity.

5.1 Introduction

Network level multipoint communications are useful for many applications and network operations as it reduces the overhead of maintaining multiple one-to-one communications. With the emergence of SDN with its centralized control, on-the-fly control and the fine-grained flow based forwarding capabilities, a better perspective for supporting multipoint communications with QoS requirements emerges. Some research work have proposed to leverage SDN capabilities for multipoint communications in mobile [Seah 2006] [Li 2004] [Sebastian 2010] and data-centre networks [Cao 2013].

In this chapter, we consider establishing multipoint communications in a multi-domain context, which consists in establishing a tree that spans multiple domains. In this particular context, in practice[Jeon 2006] [Uludag 2005], the network operator that operates a network domain hides the detailed topology information and simply discloses and advertises to other domains an abstracted view of its domain. Clearly, the abstracted view has an impact on the convergence time of the algorithm in charge of computing the global tree as well as the optimality of the obtained tree (in terms of the extra cost with respect to the optimal solution computed on the detailed view of each domain).

Conventionally, full mesh is the most used topology aggregation technique, which consists in aggregating the topology with links connecting all pairs of border nodes. It is used as a general-purpose aggregation technique and finds widespread usage in multi-domain networks. Recent works regarding multi-domain SDN use full mesh as well, as presented in [Egilmez 2012] [Zhao 2015]. However, this conventional topology aggregation technique targets inter-domain one-to-one communication. Applying full mesh as the topology aggregation technique, as we will present in this chapter, is not suitable for multi-domain multipoint communications, the main pitfalls being a non-negligible loss of accuracy and a significant computational complexity. Therefore, an efficient and effective topology aggregation targeted for multipoint communications should be proposed.

The approach proposed in this chapter to efficiently support multipoint communications in a multi-domain context relies on the construction of an approximate global Steiner tree with the following characteristics.

— Our approach adopts a Steiner tree based topology aggregation to abstract each domain's topology. Through experimentation, we show that the Steiner tree abstraction is significantly more effective than the full mesh abstraction, for both randomly generated topologies and real network topologies. Also, the Steiner tree abstraction requires less computational time with less spatial complexity.

  Two algorithms are proposed. In the first algorithm, a generic link cost function is used, and regardless of the multipoint communication's QoS requirements, an approximate minimum Steiner tree is computed from a set of pre-selected nodes (typically, border nodes and eventually some others derived from the traffic engineering policy). This algorithm assumes that the bandwidth requirement of the requested multipoint link can be supported by any link and node, and if the costs of links are set to 1, the computed Steiner tree minimizes network resources (bandwidth and switching) consumption. The second considers the general case where some links or nodes may suffer from the lack of resources to support an incoming multipoint link. A dynamic link cost function that captures the actual resource utilization is used to compute an approximate minimum Steiner tree that maximizes the available resources. A simple traffic engineering strategy is also used to spread the load

of accepted links.

— From the requested multipoint link, an approximate global Steiner tree is derived based on the Steiner tree abstractions of each domain. This latter is the final result of the multipoint link embedding.

— We adopt a shortest path heuristic to compute both the Steiner tree abstraction of each domain and the global Steiner tree. We show through experimentation that this heuristic is robust, flexible and resilient.

This chapter is organized as follows. Section 5.2 reviews previous work from the literature that are related to virtual network embedding in multi-domain networks. Then, Section 5.3 describes the system model and notations used in our formulations. Section 5.4 presents our heuristic for Steiner tree construction in comparison to an ILP-based formulation. Section 5.5 details our proposal of resource allocation with Steiner tree as topology aggregation in multi-domain networks. Section 5.6 presents the performance analysis of the proposed methods. Finally, Section 5.7 concludes the chapter.

5.2 Related works

Virtual network embedding have attracted a lot of attention in recent years. Table 5.1 summarizes existing work that propose solutions on multi-domain network virtual link embedding or routing.

Among those works, only our work considers the case of multi-domain multipoint communications. Also, our work is unique in that Steiner tree is used both in topology aggregation as well as in path calculation. Finally, to our best knowledge, our work is the first to consider switch flow table resources.

5.3 System model and notations

5.3.1 Multi-domain SDN architecture

As shown in Figure 5.1, each domain $d \in D$ has its own SDN controller C_d ("logical") that implements the network control services that are meaningful to the network operator that owns the domain. For the purpose of multi-domain virtual link embedding, it is required that controller C_d implements :

— A topology discovery service that discovers, keeps track and maintains the domain's topology as well as the available link and node resources.

— A "topology abstraction" network function in charge of computing and updating the topology abstraction that is exposed to the master controller or to other domains.

— A virtual link embedding algorithm that applies to its domain.

TABLE 5.1 – Classification of virtual link embedding schemes for multi-domain networks

	Routing & VL Type	SDN substrate	Topology Aggregation	Path Calculation Method	Network Resource	QoS type
[Egilmez 2014]	P2P	Yes	Full mesh based on K-shortest paths	Lagrangian relaxation based solution	Link	Bandwidth Delay Jitter
[Fu 2015]	P2P	Yes	Multi-layer abstraction, pre-calculated shortest paths	Hierarchical routing (Dijkstra's algorithm)	Not specified	Not specified
[Zhu 2015]	P2P	Yes	Virtual node, pre-calculated K-shortest paths	Not specified	Light wavelength	Bandwidth
[Samuel 2013]	P2P	No	Single node abstraction	Least cost path	Financial cost	Bandwidth
[Houidi 2011]	P2P	No	Full mesh and partial mesh	Mixed integer programming	Link Node	Bandwidth
Our work	P2P P2MP	Yes	Steiner tree	Steiner tree	Link Node	Bandwidth

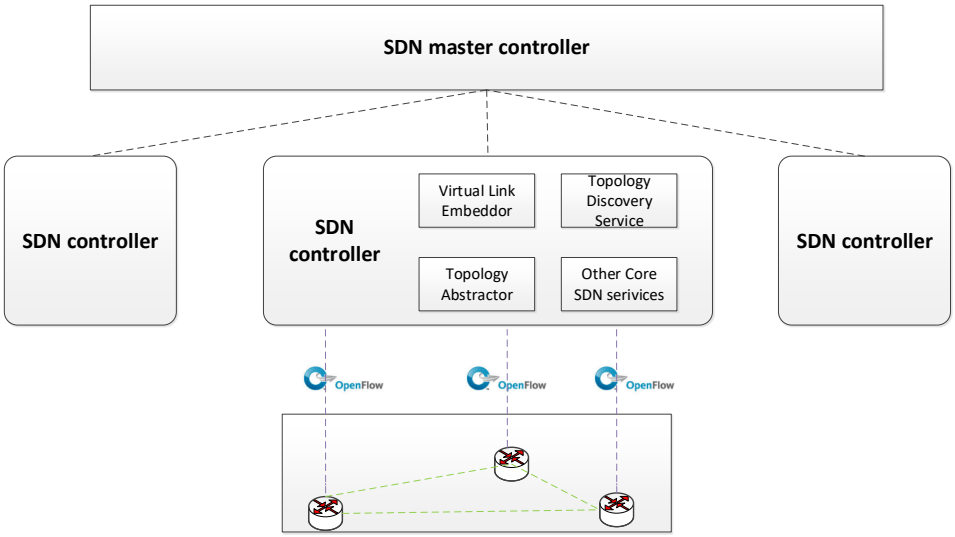

FIGURE 5.1 – Illustration of a two level-hierarchy of controllers for multi-domain networking

Figure 5.1 also highlights a two level-hierarchy of controllers as promoted by TAPI (Transport API) from the ONF and COMS (Common Operation and Management of Network Slicing from the IETF [Liang Geng 2018]). A SDN master controller sits at the top level of the hierarchy and is in charge of coordinating the control of domains' controllers to enable the provision of end-to-end services that span multiple domains. For our embedding technique, the orchestrator controller :

— gathers each domain's abstracted topology $(G_d^a, W_d^a, \forall d \in D)$ and maintains the global abstracted topology G^a, W^a.

— receives multi-domain Virtual Link (VL) requests, from which : (1) it computes the data-paths that support the incoming VL from the global abstracted topology ; from there, (2) it derives the domains that are supporting the requested multi-domain VL as well as the sub-VLs to embed on each traversed domain ; (3) it triggers the embedding of identified sub-VLs on their respective domains ; (4) upon the acknowledgement of the embedding of all sub-VLs, it acknowledges the embedding of the requested multi domain VL.

5.3.2 System model and notations

We consider a multi-domain network composed of multiple network domains interconnected from their border nodes with inter-domain links. The physical network of each domain $d \in D$ of the network is modelled as an undirected graph $G_d^p = (V_d, E_d)$. Each link e within a domain has a link capacity λ_e and residual link capacity λ'_e. Each node v within a domain has a node switch resource capacity ψ_v and residual switch resource capacity ψ'_v. We denote as $Z_d \subset V_d$ the set of border nodes of domain d. The set of all inter-domain links connecting domains via border

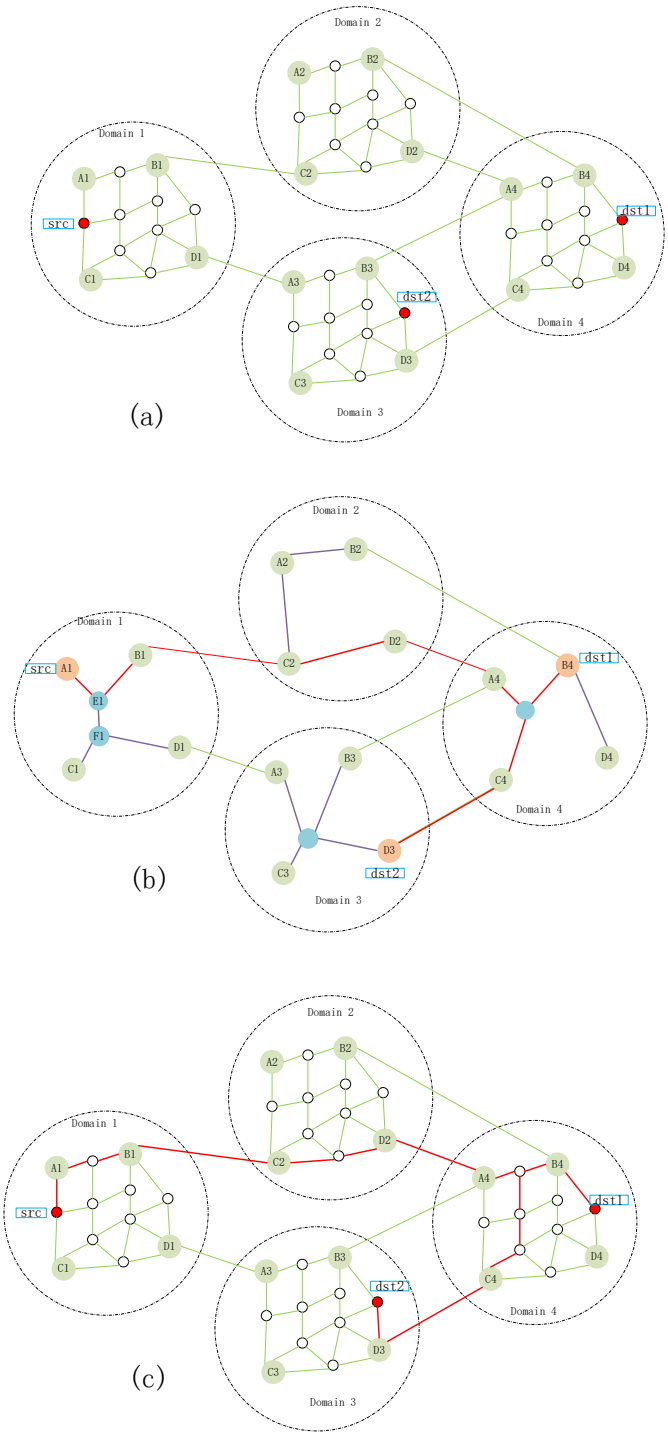

FIGURE 5.2 – Overview of the multi-domain virtual embedding process : (a) Global physical multi-domain network ; (b) Domains' abstracted exposed topologies and the corresponding global abstract view ; (c) Embedding in each domain.

nodes is denoted as E_{inter_domain}. For graph computation purpose, as is usually the case, it's convenient to attach to each link $e \in E_d$ a cost w_e, which forms a set W_d. This link cost function can be defined depending on the application context.

The global physical network is noted as G^p, which is the union of all nodes and links in $d \in D$, as well as inter-domain links, shown as in Figure 5.2 (a). Therefore, $G^p = (\bigcup_{d \in D} V_d, \bigcup_{d \in D} E_d \cup E_{inter_domain})$. Accordingly, the weight set $W = (\cup_{d \in D} W_d) \cup W_{inter_domain}$ refers to the set of all physical link's cost in G^p. This network has a theoretical existence since in practice network domains only reveal an abstracted vision of their network.

The abstracted network disclosed by a network domain d is denoted as $G_d^a = (V_d^a, E_d^a)$, V_d^a includes the set of domain d's border nodes Z_d, but could also include some other nodes if necessary. The links belonging to E_d^a are typically logical, each is established on one or many data paths (often multi-hop). W_d^a refers to the set of domain-level link cost after topology aggregation, clearly, $W_d^a = \cup_{e \in E_d^a} w_e^a$.

We denote as G^a the global undirected graph obtained from the network abstractions exposed by the different domains, i.e. $G^a = \bigcup_{d \in D} G_d^a \cup E_{inter_domain}$. This G^a and is illustrated in 5.2 (b). W^a refers to the set of global link cost after topology aggregation, i.e. $W^a = \bigcup_{d \in D} W_d^a \cup W_{inter_domain}$.

5.3.3 Multi-domain requests model

A multi-domain request r is characterized by :

— A source node s and one or several destination nodes T. Those nodes can be distributed in different domains. This is illustrated in Figure 5.2 (a)-(c).

— A bandwidth requirements of b_r.

5.4 Steiner tree construction

Let's consider a graph $G = (N, A)$, each link $e \in A$ labelled with a cost $w(e)$, with $n = |N|$ and $m = |A|$, a Steiner tree $T = (N_T, A_T)$ is defined as the tree that spans $Z \subset N$ (called Steiner nodes, $q = |Z|$) with minimum total cost $\sum_{e \in A_T} w(e)$.

Finding the optimal Steiner tree is well known to be NP-hard. Here in this chapter, we propose using Integer Linear Programming for the exact solution for small networks, and the shortest path heuristic for approximate solutions for large networks.

5.4.1 Integer Linear Programming for Steiner Tree Construction

Steiner tree problem can be easily modelled as Multi-commodity Flow Based Formulations and hence Integer Linear Programming, as is presented in [Chopra 2001].

For graph $G = (V, E)$, we model each link e with two directed arcs (i, j) and (j, i), the variable f_{ij}^k represents the flow of commodity k from node i to node j. The variable x_e defined for each edge e takes on the value 1 if edge e is in the Steiner

tree and 0 otherwise. Without loss of generality, we assume that node 1 is one of the Steiner nodes. Then, the exact multi-commodity (or, ILP) formulation to solve Steiner tree problem is as follows :

$$\text{Minimize} \sum_{e \in E} w_e x_e$$

Subject to :

$$\sum_{i \in V} f_{ij}^k - \sum_{i \in V} f_{ji}^k = \begin{cases} -1 & \text{if } j = 1 \\ 1 & \text{if } j = k \\ 0 & \text{otherwise for } j \in V - \{1\} \end{cases} \tag{5.1}$$

$$f_{ij}^k \leq x_e \text{ for every edge e = (i, j), commodity } k \tag{5.2}$$

$$f_{ij}^k \geq 0, x_e \text{ integer} \tag{5.3}$$

In the resolved optimal solution, those links with $x_e = 1$ form the Steiner tree.

5.4.2 Shortest Path Heuristic for Steiner Tree Construction

The algorithm is presented in Algorithm 11 and can be reformulated as in Algorithm 12. The algorithm starts with a random node from Z and chooses it as the Steiner tree T. At each iteration, T grows by the shortest path from T to a next element of Z. When all elements of Z are reached, the algorithm ends.

It is to be noted that the Step 2 in Algorithm 11 is essentially the Dijkstra's algorithm, the difference being that the starting point of Dijkstra's algorithm is replaced by the tree T here.

In our implementation, we use the Fibonacci Heap instead of a priority queue or a heap, which has the specificity that the insert and decrease key operations take constant $O(1)$. This brings down the complexity of Step 2 to $O(n \log n + m)$. Since the Step 2 is repeated $q - 1$ times, the complexity of the whole algorithm is $O(qn \log n + qm)$.

The heuristic can also be reformulated as in Algorithm 12. The heuristic approximates the Steiner tree within ratio of 2, which means that the constructed tree is at most 2 times the optimal solution, expressed as :

$$\frac{c(T_{SPH})}{c(T_{exact})} \leq 2(1 - \frac{1}{q})$$

Algorithm 11: Shortest Path Heuristic for Constructing Steiner Tree

```
1 begin
2     Step 1 : Initialization. T = (z_0, ∅), with z_0 ∈ Z being a random element
      of Z.
3     Step 2 : Choice of a shortest path. Choose a node z of Z that is the
      most in proximity to T. Add to T all the edges of the path from z to T.
4       a. Consider the tree T as a super-node and of zero cost. Add its
      neighbours to the set of nodes to be visited.
5       b. Repeat :
6           Choose the unvisited node with minimum cost. Add its unvisited
      neighbours to the set of nodes to be visited (as in Dijkstra's algorithm).
7           Until we reach a node z of Z.
8       c. Add to T all the edges of the path from z to the tree T (the new
      super-node).
9     Step 3 : End of iteration. Repeat Step 2 as long as all elements in Z are
      not connected.
```

Algorithm 12: SPH-Steiner reformulated

```
  Input  : G (an undirected graph), W (associated link cost set),
           Z (Steiner nodes)
  Output: τ (Steiner tree spanning Z on G)
1 begin
2     τ ⟵ Z[0]
3     Z ⟵ Z − Z[0]
4     while Z ≠ ∅ do
5         z ⟵ argmin_{z∈Z} dist(z, τ) (computed over W)
6         τ ⟵ superNode(τ ∪ shortestPath(z, τ))
7         Z ⟵ Z − z
```

5.4.3 Analyses of SPH-Steiner's properties

The performance evaluation of SPH-Steiner is conducted using the networkx graph generator [NetworkX 2018]. Two types of graphs are formulated, namely gnp (also known as an Erdos-Renyi graph or a binomial graph) and grid2d (2-dimensional grid graph) graphs. The conducted experiments consider the case of a single domain. The number of nodes composing the domain is set to 100. For each experiment, the graphs are generated repeatedly randomly but differently at least 1000 times. The number of nodes being fixed, links are generated randomly following the gnp and grid-2d with an associated cost that is varied from 1 to 20. The performance results are averaged on all generated graphs.

5.4.3.1 ILP v.s SPH-Steiner for Steiner tree construction

Implementing the ILP for Steiner tree construction with CPLEX (we note this method as ILP-Steiner), with a 1% tolerance of inaccuracy, we compare the solution quality of Steiner tree computed with SPH-Steiner :

— In more than 58% of cases, SPH-Steiner and ILP-Steiner gives the same result of Steiner tree.

— In 30% of the results, the difference between the costs of the Steiner tree constructed with SPH-Steiner and ILP-Steiner is within 0∼5%.

— In 8% of the results, the difference between the costs of the Steiner tree constructed with SPH-Steiner and ILP-Steiner is within 5∼10%.

— On average, SPH-Steiner brings an increase of 2.7% in the total cost of the constructed Steiner tree, compared to ILP-Steiner.

From our opinion, the results given by SPH-Steiner are quite satisfactory. For the rest of this chapter, we adopt SPH-Steiner algorithm for the construction of Steiner trees, instead of using ILP-Steiner for optimal solutions, which requires much more computational time.

5.4.3.2 Choice of starting node

As the algorithm of SPH-Steiner starts from a random node from Z, the resulted Steiner tree constructed could be seriously impacted by the choice of the initial node. It's important to make sure that the choice of the starting point has limited impact on the performance of the heuristic.

We conduct a set of experiments, by changing the starting point of the SPH-Steiner each time and compare the total cost of the constructed Steiner tree. More precisely, for each graph, we vary the number of Steiner nodes in each domain and also the starting point (there is as many different choices of starting point as there are Steiner nodes), and calculate the resulting coefficient of variance c_v of the total costs, defined as the ratio of the standard deviation σ to the mean μ.

The results are plotted in Figure 5.3. We observe that whatever the number of Steiner nodes in the graph, the coefficient of variance of the Steiner trees constructed from different starting points is always around 0.01% and never exceeds 0.02%, which is quite insignificant. This observation is quite useful as this leads to the effectiveness of starting from a random starting node to construct the Steiner tree with SPH.

5.4.3.3 Constructing Steiner tree from multiple starting nodes

The very limited impact of choosing the starting node with SPH-Steiner leads to the consideration of having multiple starting nodes, instead of one, so that for really large graphs and in a multi-domain context, the calculation could be distributed on multiple entities. Therefore, we investigate the possibility of extending the heuristic

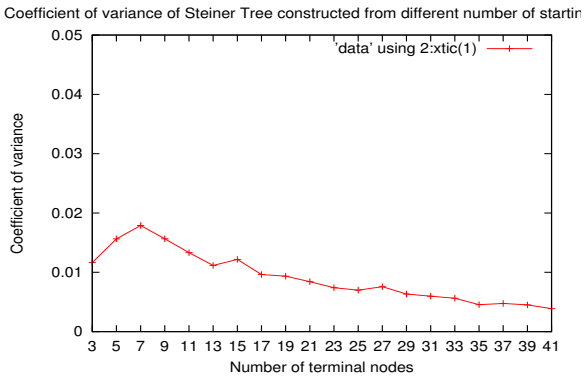

FIGURE 5.3 – Coefficient of variance for total costs calculated with SPH-Steiner from different starting points

of shortest path to start from multiple starting nodes. With simple modifications, we have the algorithm shown in Algorithm 13.

We vary the number of starting nodes $\in [2, 16]$ the single-domain graphs. As is shown in Figure 5.4, for G_d, constructing a Steiner tree from multiple starting points has negligible impact on the total cost of the tree. On average, we observe a variation within the scope of 0.05% in the total cost of the constructed Steiner tree in comparison to the tree constructed from a single starting node. We observe the same result for larger graphs (with number of nodes exceeding 100,000).

Therefore, while constructing a Steiner tree from multiple starting points could be used to bring down greatly the computation time, this is not in the sacrifice of other performance indicators.

5.4.3.4 Addition of new Steiner nodes

Our approach is able to handle quite efficiently the case where new nodes join the multipoint communication. Indeed, the addition of a new Steiner node can be easily handled with the SPH heuristics as explained in Algorithm 14. As the experiments from previous subsections show, the impact of starting from multiple nodes have negligible impact on the overall cost.

5.4.3.5 Recovery of Steiner tree and Re-aggregation

A graph (such as the network) can be dynamic and changing, nodes and links can fail. In that case, it's possible that we need to reconstruct a Steiner tree. On the advent of a link or node failure, two recovery alternatives can be adopted :

– either we can reconstruct a new complete Steiner tree with SPH-Steiner following Algorithm 11.

– or, consider the two broken parts of the Steiner tree as two super-nodes, and use the SPH-Steiner to connect them (Algorithm 15).

In comparison to the former method, the second one requires less computation

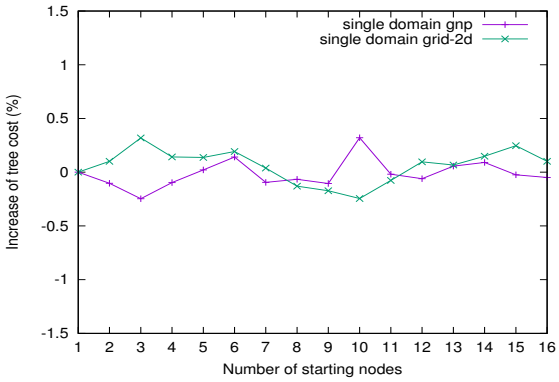

FIGURE 5.4 – SPH-Steiner from multiple staring points : increase of total costs

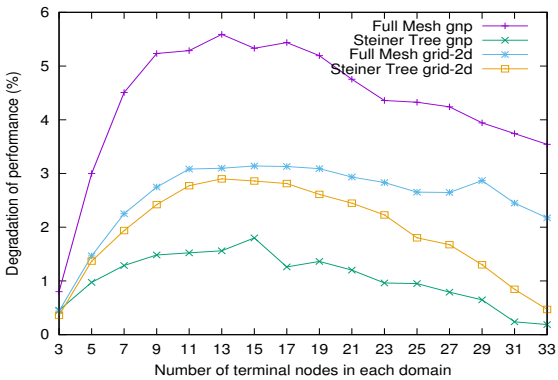

FIGURE 5.5 – Full mesh v.s. Steiner tree as topology aggregation for simulated topologies : performance degradation

Algorithm 13: SPH-Steiner with multiple starting point

```
1  begin
2      Step 1 : Initialization. T = (Z, ∅)
3      Step 2 : Choose a set of s ⊂ Z as the starting nodes (super-nodes). For
       each starting point, make them zero-potential.
4      Step 3 : For each super-node s_i :
5        1. increment its potential.
6        2. If one node z ∈ Z is reached :
7              a. add to s_i z as well as all the nodes in the path from s_i to z.
8              b. make s_i zero potential.
9        3. Else if one node s' ∈ s_j is reached :
10             a. add to s_i s' as well as all the nodes in the path from s_i to s'.
11             b. Union s_i and s_j as one super-node.
12             c. make s_i zero potential.
13     Step 4 : End of iteration. Repeat Step 3 as long as T is not connected.
```

Algorithm 14: Addition of a new Steiner node

```
1  begin
2      Use the SPH-Steiner presented in Algorithm 11 (Step 2) to reach out,
       until the newly to-be-added multicast node is reached.
```

time ($O(Nlg(N))$ v.s. $O(qnlog(n) + qm)$), and have less impact on the recomputed Steiner tree, but it implies an increase in the total cost. Our experiments show that the increase remains under 0.04%. This is why we adopt the second method which is typically used as the recovery mechanism by a domain each time a node failure or link failure occurs.

In fact, as we have seen, Steiner tree constructed with SPH-Steiner is very robust with the mechanism of recovery in the case of node failure or link failure. Each time a node failure or link failure occurs, the Steiner tree is reconstructed within $O(Nlg(N))$ and then the global aggregated graph is also updated, which then again could be used the mechanism of Steiner Tree Recovery in Algorithm 15 to repair the multipoint communication.

5.5 Resource allocation with Steiner tree as topology aggregation in multi-domain SDN

The approach proposed in this chapter promotes the use of a Steiner tree based network domain abstraction. In this section, we will develop our proposal of using the SPH-Steiner of Section 5.4 to construct this topology abstraction as well as to construct the Steiner tree that spans nodes belonging to different domains. We show how resource allocation is performed with our Steiner tree based topology

Algorithm 15: SPH-Steiner Recovery of broken link & node

1 **begin**
2 **Step 1**. When there is a broken link & node, the Steiner tree constructed is broken into two parts. Identify the one of two broken part s_{b1} as a super-node with zero potential and the other part as s_{b2}.
3 **Step 2**. Using the Algorithm 11 (Step 2) to incrementally increase the potential of the super-node s_{b1}, till the moment when the other part s_{b2} is reached.

Algorithm 16: Key principles and main steps of multi-domain virtual link embedding with topology aggregation

Input : $D, r, N^l_{top}, N^s_{top}$, $W = \{W_1, W_2, ..., W_{|D|}\}$,
$Z = \{Z_1, Z_2, ..., Z_{|D|}\}$,
$G^a = \{G^a_1, G^a_2, ..., G^a_{|D|}\} \cup E_{inter_domain}$,
$R^a = \{R^a_1, R^a_2, ..., R^a_{|D|}\}$,

1 **begin**
2 $r', p^{extra} \longleftarrow getDerivedRequest(r, D)$
3 **foreach** $d \in D$ **do**
4 $G^a[d], W^a[d] \longleftarrow SteinerSPH(G_d, W_d, Z_d)$
5 $\chi_r \longleftarrow SteinerSPH(G^a, W^a, r')$
6 $\{\chi_r[1], \chi_r[2], ..., \chi_r[|D|]\} \longleftarrow DecomposeIntoSubTrees(\chi_r)$
7 **foreach** $d \in D$ **do**
8 $embedSubTreeOnDomainNetwork(d, \chi_r[d], p^{extra}[d])$

aggregation to support multipoint communications in a multi-domain context.

5.5.1 Key principles and main steps

The key principles and main steps of our resource allocation with topology aggregation in a multi-domain context for multipoint communications are described as follows (which is also described in Algorithm 16) :

— The computation and maintenance of the aggregated topology of each domain is done with Steiner tree (computed with SPH-Steiner as presented in previous section). Each domain then disseminates the aggregated topology to the master controller (Algorithm 16-Line 3-4).

— Based on the abstracted topology exposed by the domains, the master controller constructs the global view and further computes the global Steiner tree for the multipoint communication (Algorithm 16-Line 5).

— The computed global Steiner tree is then decomposed into sub trees that belongs to different domains (Algorithm 16-Line 6). Each domain then conducts the resource allocation for the sub tree sent from the master controller.

The process of resource allocation within each domain might involve QoS constraints or not. This is illustrated in Figure 5.2 (c).

It is to note that the source and destination nodes of the multipoint request are not necessarily aggregation nodes : they can as well be other nodes from different domains. However, as we stated, domains should preserve their privacy by not exposing details of inner nodes. As part of the aggregated topology advertised by each domain, network/node reachability information is provided. On the other hand, the domain specifies from which aggregation node each inner network or node can be reached. From a multipoint communication request, the SDN master controller derives the multipoint communication request, with aggregation nodes as destinations that will be in fact mapped on the multi-domain network. How nodes get attached to aggregation nodes is at the discretion of the domain and is typically subject to the traffic engineering policy of the domain. In our work, it is based on the idea of nearest neighbour, i.e. the most in proximity in terms of link cost to any aggregation node (easily adapted from Dijkstra's Algorithm). This process of derivation will introduce extra paths p^{extra} from the original request nodes to aggregation nodes. The master controller will afterwards find a VL embedding solution for the derived request instead of for the original request. Each domain then further embed the extra paths, so that the original request is embedded, as is shown in Algorithm 16 Line 7-8.

5.5.2 Embedding without QoS constraints

We consider the case of embedding without QoS constraint, where each link in the substrate is assumed to dispose of abundant bandwidth resources and each node is assumed to dispose of sufficient switching resources. Each link is valued by a simple link cost.

Since the QoS requirements of the multipoint communication requests don't take into consideration the remaining resources of supporting data paths, the embedding operations reduce to the computation of an approximate multi-domain Steiner tree based on the aggregated topologies of domains. The computed tree which is the core output of the embedding is computed with SPH-Steiner. For that, we follow the same principles and main steps as presented in the previous subsection. The only thing that we should specify is the exposed aggregated topology and the resource embedding at the SDN master controller.

5.5.2.1 Exposed aggregated topology

For each domain, the exposed topology aggregation is a Steiner tree. Instead of exposing all details of the Steiner tree, only aggregation nodes (composed of border nodes as well as some internal nodes) as well as virtual nodes with a degree greater than 3 are exposed, along with the logical links connecting them within the Steiner tree, as is illustrated in Figure 5.2 (b). The logical links are valued by a link cost as well, with the link cost being the sum of all substrate link costs that constitute

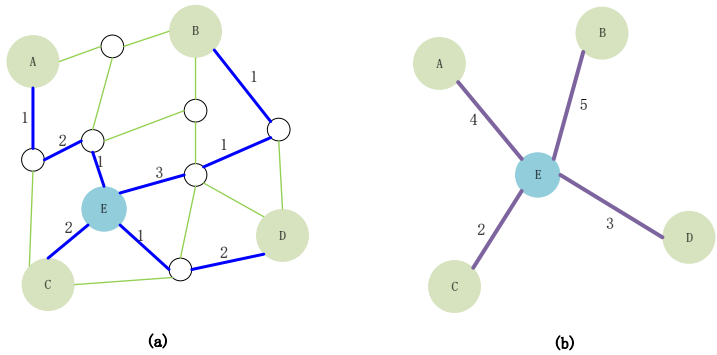

FIGURE 5.6 – Illustration of one domain's topology aggregation without QoS constraints. For the purpose of clarity, only link cost of links on the Steiner tree is shown.

the logical link. Each domain's logical link cost of the aggregated topology is the $W^a[d]$ in Algorithm 16 Line-6. This is illustrated as in Figure 5.6.

5.5.2.2 Resource embedding at the SDN master controller

Using the exposed topology aggregation from all domains as well as inter-domain links, the global tree that spans the derived request is computed over the link costs of the exposed Steiner trees as well as inter-domain links, always using SPH-Steiner. This is illustrated as in Figure 5.7. Note that inter-domain link might have a link cost also. In our case, we assume inter-domain links to be of link cost 0.

5.5.3 Embedding with QoS constraints

The embedding with QoS constraints follows the same general logic as the case with no QoS constraints. The main difficulties lie in :

— The abstracted topology domain must reveal information on the available resources along the exposed Steiner tree in order to allow the SDN master controller to contrast the QoS requirement of multipoint communication to the resources availability on candidate supporting domains.

— Some form of traffic engineering that spreads the network domain's load is used in order to increase the amount of available resources associated to the aggregated topology exposed by the domain. This is achieved by adjusting the cost function of links according to their load. The Steiner tree based abstracted topology will favour less loaded links exposing higher available resources.

Below, we present our dynamic link cost mapping that is enforced at each domain's level, the domain's exposed aggregated topology, and the admissibility control at the SDN master controller.

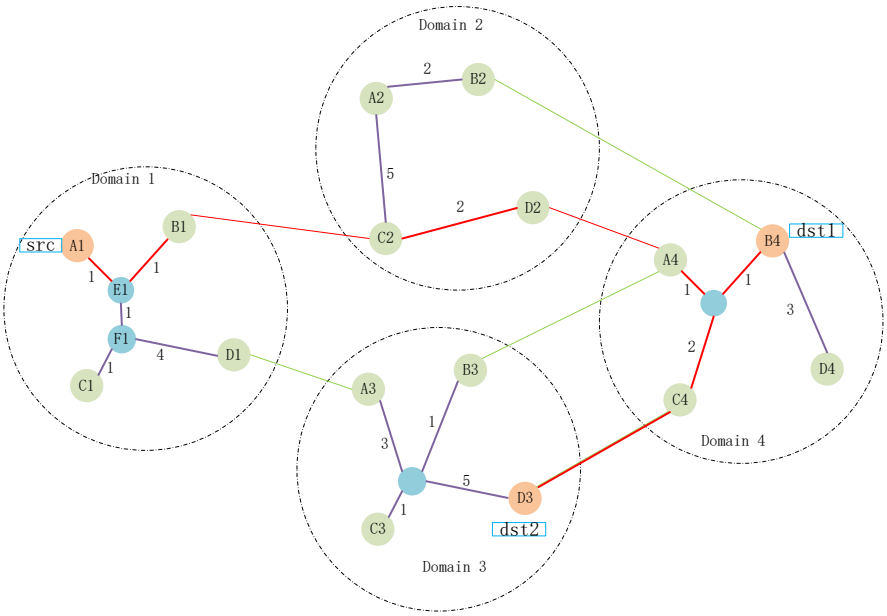

FIGURE 5.7 – Global view and global tree without QoS constraints

5.5.3.1 Dynamic link cost mapping

We consider two resources as QoS constraint : link bandwidth resources and node switching resources. Our consideration is that within each domain d, it is desirable to well spread the link and node load. To this end, our approach is to map multiple resources into one single link metric. This procedure consists hence in giving a link in a domain $e \in G_d$ a pertinent link cost w_e. The reason for this is that with one single link metric, traditional graph algorithms (classical shortest-path-like algorithms) can be easily employed.

Our link cost mapping algorithm functions at domain level d. In fact, the process of giving link cost is managed by the domain controller, and won't be revealed to the (logical) master controller. The main objective is to derive a pertinent topology aggregation (G_d^a, W_d^a) from an appropriate link cost distribution W_d, so that when the topology aggregation is used by the master controller, this goal is achieved : when a portion of the aggregated topology is used by the master controller, there is a decent resource consumption and balancing within the domain. Hereafter we detail our design, i.e. the dynamic mapping of two-dimensional network resources into one-dimensional link cost.

At the beginning, the initial (normal) link cost in each domain is set as 1. If the domain controller judges that the link utilization within a domain is unbalanced (e.g when the utilization rate of most loaded link minus that of the least loaded link is greater than 20%), we should enforce the intra-domain link utilization balancing. To this end, at each new request r :

— link cost of top N_{top}^l most loaded links should be multiplied by α with $\alpha > 1$. Therefore, those most loaded links will be disfavored in forthcoming link

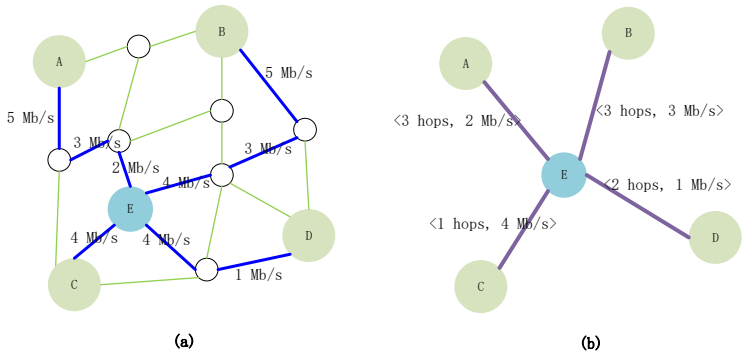

FIGURE 5.8 – Illustration of one domain's topology aggregation with QoS constraints. For the purpose of clarity, only link residual bandwidth of links on the Steiner tree is shown.

embedding solutions. This is shown in Algorithm 17-Line 15.

— If a link e, already among top N^l_{top} most loaded links, is yet used in satisfying the previous request, i.e. $e \in \chi_{prev}$, the multiplier factor becomes β with $\beta > \alpha > 1$. The further negative feedback will help get the embedding solutions out of situations where e continues to be used, leading to a even more unbalanced resource utilization. This is shown in Algorithm 17-Line 13.

— Reverse actions are carried out on top N^l_{top} least loaded links.

Similarly, taking into consideration the switch resource balancing within a domain, if the switch utilization within a domain is unbalanced, we should enforce the intra-domain flow table utilization balancing. In the same manner, at each new request, if a link has one end that is among top N^s_{top} most loaded switches, its link cost should be multiplied by α. If the link is yet used in satisfying the previous request χ_{prev}, the multiplier factor is β. Reverse actions are carried on links that have one end that is among top N^s_{top} least loaded switches.

Other links are always given the normal link cost 1. As using those links or not in the forthcoming topology aggregation and embedding would not have a significant impact on the resource balancing, for those links minimum-hop paths should be constructed to find topology aggregation and embedding solutions.

Finally, the function *updateProhibitiveLinkCost* assigns an infinity link cost to links with not enough bandwidth left for future VLs, and to links with one end node with no flow table entries left. In this way, infeasible solutions are ignored (Algorithm 17-Line 9).

5.5.3.2 Exposed aggregated topology

The topology exposed by a domain d is the approximate Steiner tree computed using the dynamic link cost function $\{W_d, d \in D\}$ of previous subsection. The obtained costs of the logical links that compose this Steiner are not exposed to the SDN master controller as they are only meaningful to the domain. Instead, in this

work, the cost is set to the number of hops that compose the domain's data path on which the virtual link is established. Indeed, the number of hops is straightly related to the consumed resources. The longer (in number of hops) a logical link the more bandwidth and switching resources are consumed to support the link. With such a cost function, the multi-domain embedding algorithm minimises the overall resources consumed at the selected domains.

As part of the aggregate topology exposed by a domain, we also annotate logical links with a minimum residual bandwidth. When embedding a point-to-multipoint virtual link with a bandwidth requirement, this information allows the SDN master controller to consider only the links with sufficient bandwidth resources. The minimum residual bandwidth that is advertised to the SDN master controller can be computed in different ways at the discretion of the domain. In this work, it is computed as the minimum residual bandwidth of the substrate network links that are supporting the logical links.

In summary, each logical link in the aggregated Steiner tree should expose a pair of information : $< hop_count, bw_min >$. This is illustrated in Figure 5.8.

5.5.3.3 Resource embedding at the SDN master controller

From the aggregated topology exposed by each domain, the SDN master controller maintains the whole multi-domain abstracted graph (see Figure 5.9). For the sake of clarity, the annotation in the figure is shown with a pair of real numbers, with the first being the number of hops and the second being the residual bandwidth.

On a multi-domain point-to-multipoint virtual link request arrival,

— the SDN master controller derives the corresponding virtual link, composed exclusively of the aggregation nodes to which end nodes are associated ;

— from the bandwidth requirement, it ignores all the links belonging to the multi-domain abstracted graph that do not meet the bandwidth requirement. Then, it computes the multi-domain Steiner tree with the corresponding aggregation nodes as the Steiner nodes. The Steiner tree identifies the supporting domains and their associated sub-tree ;

— then, it triggers the embedding of each sub-tree from the SDN controller of the corresponding domain as well as the extra paths to reach end nodes from the aggregation nodes.

5.6 Performance evaluation

This section gives the performance evaluations of our proposal. Two evaluations on resource allocation in multi-domain networks with topology aggregation are conducted separately, i.e. without QoS constraints and with QoS constraints. The main objective of the performance evaluation is to show that Steiner tree is a good topology aggregation technique for both cases. In the case of without and

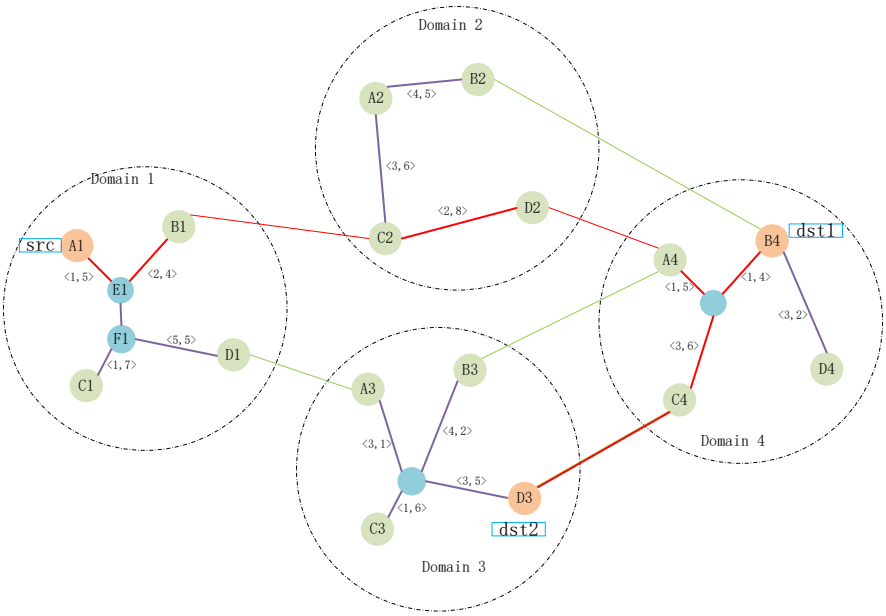

FIGURE 5.9 – Global view and global tree with QoS constraints

with QoS constraints, we compare the results between our proposal and that obtained with full mesh as topology aggregation. In the case of with QoS constraint, we compare the embedding algorithm with an ILP-based solution.

5.6.1 Resource embedding without QoS constraints

One of the most important evaluation metrics for an aggregation technique is the degradation of performance, which measures the difference between the total cost of the Steiner tree constructed with the global view and that constructed with the abstracted view (in %).

In fact, as an aggregated topology hides the details about the intra-domain connectivity, inter-domain links with the aggregated topologies often don't lead to the optimal solution. And this holds true both for multi-domain one-to-one communications and multi-domain multipoint communications. We assess the performance degradation and the computation time of each aggregation technique as a function of the number of target nodes.

5.6.1.1 Simulated topologies

The experiment considers a network with 20 domains, each consisting of 36 nodes. The domains are connected via 80 inter-domain links. For each experiment, the graphs are generated repeatedly at least 1000 times. The number of nodes being fixed, links are generated randomly following the gnp and grid-2d with an associated cost that is varied from 1 to 20. The performance results are averaged on all generated graphs.

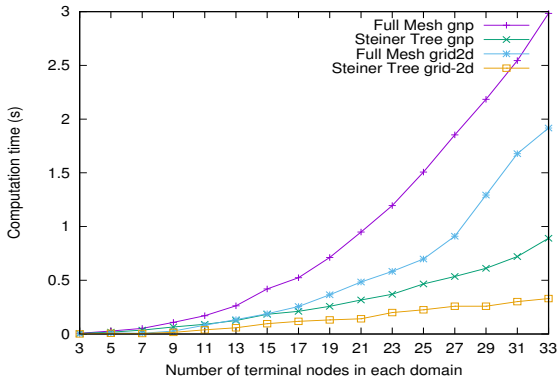

FIGURE 5.10 – Full mesh v.s. Steiner tree as topology aggregation for simulated topologies : computation time

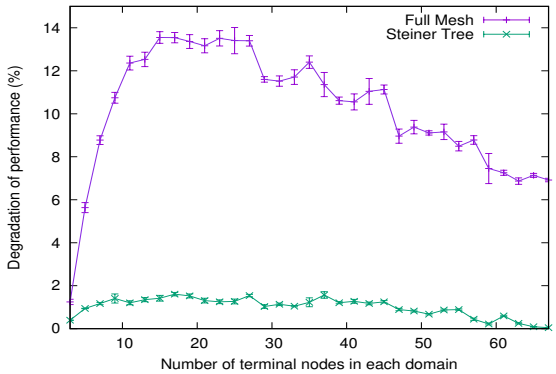

FIGURE 5.11 – Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from ESNET

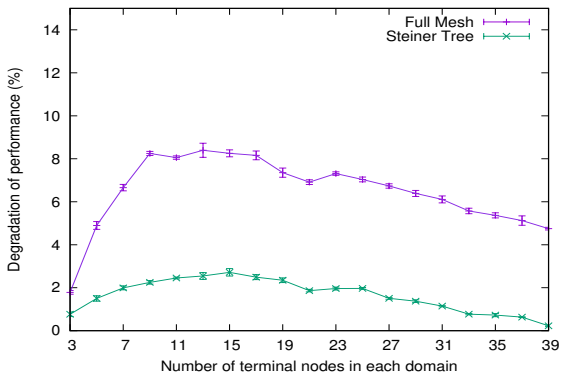

FIGURE 5.12 – Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from GEANT

For each G_d^p, we have the choice to aggregate the topology of each domain in Steiner tree or in full mesh. We vary the number of Steiner nodes in each domain and compare the average degradation of performance (as well as its standard variance) and time complexity in constructing minimum-weight multicast trees, with the two techniques of topology aggregation, as presented in Algorithm 2.

Performance degradation : The result for performance degradation is shown in Figure 5.5, which varies around 1% (gnp) to 3% (grid2d) with the Steiner tree as topology aggregation. When network domains are aggregated in full mesh, the performance degradation increases from 3% (grid-2d) to 5% (gnp).

Time complexity : In fact, the time complexity for building the Steiner tree abstraction is $O(qnlog(n) + qm)$ and the space complexity of the Steiner tree abstraction is $O(q)$. We can see clearly that the time complexity of building multicast tree grows linearly as the number of Steiner nodes of each domain increases.

With full mesh as topology aggregation, the computation complexity increases quadratically as the number of terminal nodes for each domain increases. In fact, the time complexity for building full mesh abstraction is $O(q^3)$ and the space complexity of the full mesh abstraction is $O(q^2)$. With Steiner tree as topology aggregation, however, the increase is in a quasi-linear manner. The result is shown in Figure 5.10.

5.6.1.2 Real topologies

We also measure the performance based on real topologies, with every G_d^p being ESNET [ESNET 2018] or GEANT [GEANT 2018], INERNET2 [Internet2 2018]. As we can see in Figure 5.11 and Figure 5.12, for both topologies, the Steiner tree always has better results than full mesh based topology aggregation. Our experiments with the topology of INTERNET2 [Internet2 2018] also confirm this result.

As a conclusion, it's safe to say that Steiner tree is a better candidate than full mesh based topology aggregation, with the aim of constructing an approximate minimum-cost multi-domain multipoint communication trees.

As a complementary result, from our experiments, the Steiner tree based aggregation is always better than full mesh when we target multi-domain broadcast. However, when we target point-to-point communications, full mesh is better than the Steiner tree.

5.6.2 Resource embedding with QoS constraints

The objective of this performance analysis is to show that our methods, compared to those with less sophisticated link metrics, achieves those goals : (1) satisfactory bandwidth allocation within domains, with a decent bandwidth allocation balancing ; (2) satisfactory switch resource allocation within domains, with a decent switch resource allocation balancing ; (3) those two factors contribute to a decent admissibility of VL request. Below, we describe our simulation model, the main performance metrics and some of the obtained results.

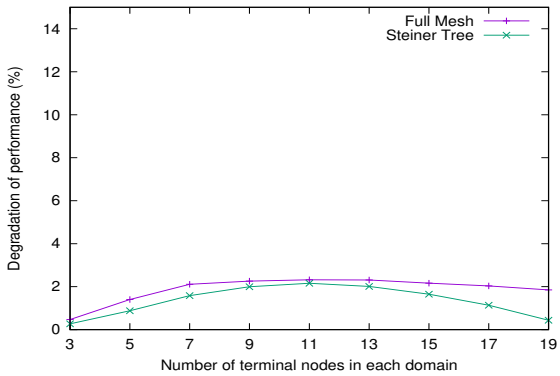

FIGURE 5.13 – Full mesh v.s. Steiner tree for topology aggregation, with each domain derived from INTERNET2

5.6.2.1 Network Model

For space reasons, one single multi-domain network instance is considered in the presented results, which is composed of 9 domains. Each domain takes a GEANT topology, which is composed of 41 nodes and 59 links. For each domain, the 5 nodes with a degree greater than six are chosen as aggregation nodes. Those aggregation nodes are connected with each other by 120 inter-domain links. Within the domain, the initial link capacity of each link is set to 50 units of bandwidth (UB). Unless specified, the initial switch resource volume of each node is set to 100. Aggregation nodes are assumed to have an infinity switch resource capacity. In fact, border nodes are often proposed to be software-based and disposing of memory-based large flow table entries, as is proposed in [Iyer 2013], [Katta 2014], [Huang 2018], [Ghorbani 2017], [Jin 2013], [Casado 2012] etc. Similarly, inter-domain links are assumed to have an infinity link capacity.

5.6.2.2 Load Model

Virtual links are supposed to be point-to-point and point-to-multipoint, each of which has a number of destinations randomly chosen between 1 and 64. The bandwidth requirement of each virtual link is also chosen randomly from 1 to 2 UB. Source and destination selection is performed on a random basis distributed in different domains. The request arrivals follow a Poisson process with an arrival rate r of 0.1, 0.2, 0.3, 0.4, i.e. in average 10, 20, 30 or 40 requests each 100 units of time (UT). The request life-time conforms to an exponential distribution with an average of 1000 UT.

5.6.2.3 Comparisons with global ideal solution based on ILP

Our comparison concerns the ideal case where the topology of each domain is not aggregated. Instead, every detail of every domain is revealed to the master controller and a global solution χ^{global} is then computed. That is, the solution is

computed on the G^p, instead of G^a. Note that this is only the ideal case, in reality this should not happen. We take this as a comparison to measure to which extent our proposal degrades in performance result compared to the ideal case.

To this end, we use our previous ILP formulation presented in [Capelle 2015] to calculate the solution, except that (1) we no longer take the maximum link utilization, but maximum link utilization minus minimum link utilization, as a constraint to enforce link utilization balance among all network links. (2) similarly, we no longer take the maximum flow table utilization, but maximum flow table utilization minus minimum flow table utilization, as a constraint to enforce flow table utilization balance among all switches. (3) the delay and path splitting is not taken into account. This is a simple adaptation, hence we don't detail it in this chapter.

In the following experiments, we note this method as ILP-global.

5.6.2.4 Simulation Settings

The Integer Linear model was implemented in Python with CPLEX 12.63 solver. The experiments were carried out on a virtual machine with 25 vCPU and 16GB of RAM and running Ubuntu 14.04. A gap of less than 1% to the optimal solution is considered satisfactory. The rest of all algorithms were implemented in Python.

N^l_{top} is set to 7 and N^s_{top} is set to 5 throughout all evaluation experiments. α is set to 1.1 and β is set to 1.5. The simulation horizon is fixed to 10000 UT (this time period is sufficient to have our methods in the stationary regime).

5.6.2.5 Performance Metrics

The following performance metrics are computed during simulation for performance analysis purposes :

- Acceptance rate (ac, in %) : the percentage of successful virtual links requests out of all the requests that arrived during the simulation time.
- Link utilization (lu, in %). bandwidth allocated at the link divided by its initial link capacity.
- Switch flow table utilization(su, in %) : flow table utilization at the nodes divided by the initial flow table size.
- Computation time (in seconds) : the average computation time for one request.

5.6.2.6 Performance Results

Resource consumption and balancing within a domain :

Let's focus on one domain among the 9 domains, to understand its inner behaviour in resource consumption and balancing.

Figure 5.14 shows the link utilization of 20 links in this domain (for the sake of clarity, only 1/3 of all links, sampled based on an uniform basis, are plotted ; however, this doesn't affect our observations and conclusions). We observe that

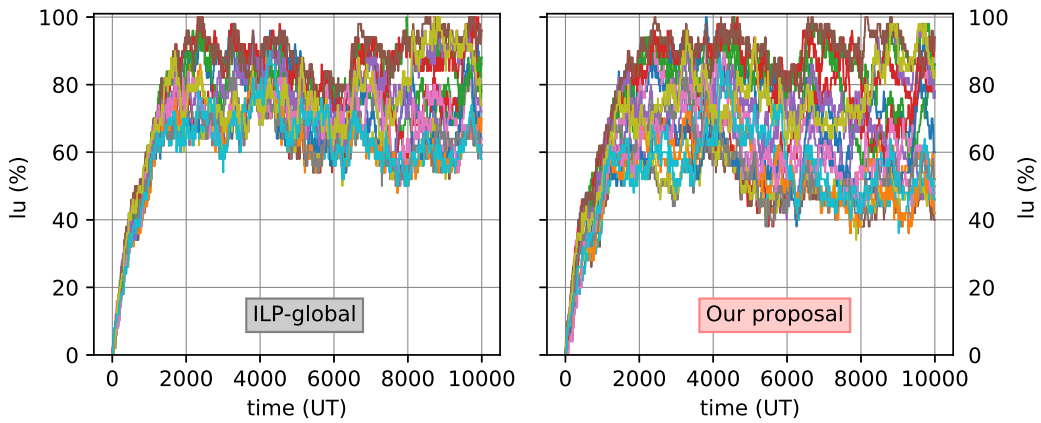

FIGURE 5.14 – BW balancing comparison of ILP-Global v.s. Steiner-Aggregation

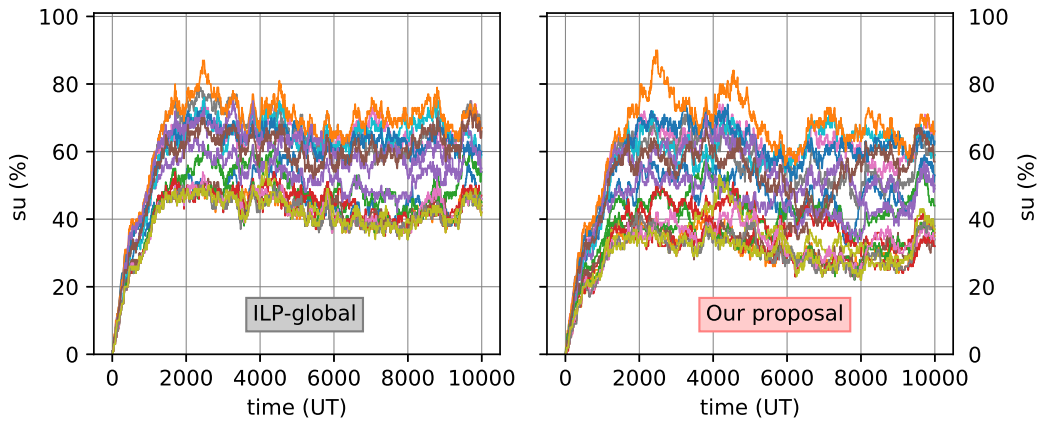

FIGURE 5.15 – SW balancing comparison of ILP-Global v.s. Steiner-Aggregation

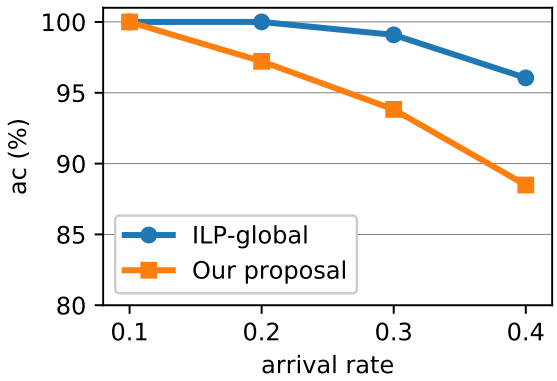

FIGURE 5.16 – ac normal comparison of ILP-Global v.s. Steiner-Aggregation

Steiner-Aggregation gives decent link utilization balancing among links, although slightly falling behind ILP-global in that regard.

Figure 5.15 shows the flow table utilization of 20 switches in this domain (again, for the sake of clarity, only 1/2 of all switches are plotted). We observe that Steiner-Aggregation achieves a decent flow table utilization balancing among switches, though falling behind ILP-global.

Acceptance rate :

Figure 5.16 depicts the acceptance rate using different methods, with different arrival rate of requests. In this setting, the limited link capacity is the main bottleneck. We observe that ILP-global gives slightly better acceptance rate than Steiner-Aggregation. This is in coherence with our observation of link utilization balancing within one domain and among domains.

Computation time : With our proposed method based on topology aggregation and SPH-Steiner (i.e. Steiner-Aggregation), the computation time of satisfying a request is in the order of tens of milliseconds. With ILP-global, however, it takes 40 to 100 seconds. Therefore, it's simply no comparison. Our proposal has therefore a great advantage in computation time.

5.7 Conclusions

We have proposed in this chapter a Steiner tree based approach to efficiently support multipoint communications in a multi-domain context. Our approach advocates the use of a Steiner tree topology aggregation of domains' networks in replacement of the commonly used full mesh topology aggregation. We also propose to use the SPH-Steiner heuristic for building the topology aggregations as well as the global Steiner tree that supports the multipoint communication. Through experimentations, we have shown that our algorithms are efficient in terms of computation time and induced total cost and exhibit some interesting properties that make them scalable and capable of coping with the dynamicity of both the network

and the multipoint communication. Our methods can deal with situations both with and without QoS constraints well.

Algorithm 17: Dynamic Mapping Network Resources To Link Cost

Input : $G_d = (V_d, E_d); N^l_{top}; N^s_{top}; \chi_{prev}; W_d = [w_{e_1}, w_{e_2}, ..., w_{e_{|E_d|}}]$
Output: W_d (modified in place)

```
1  begin
2      L_most, L_least, S_most, S_least ⟵ ∅
3      if link_utilization_not_balanced then
4          L_most ⟵ mostUsedLinks(E_d, N^l_top)
5          L_least ⟵ leastUsedLinks(E_d, N^l_top)
6      if flow_table_utilization_not_balanced then
7          S_most ⟵ mostUsedSwitches(V_d, N^s_top)
8          S_least ⟵ leastUsedSwitches(V_d, N^s_top)
9      updateProhibitiveLinkCost()
10     foreach e ∈ E_d do
11         is_normal_link ⟵ true
12         if e ∈ L_most or e ∈ S_most then
13             if e ∈ χ_prev then
14                 W_d[e] ⟵ W_d[e] × β
15             else
16                 W_d[e] ⟵ W_d[e] × α
17             is_normal_link ⟵ false
18         if e ∉ L_least or e ∉ S_least then
19             if e ∈ χ_prev then
20                 W_d[e] ⟵ W_d[e] ÷ β
21             else
22                 W_d[e] ⟵ W_d[e] ÷ α
23             is_normal_link ⟵ false
24         if is_normal_link = true then
25             W_d[e] ⟵ 1
```

Chapitre 6

Conclusions

6.1 Conclusions

In this work, we consider the key issues related to the embedding of point-to-multipoint end-to-end virtual links, potentially dynamic, and with QoS requirements on a multi-domain SDN based network, with some of the domains being wireless multi-hop networks. Contributions are made in correspondence to the identified key issues : topology aggregation in a multi-domain context, the topology discovery of wireless domains, the resource embedding of VLs, and the re-embedding of VLs.

For the multi-domain aspect of the problem, we have identified the need for a new topology aggregation technique that is more suited for point-to-multipoint communications. To tackle this problem, a Steiner tree based solution is proposed and a shortest path heuristic is used, demonstrating satisfactory performances of aggregation, which is efficient in terms of computation time and induced total cost and exhibit some interesting properties that make them scalable and capable of coping with the dynamicity of both the network and the multipoint communication. We showed that we can have satisfactory resource allocation results in a multi-domain situation.

At the level of each domain, we have identified that there is a gap to fill in the topology discovery service when applying the SDN paradigm to wireless multi-hop networks. Based on the legacy OFDP-based discovery service used in wired SDN networks, we have enriched the information to expose by the controller for networking applications, by including new node and link attributes specific to multi-hop wireless communications. We have specified the procedures and mechanisms that establish and maintain this representation with reduced overhead. We have also adjusted the discovery service so that multipoint links are discovered, which is specific to wireless communications. A proof-of-concept implementation on a SDN-enabled multi-channel multi-interface wireless multi-hop network testbed showed the feasibility of our proposal.

Also at the domain level, the problem of resource allocation is investigated. The main challenge resides in how to minimize bandwidth resource consumption, how to model and minimize wireless interference, how to benefit from the multicast advantage of wireless communications for point-to-multipoint communications, how to balance channel utilizations among different channels, and how to minimize and balance switch table resource consumption. Two methods, one in ILP and another in GA, are proposed to solve the problem. Through extensive experiments, we showed that both methods work well and we were able to have satisfactory embedding

results that achieved those stated goals.

Finally, we consider that virtual link requests might evolve over time. The difficulty lays in the trade-off between resource consumption minimization and service disruption. We proposed an ILP and a GA-based solution to tackle the problem and we showed that our proposals work well.

6.2 Thesis technological scope

For each problem defined in the PhD work, we propose a specific solution with different techniques. Table 6.3.3 shows the technological scope of the PhD work from an implementation point of view. We can see that our implementations cover mainly graph algorithms, optimization algorithms and open-source networking projects.

6.3 Perspectives

In this work, we tried to propose an integral solution for embedding point-to-multipoint virtual links across SDN domains. However, we see that there is room for further improvements. Here are our perspectives for those improvements to be made :

6.3.1 Topology discovery in wireless SDN

In the work [Pakzad 2016], an efficient topology discovery mechanism was proposed which reduces the topology discovery overhead by minimizing *Packet_Out* messages generated from the controller. In their proposal, a single LLDP packet is sent to each of the OF switches rather than the *de-facto* standard of sending each LLDP packet to each of the ports of the OpenFlow switch. Then, the switch broadcasts the LLDP packet to all its active ports which allows the discovery of links between the switches. It would be an interesting path to follow to couple the OFDPv2 [Pakzad 2016] and our proposal to have an even more efficient topology discovery service for software-defined wireless multi-hop networks.

Moreover, there is always room for optimization for a better efficiency. For example, when the wireless network is dense, the procedure of topology discovery might be offloaded to a part of the wireless nodes, instead of all of them. It might be interesting to use classic Minimum Dominating Set to reduce the number of nodes that directly report to the SDN controller about the underlying topology status.

Finally, there is the security aspect of LLDP/OFDP to address, as is investigated in [Azzouni 2017b] and [Khan 2017].

6.3.2 Resource allocation for virtual link embedding

Regarding the problem of resource allocation for virtual link embedding, in addition to bandwidth and switch resources, other network resources could be taken into consideration. For example, wireless nodes are often constrained by their CPU,

RAM, battery level etc. It would be interesting to take those node resources into consideration as well. Other QoS constraints on wireless links, such as delay, packet loss, jitter rate etc. might as well be incorporated into our scheme.

We note that our resource allocation problem is essentially a combinatorial optimization problem on graphs. Recent years machine learning techniques (especially deep learning and deep reinforcement learning) begin to find more and more applications in combinatorial optimization over graphs, to solve classical graph problems such as Minimum Vertex Cover, Maximum Cut and Travelling Salesman problems [Khalil 2017] [Vinyals 2015] [Bello 2016]. It would be a good direction to follow to enhance our optimization problem with such techniques. Indeed, for QoS routing or traffic engineering in SDN, pioneering works such as [Zuo 2019] [Azzouni 2017a] [Valadarsky 2017] propose to solve in deep reinforcement learning. It is therefore a good path to follow and to explore.

Regarding the evolving re-embedding scheme, more than request dynamicity, it can be interesting to consider also the network substrate dynamicity, such as link or node failure. In such cases, apart from an efficient re-embedding scheme, failover with SDN is a good direction to consider [Pignolet 2017], so as to provide with more fault-tolerance, robustness and resilience, especially in a wireless environment.

6.3.3 Topology aggregation in multi-domain networks

Concerning our multi-domain topology aggregation, the future direction is to design a peering procedure between heterogeneous network domains, e.g. wired and wireless ones, so that their distinct QoS parameters could be negotiated and adjusted. The work in [Das 2017] points out a good direction to follow.

Also, it is important to define an interface between the SDN controllers and the coordinator. East-west interface should be defined so that our topology aggregation mechanism could be applied successfully.

Table 6.1 – Thesis Technological Scope

Problem	Validation Method	Employed Open-source Projects or Commercial Software	Solving Tehchinque Field	Solving Method
Topology Aggregation	Simulation	NetworkX [Hagberg 2008]	Heuristics, Graph Algorithms	Shortest Path Based Heuristic
Topology Discovery	Real Platform	Ryu [Ryu 2017], Open vSwitch [Pfaff 2015]	SDN controller, virtual switch, Linux Kernel programming	Modification of source code to Ryu and Open vSwitch, Verification on a real platform of OpenWRT [Fainelli 2008] nodes
Resource Allocation in one domain, in dynamic environment, in multi-domain context	Simulation	Deap [De Rainville 2012], Open Beagle [Gagné 2006], NetworkX, C++ boost graph library [Siek 2001], IBM CPLEX [CPLEX 2018]	Combinatory optimization, Meta-heuristics, Graph Algorithms	Integer Linear Programming, Genetic Algorithm

Bibliographie

[3GPP 2004] 3GPP. *Introduction of the Multimedia Broadcast Multicast Service (MBMS) in the Radio Access Network (RAN); Stage 2 (3GPP TS 25.346 version 6.0.0 Release 6).* `https://www.etsi.org/deliver/etsi_ts/125300_125399/125346/06.00.00_60/ts_125346v060000p.pdf`, 2004. (Cited in page 1.)

[Abdelwahab 2016] S. Abdelwahab, B. Hamdaoui, M. Guizani and T. Znati. *Efficient Virtual Network Embedding With Backtrack Avoidance for Dynamic Wireless Networks.* IEEE Transactions on Wireless Communications, vol. 15, no. 4, pages 2669–2683, April 2016. (Cited in page 30.)

[Adya 2004] A. Adya, P. Bahl, J. Padhye, A. Wolman and Lidong Zhou. *A multi-radio unification protocol for IEEE 802.11 wireless networks.* In First International Conference on Broadband Networks, pages 344–354, Oct 2004. (Cited in page 13.)

[Alotaibi 2008] E. Alotaibi and S. Roy. *A Location-Aware Routing Metric (ALARM) for Multi-Hop, Multi-Channel Wireless Mesh Networks.* In 2008 IEEE Wireless Communications and Networking Conference, pages 2081–2086, March 2008. (Cited in page 13.)

[Amusa 2011] E. Amusa, O. Adjei, J. Zhang, A. Mansour and A. Capone. *An efficient RSSI-aware metric for wireless mesh networks.* In 2011 International Symposium of Modeling and Optimization of Mobile, Ad Hoc, and Wireless Networks, pages 314–320, May 2011. (Cited in page 13.)

[Azzouni 2017a] Abdelhadi Azzouni, Raouf Boutaba and Guy Pujolle. *NeuRoute : Predictive dynamic routing for software-defined networks.* arXiv preprint arXiv :1709.06002, 2017. (Cited in page 105.)

[Azzouni 2017b] Abdelhadi Azzouni, Raouf Boutaba, Nguyen Thi Mai Trang and Guy Pujolle. *sOFTDP : Secure and Efficient Topology Discovery Protocol for SDN.* arXiv preprint arXiv :1705.04527, 2017. (Cited in page 104.)

[Bello 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi and Samy Bengio. *Neural combinatorial optimization with reinforcement learning.* arXiv preprint arXiv :1611.09940, 2016. (Cited in page 105.)

[Cai 2010] Z. Cai, F. Liu, N. Xiao, Q. Liu and Z. Wang. *Virtual Network Embedding for Evolving Networks.* In 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, pages 1–5, Dec 2010. (Cited in page 62.)

[Cao 2013] J. Cao, C. Guo, G. Lu, Y. Xiong, Y. Zheng, Y. Zhang, Y. Zhu, C. Chen and Y. Tian. *Datacast : A Scalable and Efficient Reliable Group Data Delivery Service for Data Centers.* IEEE Journal on Selected Areas in Communications, vol. 31, no. 12, pages 2632–2645, December 2013. (Cited in page 75.)

[Capelle 2015] M. Capelle, S. Abdellatif, M. Huguet and P. Berthou. *Online virtual links resource allocation in Software-Defined Networks.* In 2015 IFIP Networking Conference (IFIP Networking), pages 1–9, May 2015. (Cited in pages 33 and 98.)

[Casado 2012] Martin Casado, Teemu Koponen, Scott Shenker and Amin Tootoonchian. *Fabric : a retrospective on evolving SDN.* In Proceedings of the first workshop on Hot topics in software defined networks, pages 85–90. ACM, 2012. (Cited in page 97.)

[Cazals 2008] F. Cazals and C. Karande. *A note on the problem of reporting maximal cliques.* Theoretical Computer Science, vol. 407, no. 1, pages 564 – 568, 2008. (Cited in page 32.)

[Centrality 2018] Centrality. *Centrality.* `en.wikipedia.org/wiki/Centrality`, 2018. [Online]. (Cited in page 21.)

[Chao 2003] Chih-Min Chao, Jang-Ping Sheu and Cheng-Ta Hu. *Energy-conserving grid routing protocol in mobile ad hoc networks.* In 2003 International Conference on Parallel Processing, 2003. Proceedings., pages 265–272, Oct 2003. (Cited in page 13.)

[Chiang 2007] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill and D. Julian. *Power Control By Geometric Programming.* IEEE Transactions on Wireless Communications, vol. 6, no. 7, pages 2640–2651, July 2007. (Cited in page 14.)

[Chopra 2001] Sunil Chopra and Chih-Yang Tsai. Polyhedral approaches for the steiner tree problem on graphs, pages 175–201. Springer US, Boston, MA, 2001. (Cited in page 81.)

[CPLEX 2018] CPLEX. *CPLEX Optimizer.* `https://www.ibm.com/analytics/cplex-optimizer`, 2018. (Cited in page 106.)

[Das 2017] D. Das, J. Bapat and D. Das. *A Dynamic QoS Negotiation Mechanism Between Wired and Wireless SDN Domains.* IEEE Transactions on Network and Service Management, vol. 14, no. 4, pages 1076–1085, Dec 2017. (Cited in page 105.)

[De Couto 2003] Douglas S. J. De Couto, Daniel Aguayo, John Bicket and Robert Morris. *A High-throughput Path Metric for Multi-hop Wireless Routing.* In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom '03, pages 134–146, New York, NY, USA, 2003. ACM. (Cited in page 14.)

[De Oliveira 2015] Bruno Trevizan De Oliveira, Lucas Batista Gabriel and Cintia Borges Margi. *TinySDN : Enabling multiple controllers for software-defined wireless sensor networks.* IEEE Latin America Transactions, vol. 13, no. 11, pages 3690–3696, 2015. (Cited in page 9.)

[De Rainville 2012] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau and Christian Gagné. *DEAP : A Python*

Framework for Evolutionary Algorithms. In Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '12, pages 85–92, New York, NY, USA, 2012. ACM. (Cited in pages 47 and 106.)

[Dely 2011] Peter Dely, Andreas Kassler and Nico Bayer. *Openflow for wireless mesh networks.* In Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on, pages 1–6. IEEE, 2011. (Cited in page 9.)

[Draves 2004] Richard Draves, Jitendra Padhye and Brian Zill. *Routing in Multi-radio, Multi-hop Wireless Mesh Networks.* In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MobiCom '04, pages 114–128, New York, NY, USA, 2004. ACM. (Cited in pages 13 and 14.)

[Dube 1997] R. Dube, C. D. Rais, Kuang-Yeh Wang and S. K. Tripathi. *Signal stability-based adaptive routing (SSA) for ad hoc mobile networks.* IEEE Personal Communications, vol. 4, no. 1, pages 36–45, Feb 1997. (Cited in page 15.)

[Egilmez 2012] H. E. Egilmez, S. Civanlar and A. M. Tekalp. *A distributed QoS routing architecture for scalable video streaming over multi-domain OpenFlow networks.* In 2012 19th IEEE International Conference on Image Processing, pages 2237–2240, Sept 2012. (Cited in page 76.)

[Egilmez 2014] H. E. Egilmez and A. M. Tekalp. *Distributed QoS Architectures for Multimedia Streaming Over Software Defined Networks.* IEEE Transactions on Multimedia, vol. 16, no. 6, pages 1597–1609, Oct 2014. (Cited in page 78.)

[ESNET 2018] ESNET. *ESNET.* `www.es.net/engineering-services/the-network/network-maps/historical-network-maps`, 2018. (Cited in page 96.)

[Fainelli 2008] Florian Fainelli. *The OpenWrt embedded development framework.* In Proceedings of the Free and Open Source Software Developers European Meeting, 2008. (Cited in page 106.)

[Fonseca 2007] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson and Philip Levis. *Four-Bit Wireless Link Estimation.* In HotNets, 2007. (Cited in page 9.)

[Fu 2015] Y. Fu, J. Bi, Z. Chen, K. Gao, B. Zhang, G. Chen and J. Wu. *A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale Software-Defined Networks.* IEEE Transactions on Network and Service Management, vol. 12, no. 2, pages 117–131, June 2015. (Cited in page 78.)

[Gagné 2006] Christian Gagné and Marc Parizeau. *Genericity in Evolutionary Computation Software Tools : Principles and Case Study.* International Journal on Artificial Intelligence Tools, vol. 15, no. 2, pages 173–194, 2006. (Cited in pages 71 and 106.)

[Galluccio 2015] Laura Galluccio, Sebastiano Milardo, Giacomo Morabito and Sergio Palazzo. *SDN-WISE : Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks.* In Computer Communications (INFOCOM), 2015 IEEE Conference on, pages 513–521. IEEE, 2015. (Cited in page 9.)

[Gante 2014] A. De Gante, M. Aslan and A. Matrawy. *Smart wireless sensor network management based on software-defined networking.* In 2014 27th Biennial Symposium on Communications (QBSC), pages 71–75, June 2014. (Cited in page 28.)

[GEANT 2018] GEANT. *GEANT.* `www.geant.org/Networks/Pan-European_network/Pages/GEANT_topology_map.aspx`, 2018. (Cited in page 96.)

[Geng 2018] Liang Geng, Li Qiang, Jose Ordonez Lucena, Pablo Ameigeiras, Diego Lopez and Luis M. Contreras. *COMS Architecture.* Internet-Draft draft-geng-coms-architecture-02, Internet Engineering Task Force, septembre 2018. Work in Progress. (Cited in page 2.)

[Ghorbani 2017] Soudeh Ghorbani, Zibin Yang, P Godfrey, Yashar Ganjali and Amin Firoozshahian. *DRILL : Micro load balancing for low-latency data center networks.* In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pages 225–238. ACM, 2017. (Cited in page 97.)

[Gomez-Barquero 2018] D. Gomez-Barquero, D. Navratil, S. Appleby and M. Stagg. *Point-to-Multipoint Communication Enablers for the Fifth Generation of Wireless Systems.* IEEE Communications Standards Magazine, vol. 2, no. 1, pages 53–59, MARCH 2018. (Cited in page 1.)

[Gupta 2007] Rajarshi Gupta, John Musacchio and Jean Walrand. *Sufficient rate constraints for QoS flows in ad-hoc networks.* Ad Hoc Networks, vol. 5, no. 4, pages 429–443, 2007. (Cited in page 32.)

[Hagberg 2008] Aric Hagberg, Pieter Swart and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX.* Technical Report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. (Cited in page 106.)

[Haque 2016] I. T. Haque and N. Abu-Ghazaleh. *Wireless Software Defined Networking : A Survey and Taxonomy.* IEEE Communications Surveys & Tutorials, vol. 18, no. 4, pages 2713–2737, 2016. (Cited in page 27.)

[Houidi 2011] Ines Houidi, Wajdi Louati, Walid Ben Ameur and Djamal Zeghlache. *Virtual network provisioning across multiple substrate networks.* Computer Networks, vol. 55, no. 4, pages 1011 – 1023, 2011. Special Issue on Architectures and Protocols for the Future Internet. (Cited in page 78.)

[Huang 2018] Nanyang Huang, Qing Li, Dong Lin, Xiaowen Li, Gengbiao Shen and Yong Jiang. *Software-Defined Label Switching : Scalable Per-flow Control in SDN.* 2018. (Cited in page 97.)

[Internet2 2018] Internet2. *Internet2 Network Infrastructure Topology.* `www.internet2.edu/media/medialibrary/2013/07/31/Internet2-Network-Infrastructure-Topology.pdf`, 2018. (Cited in page 96.)

[Iyer 2013] A. S. Iyer, V. Mann and N. R. Samineni. *SwitchReduce : Reducing switch state and controller involvement in OpenFlow networks.* In 2013 IFIP Networking Conference, pages 1–9, May 2013. (Cited in page 97.)

[Jain 2003] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan and Lili Qiu. *Impact of Interference on Multi-hop Wireless Network Performance.* In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom '03, pages 66–80, New York, NY, USA, 2003. ACM. (Cited in pages 14 and 32.)

[Jeon 2006] SungEok Jeon. *Topology aggregation : Merged-star method for multiple non-isomorphic topology subgraphs.* Computer Communications, vol. 29, pages 1959–1962, 2006. (Cited in page 76.)

[Jiang 2007] W. Jiang, S. Liu, Y. Zhu and Z. Zhang. *Optimizing Routing Metrics for Large-Scale Multi-Radio Mesh Networks.* In 2007 International Conference on Wireless Communications, Networking and Mobile Computing, pages 1550–1553, Sept 2007. (Cited in page 13.)

[Jin 2013] Xin Jin, Li Erran Li, Laurent Vanbever and Jennifer Rexford. *SoftCell : Scalable and Flexible Cellular Core Network Architecture.* In Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT '13, pages 163–174, New York, NY, USA, 2013. ACM. (Cited in page 97.)

[Jmila 2015] H. Jmila and D. Zeghlache. *An adaptive load balancing scheme for evolving virtual networks.* In 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), pages 492–498, Jan 2015. (Cited in page 62.)

[Katta 2014] Naga Katta, Omid Alipourfard, Jennifer Rexford and David Walker. *Infinite CacheFlow in Software-defined Networks.* In Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14, pages 175–180, New York, NY, USA, 2014. ACM. (Cited in page 97.)

[Keshav 1991] Srinivasan Keshav. *A Control-theoretic Approach to Flow Control.* In Proceedings of the Conference on Communications Architecture &Amp ; Protocols, SIGCOMM '91, pages 3–15, New York, NY, USA, 1991. ACM. (Cited in page 13.)

[Khalil 2017] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina and Le Song. *Learning combinatorial optimization algorithms over graphs.* In Advances in Neural Information Processing Systems, pages 6348–6358, 2017. (Cited in page 105.)

[Khan 2017] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani and M. K. Khan. *Topology Discovery in Software Defined Networks : Threats, Taxonomy, and*

State-of-the-Art. IEEE Communications Surveys Tutorials, vol. 19, no. 1, pages 303–324, Firstquarter 2017. (Cited in page 104.)

[Ku 2014] Ian Ku, You Lu and Mario Gerla. *Software-defined mobile cloud : Architecture, services and use cases.* In Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International, pages 1–6. IEEE, 2014. (Cited in page 9.)

[Labrador 2010] Miguel A. Labrador and Pedro M. Wightman. Topology control in wireless sensor networks : With a companion simulation tool for teaching and research. Springer Publishing Company, Incorporated, 2010. (Cited in page 14.)

[Lee 2016] Won Jin Lee, Jung Wan Shin, Hwi Young Lee and Min Young Chung. *Testbed implementation for routing WLAN traffic in software defined wireless mesh network.* In 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), pages 1052–1055, July 2016. (Cited in page 28.)

[Li 2004] Deying Li, Xiaohua Jia and Hai Liu. *Energy efficient broadcast routing in static ad hoc wireless networks.* IEEE Transactions on Mobile Computing, vol. 3, no. 2, pages 144–151, April 2004. (Cited in page 75.)

[Li 2017] M. Li, C. Hua, C. Chen and X. Guan. *Application-driven virtual network embedding for industrial wireless sensor networks.* In 2017 IEEE International Conference on Communications (ICC), pages 1–6, May 2017. (Cited in page 30.)

[Liang Geng 2018] Slawomir Kuklinski et al. Liang Geng. *Problem Statement of Common Operation and Management of Network Slicing.* `https://tools.ietf.org/id/draft-geng-coms-problem-statement-03.html`, 2018. (Cited in page 79.)

[LLDP 2009] LLDP. *IEEE Station and Media Access Control Connectivity Discovery.* IEEE Standard 802.1 AB (LLDP), 2009. (Cited in page 7.)

[Lu 2013] T. Lu and J. Zhu. *Genetic Algorithm for Energy-Efficient QoS Multicast Routing.* IEEE Communications Letters, vol. 17, no. 1, pages 31–34, January 2013. (Cited in page 42.)

[Lv 2012] Pin Lv, Xudong Wang and Ming Xu. *Virtual Access Network Embedding in Wireless Mesh Networks.* Ad Hoc Netw., vol. 10, no. 7, pages 1362–1378, septembre 2012. (Cited in page 30.)

[Mijumbi 2014] R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. D. Turck and S. Latré. *Dynamic resource management in SDN-based virtualized networks.* In 10th International Conference on Network and Service Management (CNSM) and Workshop, pages 412–417, Nov 2014. (Cited in page 70.)

[Neishaboori 2008] A. Neishaboori and G. Kesidis. *SINR-sensitive routing in wireless 802.11 mesh networks.* In 2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pages 623–628, Sept 2008. (Cited in page 13.)

[NetworkX 2018] NetworkX. *NetworkX Graph Generators.* `http://networkx.github.io/documentation/networkx-1.7/reference/generators.html`, 2018. (Cited in page 83.)

[Nurmio 2015] J. Nurmio, E. Nigussie and C. Poellabauer. *Equalizing Energy Distribution in Sensor Nodes through Optimization of RPL.* In 2015 IEEE International Conference on Computer and Information Technology ; Ubiquitous Computing and Communications ; Dependable, Autonomic and Secure Computing ; Pervasive Intelligence and Computing, pages 83–91, Oct 2015. (Cited in page 13.)

[OF-Config 2010] OF-Config. *OpenFlow Management and Configuration Protocol, ONF TS-016.* `www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf`, 2010. (Cited in page 20.)

[OFDP 2017] OFDP. *OpenFlow Discovery Protocol and Link Layer Discovery Protocol.* `groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol`, 2017. [Online]. (Cited in pages 7 and 8.)

[ONF 2016] ONF. *SDN Architecture for Transport Networks.* `https://www.opennetworking.org/wp-content/uploads/2014/10/SDN_Architecture_for_Transport_Networks_TR522.pdf`, mars 2016. (Cited in page 2.)

[OpenDaylight 2017] OpenDaylight. *OpenDaylight Controller.* `www.opendaylight.org`, 2017. [Online]. (Cited in page 7.)

[OpenFlow 2015] OpenFlow. *OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06).* `www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf`, 2015. (Cited in page 18.)

[P. N. 2013] Tran P. N. and Timm-Giel A. *Reconfiguration of virtual network mapping considering service disruption.* In 2013 IEEE International Conference on Communications (ICC), pages 3487–3492, June 2013. (Cited in page 62.)

[Pakzad 2016] Farzaneh Pakzad, Marius Portmann, Wee Lum Tan and Jadwiga Indulska. *Efficient Topology Discovery in OpenFlow-based Software Defined Networks.* Comput. Commun., vol. 77, no. C, pages 52–61, mars 2016. (Cited in page 104.)

[Parsa 1998] M. Parsa, Qing Zhu and J. J. Garcia-Luna-Aceves. *An iterative algorithm for delay-constrained minimum-cost multicasting.* IEEE/ACM Transactions on Networking, vol. 6, no. 4, pages 461–474, Aug 1998. (Cited in page 42.)

[Peng 2015] Yuhuai Peng, Lei Guo, QingXu Deng, Zhaolong Ning and Lingbing Zhang. *A novel hybrid routing forwarding algorithm in sdn enabled wireless mesh networks.* In High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety

and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on, pages 1806–1811. IEEE, 2015. (Cited in page 9.)

[Pfaff 2015] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J. Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Jonathan Stringer, Pravin Shelar, Keith Amidon and Martín Casado. *The Design and Implementation of Open vSwitch.* In Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, NSDI'15, pages 117–130, Berkeley, CA, USA, 2015. USENIX Association. (Cited in page 106.)

[Pignolet 2017] Y. Pignolet, S. Schmid and G. Tredan. *Load-Optimal Local Fast Rerouting for Resilient Networks.* In 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 345–356, June 2017. (Cited in page 105.)

[Qian 2009] Li Ping Qian, Ying Jun Zhang and Jianwei Huang. *MAPEL : Achieving global optimality for a non-convex wireless power control problem.* IEEE Transactions on Wireless Communications, vol. 8, no. 3, pages 1553–1563, 2009. (Cited in page 14.)

[Qu 2016] Ying Qu, Bryan Ng and Winston Seah. *A survey of routing and channel assignment in multi-channel multi-radio WMNs.* Journal of Network and Computer Applications, vol. 65, pages 120 – 130, 2016. (Cited in page 14.)

[Ramanathan 2000] Ram Ramanathan and Regina Rosales-Hain. *Topology control of multihop wireless networks using transmit power adjustment.* In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 404–413. IEEE, 2000. (Cited in page 14.)

[Ryu 2017] Ryu. *Ryu Controller.* `osrg.github.io/ryu`, 2017. [Online]. (Cited in pages 7 and 106.)

[Samuel 2013] Fady Samuel, Mosharaf Chowdhury and Raouf Boutaba. *PolyViNE : policy-based virtual network embedding across multiple domains.* Journal of Internet Services and Applications, vol. 4, no. 1, page 6, Mar 2013. (Cited in page 78.)

[Seah 2006] W. K. G. Seah, I. I. Er, W. K. G. Seah and I. I. Er. *Distributed Steiner-Like Multicast Path Setup for Mesh-based Multicast Routing in Ad Hoc Networks.* In IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), volume 2, pages 192–197, June 2006. (Cited in page 75.)

[Sebastian 2010] A. Sebastian, M. Tang, Y. Feng and M. Looi. *A Multicast Routing Scheme for Efficient Safety Message Dissemination in VANET.* In 2010 IEEE Wireless Communication and Networking Conference, pages 1–6, April 2010. (Cited in page 75.)

[Siek 2001] Jeremy G Siek, Lie-Quan Lee and Andrew Lumsdaine. Boost graph library : User guide and reference manual, the. Pearson Education, 2001. (Cited in page 106.)

[Stasi 2013] G. Di Stasi, S. Avallone and R. Canonico. *Virtual network embedding in wireless mesh networks through reconfiguration of channels.* In 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 537–544, Oct 2013. (Cited in page 30.)

[Subramanian 2006] A. P. Subramanian, M. M. Buddhikot and o. Miller. *Interference aware routing in multi-radio wireless mesh networks.* In 2006 2nd IEEE Workshop on Wireless Mesh Networks, pages 55–63, Sept 2006. (Cited in page 13.)

[Tan 2012] W. L. Tan, P. Hu and M. Portmann. *SNR-Based Link Quality Estimation.* In 2012 IEEE 75th Vehicular Technology Conference (VTC Spring), pages 1–5, May 2012. (Cited in page 13.)

[Tegueu 2016] A. F. S. Tegueu, S. Abdellatif, T. Villemur, P. Berthou and T. Plesse. *Towards application driven networking.* In 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pages 1–6, June 2016. (Cited in page 71.)

[Tegueu 2017] A. F. S. Tegueu, S. Abdellatif, T. Villemur and P. Berthou. *A reactive resource defragmentation method for Virtual Links Mapping in software-defined networks.* In 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pages 1–6, June 2017. (Cited in page 29.)

[Toh 1997] Chai-Keong Toh. *Associativity-Based Routing for Ad Hoc Mobile Networks.* Wireless Personal Communications, vol. 4, no. 2, pages 103–139, Mar 1997. (Cited in page 15.)

[Toh 2001] C. . Toh. *Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks.* IEEE Communications Magazine, vol. 39, no. 6, pages 138–147, June 2001. (Cited in page 13.)

[Uludag 2005] Suleyman Uludag, Klara Nahrstedt, King shan Lui and Greg Brewster. *Comparative Analysis of Topology Aggregation Techniques and Approaches for the Scalability of QoS Routing.* Technical Report, DePaul University, 2005. (Cited in page 76.)

[Valadarsky 2017] Asaf Valadarsky, Michael Schapira, Dafna Shahaf and Aviv Tamar. *Learning to Route.* In Proceedings of the 16th ACM Workshop on Hot Topics in Networks, HotNets-XVI, pages 185–191, New York, NY, USA, 2017. ACM. (Cited in page 105.)

[Vinyals 2015] Oriol Vinyals, Meire Fortunato and Navdeep Jaitly. *Pointer networks.* In Advances in Neural Information Processing Systems, pages 2692–2700, 2015. (Cited in page 105.)

[Wu 2018] Hsiao-Chun Wu, Cristiano Akamine, Bo Rong, Manuel Velez, Chenwei Wang and Jintao Wang. *Point-to-Multipoint Communications and Broadcasting in 5G.* Comm. Mag., vol. 56, no. 3, pages 72–73, mars 2018. (Cited in page 1.)

[Yang 2005] Yaling Yang, Jun Wang and Robin Kravets. *Interference-aware load balancing for multihop wireless networks.* Technical Report, 2005. (Cited in page 13.)

[Yu 2017] H. C. Yu, G. Quer and R. R. Rao. *Wireless SDN mobile ad hoc network : From theory to practice.* In 2017 IEEE International Conference on Communications (ICC), pages 1–7, May 2017. (Cited in page 28.)

[Yun 2013] Donggyu Yun, Jungseul Ok, Bongjhin Shin, Soobum Park and Yung Yi. *Embedding of virtual network requests over static wireless multihop networks.* Computer Networks, vol. 57, no. 5, pages 1139 – 1152, 2013. (Cited in page 30.)

[Zhao 2015] L. Zhao, J. Hua, X. Ge and S. Zhong. *Traffic engineering in hierarchical SDN control plane.* In 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), pages 189–194, June 2015. (Cited in page 76.)

[Zhu 2006] Y. Zhu and M. Ammar. *Algorithms for Assigning Substrate Network Resources to Virtual Network Components.* In Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, pages 1–12, April 2006. (Cited in page 62.)

[Zhu 2015] Z. Zhu, C. Chen, X. Chen, S. Ma, L. Liu, X. Feng and S. J. B. Yoo. *Demonstration of Cooperative Resource Allocation in an OpenFlow-Controlled Multidomain and Multinational SD-EON Testbed.* Journal of Lightwave Technology, vol. 33, no. 8, pages 1508–1514, April 2015. (Cited in page 78.)

[Zuo 2019] Yuan Zuo, Yulei Wu, Geyong Min and Laizhong Cui. *Learning-based network path planning for traffic engineering.* Future Generation Computer Systems, vol. 92, pages 59 – 67, 2019. (Cited in page 105.)