Conception et Ingénierie de Réseaux Nouvelle Génération Orientés Ethernet

Yannick Brehon

THÈSE

Présentée pour obtenir le grade de

Docteur de l'École Nationale Supérieure des Télécommunications de Paris

Spécialité : Informatique et Réseaux

par

Yannick BRÉHON

Conception et Ingénierie
des Réseaux de Nouvelle Génération Orientés Ethernet

Soutenue le 11 mai 2007 devant la Commission d'Examen :

Jean-Claude Bermond (INRIA)	Président
Michal Pióro (WUT & ULUND)	Rapporteur
Deepankhar Medhi (UMKC)	Rapporteur
Emmanuel Dotaro (Alcatel-Lucent)	Examinateur
Daniel Kofman (ENST)	Directeur de Thèse
Augusto Casaca (INESC-ID)	Co-Directeur de Thèse

A Amélie.

Résumé

Introduction

Les opérateurs de réseaux tendent aujourd'hui à déployer la technologie Ethernet le plus largement possible, dans les segments d'accès, métropolitains et de cœur, afin de construire les Réseaux de Nouvelle Génération. Cette mouvance s'accompagne d'un besoin d'optimiser l'utilisation des ressources du réseau, et de réduire les coûts opérationnels et d'infrastructure.

Les Réseaux de Nouvelle Génération sont à l'étude afin de satisfaire les besoins futurs des utilisateurs. Dès aujourd'hui, les usages évoluent et de nouveaux services émergent. Du fait de l'augmentation à la fois de la bande passante mise à la disposition de l'utilisateur et de la puissance des ordinateurs personnels, on observe l'apparition de nouveaux services et applications, comme le *peer-to-peer*, la télévision Haute Définition, la convergence fixe-mobile, etc. En ce qui concerne les services aux professionnels, et du fait des mêmes facteurs, de nouveaux services émergent, tels que des réseaux privés virtuels à très haut débit, des *grids* (plateformes de calcul distribué), la visioconférence. C'est la satisfaction de tous ces besoins que les Réseaux Nouvelle Génération doit offrir. Or du fait de l'agressivité du marché, constamment sous l'assaut de nombreux nouveaux entrants, les opérateurs doivent être de plus en plus efficaces tout en offrant de plus en plus de services.

Dans ces conditions, l'intégration et l'optimisation multi-couches est un problème crucial à résoudre pour les opérateurs, afin de réduire notamment les coûts opérationnels, directement liés à la quantité de main d'œuvre nécessaire à la bonne gestion et surveillance du réseau. En parallèle, les opérateurs ont besoin de méthodes d'ingénierie de trafic afin d'utiliser au mieux les ressources de leur réseau. L'utilisation croissante d'Ethernet comme technologie prévalente dans tous les segments du réseau est l'une des réponses à ces problèmes.

Ethernet est la technologie de couche 2 (selon le modèle OSI [Zim80]) la plus répandue chez l'utilisateur final, grâce à sa simplicité de déploiement et d'utilisation et son bas coût. Afin de réduire les coûts d'intégration, les opérateurs cherchent donc à pousser Ethernet de plus en plus profondément dans le réseau. Dans les réseaux de cœur, l'une

des tâches du groupe de travail *Common Control And Measurement Plane (CCAMP)* de l'IETF a donc été de rendre Ethernet compatible avec les contraintes propres à ce segment, on parle d'Ethernet *carrier-class*: cet effort s'est concrétisé sous la forme des *Layer 2 LSPs* [Pap04], qui font d'Ethernet une solution orientée connexion, contrôlée avec GMPLS. Dans le segment métropolitain, les Réseaux Privés Virtuels Provisionnés par le Fournisseur (PPVPN) sont utilisés pour fournir les services d'interconnexion entre les sites de leurs clients. Ce service permet une communication privée et sécurisée entre ces sites en passant au travers d'une infrastructure commune et partagée, telle qu'Internet.

Contribution au dimensionnement des réseaux métropolitains

Dans le segment métropolitain des réseaux, Ethernet est déjà largement déployé et utilisé, en particulier afin de fournir le service de Réseaux Privés Virtuels (RPV). Néanmoins, ce déploiement est basé sur des protocoles qui évoluent afin d'offrir des fonctionnalités nouvelles et améliorées d'ingénierie de trafic.

Ethernet utilise le *Spanning Tree Protocol* pour établir un arbre couvrant de communication, qui garantit qu'il n'y aucune boucle dans le transport de l'information d'une source vers une destination. Un arbre couvrant n'utilise par définition qu'un nombre restreint de liens (un de moins que le nombre total de nœuds dans le réseau). De ce fait, l'utilisation d'un seul arbre soutenant tous les RPV implique que de nombreux liens ne sont pas du tout utilisés, tandis que d'autres liens peuvent être surchargés: il n'y a pas d'équilibrage de la charge.

Avec l'apparition du *Multiple Spanning Trees Protocol* [IEE02], plusieurs arbres couvrants différents peuvent être utilisés conjointement sur un même réseau. Il est ensuite possible de faire correspondre un ou plusieurs RPV à un arbre couvrant - mais un même RPV ne peut pas utiliser plus d'un seul arbre. Ces arbres de communication comprennent tous des liens du réseau, mais en utilisant plusieurs arbres, il est possible de répartir la charge du réseau sur plus de liens. Un problème d'optimisation double se pose donc: il faut choisir l'ensemble des arbres couvrants de façon optimale, et associer à chacun de ces arbres un ou plusieurs RPV de façon optimale. L'optimalité peut se définir selon divers critères. On peut chercher à minimiser l'usage global des ressources, ou bien minimiser la charge du lien le plus chargé (cela permet de garantir une augmentation proportionnelle maximale du trafic accepté dans le réseau, et revient à équilibrer la charge), ou encore, avoir des arbres les plus divers possibles afin de garantir la plus grande tolérance aux pannes.

Dans notre étude présentée dans le chapitre 2, nous formulons mathématiquement ce problème sous la forme d'un programme d'optimisation non linéaire avec variables entières. Néanmoins, il n'est pas possible de le résoudre de façon exacte avec des solveurs commerciaux de type CPLEX [CPLEX]. De ce fait nous présentons une heuristique de résolution

très rapide et efficace. Nos résultats sont comparés à ceux d'autres méthodes présentes dans la littérature scientifique et obtiennent de meilleurs résultats, tant sur la répartition de la charge que sur la résistance aux pannes. En moyenne, sur les trois réseaux testés, notre méthode permet une réduction de 8% de la charge du lien le plus chargé, tout en n'augmentant la charge globale que marginalement. L'augmentation globale de la charge est nécessaire, mais est due à l'équilibrage de la charge de quelques liens les plus chargés vers les liens les moins chargés. Elle n'impacte donc pas les coûts effectifs de l'opérateur, qui au contraire, peut voir son trafic évoluer linéairement de façon plus importante.

Présentation du concept de bus-LSPs

Dans les réseaux orientés-connexions avec un plan de contrôle GMPLS, on appelle *Label Switched Path(LSP)* une connexion entre un équipement d'entrée (*ingress*) et un équipement de sortie (*egress*). Cette connexion utilise un ou plusieurs liens reliant des équipements appelés *Label Switching Routers (LSRs)*. Ces liens peuvent être des liens physiques, comme des fibres, ou des LSPs de plus bas niveau servant à créer le LSP supérieur. Une adjacence d'acheminement, ou *Forwarding Adjacency (FA)*, est une représentation dans le plan utilisateur de la connectivité qu'un ou plusieurs LSPs établissent entre le nœud ingress et le nœud egress. Cette FA peut ensuite être utilisée dans les couches supérieures, tout comme un lien physique.

Nous introduisons dans le chapitre 3 les *bus-LSP* et les *bus-FA* qui sont des extensions de ces concepts de LSP et de FA. Avec un bus-LSP, les LSRs intermédiaires ont la possibilité d'injecter et d'extraire du trafic circulant sur le LSP. De ce fait, le bus-LSP offre une connectivité entre les nœuds ingress, egress et nœuds intermédiaires. La bus-FA est une représentation de cette connectivité dans le plan utilisateur. Comme le bus-LSP est unidirectionnel, la donnée ne peut être envoyée que vers une destination en aval de la source.

Un mécanisme permet à tout nœud (ingress ou intermédiaire) d'indiquer dans le paquet émis pour quel destinataire il est adressé, par un système de compteur décrémenté dans chaque nœud. Comme avec les LSPs, il est possible pour un bus-LSP d'utiliser un autre bus-LSP (ou un segment de celui) comme lien virtuel (principe de *nesting*).

Nous présentons aussi dans le chapitre 3 une manière de représenter les bus-LSPs dans les graphes modélisant les réseaux.

Bus-LSPs dans des environnements mono-couches

Dans le chapitre 4, nous étudions le problème d'optimisation sous-jacent à l'introduction de bus-LSPs dans des réseaux mono-couches. Dans ce type de réseau, l'intuition suggère

que les bus-LSPs doivent permettre une réduction du nombre total de connexions établies, et de ce fait, de réduire les coûts opérationnels du réseau. En effet, moins il y a de connexions, moins il y a besoin de personnel pour surveiller et gérer le réseau.

Le problème est donc formulé comme deux programmes d'optimisation linéaire avec variables entières. Le premier cherche à minimiser uniquement ces coûts opérationnels, tandis que le second vise à réduire ces coûts mais en maintenant une utilisation de la bande passante disponible dans le réseau au minimum. Une résolution de ces programme est réalisée pour deux réseaux d'illustration, et les résultats confirment donc l'hypothèse. En effet, on observe une réduction significative, dans tous les cas, du nombre de connexions nécessaires aux opérations du réseau (de 75% à 95% pour l'étude sur les coûts opérationnels simples, de 66% à 87% pour l'étude incluant l'utilisation optimale de la bande passante du réseau).

Bus-LSPs dans des environnements multi-couches

Dans le chapitre 5, nous étudions le problème d'optimisation induit par l'introduction de bus-LSPs dans des réseaux multi-couches. Ici, l'étude qualitative réalisée dans le chapitre 3 permet d'espérer une réduction de la quantité de trafic circulant dans les couches supérieures du réseau, une meilleure utilisation des ressources du réseau et à nouveau une réduction des coûts opérationnels par une baisse du nombre total de connexions établies.

Le problème est formulé mathématiquement sous la forme d'un programme d'optimisation linéaire avec variables entières. Ce problème est trop complexe pour obtenir des résultats issus d'une résolution exacte. C'est pourquoi nous proposons une approche heuristique (non-exacte). Pour cela, nous avons développé des variantes des algorithmes MIRA [KKL00] et Dijkstra [Dij59], respectivement PIBRA et FT-SPF. Ces algorithmes sont à la base de notre heuristique, dont les détails sont donnés dans la section 5.3.

Cette heuristique obtient de bons résultats pour tous les critères jugés. Ainsi, sur les deux réseaux de cœur utilisés pour la tester, nous obtenons une quantité de trafic routée dans la couche supérieure quasi-nulle, même sous de fortes charges. Le nombre de bus-LSPs établis est très réduit (6 bus pour 110 demandes dans un cas, 20 bus pour 420 demandes dans l'autre). Le délai de propagation n'est quasiment pas augmenté bien que la répartition de charge permette une grande augmentation linéaire de la matrice de trafic. C'est sur ce dernier point néanmoins que l'approche montre ses limites pour le plus grand des deux réseaux, et dans la section 5.5, nous présentons donc des pistes pour améliorer l'approche.

Intégration des bus-LSPs et des bus-FAs dans un plan de contrôle GMPLS

Afin de permettre une utilisation concrète des bus-LSPs dans les réseaux de cœur, il est nécessaire de fournir les outils pour les manipuler dans le plan de contrôle. Dans le chapitre 6, nous les présentons pour le protocole de routage interne de GMPLS (OSPF-TE [RFC3630]) et pour le protocole de signalisation (RSVP-TE [RFC3473]).

Le protocole de routage permet l'échange d'informations de routage entre les divers nœuds appartenant au réseau. De ce fait il est naturel de l'utiliser pour que les nœuds communiquent leur support éventuel des bus-LSPs. De plus, il est nécessaire d'annoncer dans le réseau les bus-FAs induites par les bus-LSPs. L'extension au protocole proposée permet de faire ceci tout en étant rétro-compatible avec les versions antérieures du protocole.

Au niveau de la signalisation, l'introduction des bus-LSPs introduit de nouvelles contraintes dans le réseau. Il faut en effet pouvoir rediriger le bus, mais aussi changer ses extrémités sans avoir d'effet sur les nœuds intermédiaires, ou encore fusionner plusieurs bus en un seul. Après avoir analysé comment ces fonctions peuvent être réalisées, pour certaines d'entre elles, avec les blocs existant de RSVP-TE (section 6.2.3), l'impossibilité de manipuler aisément les bus-LSPs sans une extension ad-hoc est indéniable. Naturellement, nous introduisons une extension au protocole (section 6.2.4) qui permet de remplir intégralement ce cahier des charges en un nombre minimal d'opérations.

Acknowledgements

I would like to thank my thesis director, Daniel Kofman for the patience he showed all along my years preparing this thesis. He found time to advise me, both scientifically and morally. He brought out the ideas which were in me, helped me formulate them and investigate them. For this and for helping me achieve my objectives, I am very grateful.

I sincerely thank Professor Augusto Casaca from INESC-ID, Lisboa, for having been my co-advisor. He followed my progress and gave me valuable input continuously. I also want to thank him for the great help he provided on the article we wrote together. Generally, he helped me structure my work and the thesis, and gave me very sound advice.

Professor Pióro and Professor Medhi have agreed to be the reviewers of my thesis, and this regardless of all the obstacles. Despite the very short deadlines, they gave me their comments on my thesis and helped improve it. Also, Professor Pióro provided major contributions to the formulation of problem presented in Section 5.2.2. I also thank Professor Bermond and Emmanuel Dotaro for being examiners of the thesis, I know I have imposed on their heavily loaded schedules.

I also wish to thank all the people at the RHD department of the ENST, who have made my time spent at the ENST an enjoyable one. In particular, I wish to thank Ramon Casellas and Jean-Louis Rougier who I have always been able to count on, and who provided a large amount of the work on the control plane issues tackled in this thesis. Also I wish to thank Myriam Morcel whose kindness and helpfulness I have always been able to rely on.

I have appreciated the possibility to collaborate with great people at Alcatel-Lucent, namely Emmanuel Dotaro and Martin Vigoureux, and then join them for my career. They gave me time to complete my thesis, and welcomed me, in the PTI research team.

For her personal and moral support, I am extremely grateful to Amélie Cook, the love of my life. She has always believed in me and showed it, I deeply thank her for it. To my parents, I am grateful and glad to make them proud, as it is their love and education which has made me the man I am today. Also, I thank my brother David, he has made time and been there for me when I needed him; in short, he has been a great brother. Last but not least, I wish to thank all my beloved friends who have meant so much along the years, *Ze Groop* and *Les Frères Daltons* in particular.

Abstract

The increase in individual and professional customer expectations, as well as the fast technological evolutions, are leading to the design of the Next Generation Internet (NGI). Network optimization allows operators to efficiently deliver high-quality services at reduced costs. Due to the pre-eminence of Ethernet technologies at the customer premises and in the access network segment, and due to its low cost, Ethernet is a natural candidate to serve as the technology of choice of the NGI.

In the metropolitan network segment, efforts have led to specifying Ethernet flavors which allow mapping client traffic into *Virtual LANs*. In this thesis, we investigate how to efficiently assign these VLANs to the Spanning Trees generated by Ethernet's Spanning Tree Protocol, since this is currently the only way to perform traffic-engineering and network optimization in this network segment. We describe a way of generating the Spanning Trees and mapping the VLANs, which

In the core network segment, several initiatives aim at turning Ethernet into a GMPLS-controlled connection-oriented technology (such as Layer 2 LSPs and PBB-TE). In this thesis, a new type of connection for packet- and frame- based, connection-oriented and GMPLS-controlled technologies is introduced and studied: the bus-LSP. Both in single-layer and in multi-layer networks, it provides the operator with significant cost reduction. We numerically quantify this reduction, as well as engineering methods for efficiently deploying bus-LSPs. We also detail the control plane protocols extensions needed to manage these bus-LSPs.

List of Abbreviations

AD	Add-Drop
ADSL	Asymmetric Digital Subscriber Line
AP	Access Point
ATM	Asynchronous Transfer Mode
CAPEX	Capital Expenditure
CCAMP	Common Control And Measurement Plane
CE	Customer Edge
C-VLAN	Customer Virtual Local Area Network
DFGM	Diversified Forest with Greedy Mapping
DSL	Digital Subscriber Line
DWDM	Dense Wavelength Division Multiplexing
EGP	External Gateway Protocol
ERO	Explicit Route Object
FA	Forwarding Adjacency
FCAPS	Fault-management, Configuration, Accounting, Performance, Security
FDDI	Fiber Distributed Data Interface
FTTx	Fiber To The Home/Curb/Office/etc.
FT-SPF	Forbidden Turns - Shortest Path First
GMPLS	Generalized Multi-Protocol Label Switching
IETF	Internet Engineering Task Force
IGP	Internal Gateway Protocol
ISP	Internet Service Provider
L1, L2, L3	Layer 1, Layer 2, Layer 3 (of the OSI model)
L1-VPN, L2-VPN, L3-VPN	Layer 1-, Layer 2-, or Layer 3-Virtual Private Network
LAN	Local Area Network
LSA	Link State Advertisement
LSP	Label Switched Path
LSR	Label Switching Router

MEF	Metro Ethernet Forum
MCL	Minimal Complexity Layout
MCL-BC	Minimal Complexity Layout with Bandwidth Constraint
MILP	Mixed-Integer Linear Program
MINLP	Mixed-Integer Non-Linear Program
MLXC	Multi-Layer Cross-Connect
MPLS	Multi-Protocol Label Switching
MSTP	Multiple Spanning Trees Protocol
OAM	Operations, Administration and Maintenance
OCX	Optical Cross-Connect
OPEX	Operational Expenditure
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OSPF-TE	Open Shortest Path First - Traffic Engineering extension
PAD	Packet Add-Drop
PCE	Path Computation Element
PE	Provider Edge
PIBRA	Potential and Interference-Based Routing Algorithm
PPVPN	Provider Provisioned Virtual Private Networks
PSC	Packet-Switching Capability
PWE3	Pseudo-Wire Emulation Edge-to-Edge
QoS	Quality of Service
RA	Router Advertisement
RRO	Record Route Object
RSTP	Rapid Spanning Tree Protocol
RSVP	Resource reSerVation Protocol
RSVP-TE	Resource reSerVation Protocol - Traffic Engineering extension
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SP	Service Provider
SRRG	Shared Risk Resource Group
ST	Spanning Tree
STP	Spanning Tree Protocol

TDM	Time-Division Multiplexing
TE	Traffic Engineering
TLV	Type, Length, Value
VLAN	Virtual Local Area Network
VNT	Virtual Network Topology
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
VPWS	Virtual Private Wire Service
VTD	Virtual Topology Design
VTHD	Vraiment Très Haut Débit
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing
WLAN	Wireless Local Area Network

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

There is currently a trend followed by network operators, for deploying the Ethernet technology as widely as possible, in the access, the metropolitan and the core topological segments, to build the Next Generation Networks. This trend is accompanied by a need to optimize network usage and reduce capital and operational costs. We will present here this general context as the framework for our thesis.

1.1 Ethernet in the Next Generation Networks

Next Generation Networks are being developed today so as to tomorrow provide the end-user with a new and improved experience. Due to the simultaneous increase of bandwidth offered to the customer and to the exponential improvement of personal computers' power, the services proposed to the user are made better continuously. Offers such as Triple-Play have become standard in many places, offering High-Speed Internet (HSI) access, telephone connection and television over a single DSL connection. The experience of the Internet itself has changed a lot, due to applications such as peer-to-peer filesharing, gaming, and "Web 2.0" browsing among others. The telephone service include Plain Old Telephony Service (POTS), but with VoIP (Voice over IP), new services are being developed, such as mobile-fixed line handover and video conferencing. Many additional video services are being developed, such as IPTV, Video-On-Demand (VOD), which is more and more widely offered, High-Definition TeleVision (HDTV), personal video-conferencing, video sharing and broadcasting, and many more. All these services are very bandwidth-hungry and/or quality of service-demanding. At the same time, and for the same reasons, high-quality services are being developed and proposed to companies: high-speed virtual private networks, grid and distributed computing, video conferencing. Bandwidth at the access is therefore increasing by very big increments. In less than five years, DSL technologies have evolved from low-speed, a few hundred kilobits-per-second (kbps), to high speed, offering

downstream connection speeds in the vicinity of the megabit per second (Mbps) and even very high speed, with technologies such as ADSL2+ which boasts speeds of up to 28 Mbps. But even those speeds are obsolete as they are released, as we see technologies such as Fiber-To-The-Curb/Office/Building/Home (FTTx) appearing.

However, the end-user is not interested in having access to each service separately. The concept of always available service, "always on", has emerged and there is now a demand for access to all services from all locations and from various terminals: home computer, itinerant laptop, but also cell phone and public phone. Each of these terminals works with different sets of protocols and technologies. Therefore, one can talk of the evolution of multi-service networks into multi-network services. The operators therefore have to adapt their offer in many ways; they will in particular have to offer support for many different layers.

Network operators which are in a very competitive market, with many new entrants, have to focus on efficiency. This means multi-layer optimization is now a justified and significant concern for operators, as they do not want to have to operate multiple networks separately. Integration and cross layer optimization has therefore become a key problem to address, and improving cross-layer and cross-network integration will result in an operational expenditure (OPEX) reduction. OPEX is directly related to the manpower an operator must dedicate to his network management and monitoring. In parallel, operators need to develop traffic-engineering (TE) strategies which will allow them to use their resources at their full potential and thus reduce Capital Expenditure (CAPEX). This consists in developing techniques such as load balancing, so as to relieve the heavily loaded equipments while using more of other under-used ones. However, traffic engineering also has to deal with user-imposed constraints. These are known as Quality of Service (QoS) constraints, and can be expressed, among others, as a maximum delay from source to destination, as a minimal guaranteed bandwidth, as a maximal jitter (delay variation), etc. These constraints are often binding, through contracts called Service Level Agreements (SLAs).

In this context, a major trend is forming, pushing the use of Ethernet as far as possible into the network. Ethernet has been the technology of choice for end-users, its popularity has steadily increased and Ethernet now prevails as the most used layer 2 technology of the standard OSI model [Zim80]. This last fact is due to several strong points of Ethernet: it is a very cheap technology, it is very simple to deploy, and it is relatively efficient. The cost of Ethernet has always been low: initially, Ethernet's protocol simplicity allowed the equipments to be rapidly designed and brought to the market; later on, the fact that Ethernet pieces of equipments were so widespread allowed the costs to be reduced, the market being very competitive. Ethernet's popularity spawned from its very easy setup; in particular Ethernet networks are meshed and do not necessitate any special structure

such as Token Ring or FDDI's rings among others. It initially also was configuration free, which meant no particular technical skills were required to setup an Ethernet network: it was basically "plug and play". Ethernet is not the most efficient layer 2 technology available [SH80]. It is known not to scale very well, in particular on shared mediums. But its low cost allowed for high bandwidths to be designed and brought to the market very fast, compensating for its slight performance issues. Today, Ethernet has evolved quite a lot, and is now available in multiple flavors: switched Ethernet at speeds from 10 MBps to 100GBps. Wireless local area networks (WLANs) have also emerged recently, using 802.11 (WIFI) protocols. In a nutshell, these wireless protocols are a remodeling of Ethernet pushed into the air. This explains the very easy integration of WIFI networks in wireline Local Area Networks (LANs).

Operators and service providers are inclined to push Ethernet as far as possible into the networks in order to achieve previously mentioned integration, for both CAPEX and OPEX reasons: for example, ATM as a transport layer is disappearing in ADSL. In the core networks, an important work of the Common Control And Measurement Plane (CCAMP) group of the IETF has been extending Ethernet to become a carrier-class technology: this work has led to the specification of the Layer 2 LSPs [Pap04], which turn Ethernet into a circuit-oriented GMPLS-controlled technology. The Provider-Backbone Bridge - Traffic Engineering (PBB-TE) similarly aims at turning Ethernet into a connection-oriented technology, which is also supposed to use GMPLS as a control plane. In the metropolitan area, Provider Provisioned Virtual Private Networks (PPVPN) are focused on using Ethernet to bring the Virtual Local Area Network (VLAN) service to their customers.

1.2 The Virtual Private Networks Services

A Virtual Private Network (VPN) is a service which allows private communications over a public network infrastructure, as illustrated in Figure 1.1. For instance, a company may setup a VPN to connect its multiple sites over the Internet. A VPN may be setup at various levels of the OSI model: 1, 2, and 3, depending on the type of connection between the customer sites. According to [RFC4664], the term "provider provisioned VPNs" [RFC4026] refers to *Virtual Private Networks (VPNs) for which the Service Provider (SP) participates in management and provisioning of the VPN.* That is, the public network infrastructure is the provider network in this case. In all PPVPN solutions, the provider has border equipments, the Provider Edge (PE), which are connected to the client site's Client Edge (CE). The type of PE which is seen by the CE depends on the type of VPN, as will be detailed hereafter. In this thesis, we concentrate on PPVPNs, which are the VPNs found in the metropolitan network segment, though our results can be applied to all types of VPNs.

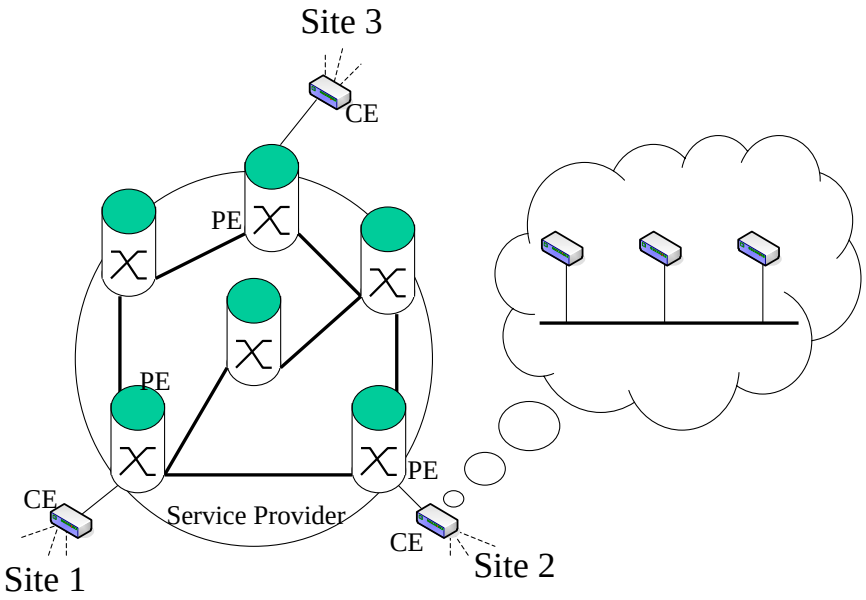


Figure 1.1: Provider Provisioned Virtual Private Network

If the customer wishes to interconnect equipments over a network which can only transport layer 1 (such as TDM or optical connections) traffic, he will require a layer 1 VPN. Note that Layer-1 VPNs are under definition at the IETF, and will require GMPLS. In this case, the PEs are optical cross-connects or SDH nodes for example, which will allow the customer to transparently connect distant sites with layer-1 connections.

There are two fundamentally different kinds of Layer 2 VPN service that a service provider could offer to a customer: Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS). A VPWS is a VPN service that supplies an L2 (Layer 2) point-to-point service. This is equivalent to having a fiber interconnecting two sites, or a leased line. However, with the emerging of MPLS-based provider networks, this service is most often based on Pseudo-Wires, according to the work [RFC3985, RFC4447] of the *Pseudo-Wire Emulation End-to-End (PWE3)* Working Group at the IETF.

A VPLS is an L2 service that emulates LAN service across a Wide Area Network (WAN). While the VPLS may be realized by a full-mesh of point-to-point connections between the various customer sites, as is the case with the current Pseudo-Wire-based solution (most popular), this is a very different type of service from the VPWS. Indeed, the customer subscribes to a single service, which allows him to interconnect all his sites, whereas a VPWS emulates a leased line.

Finally, the layer-3 VPN service allows the provider to setup routers which will directly be connected to the client sites. These various routers will be interconnected by the provider network in an opaque way to the client. The PEs are layer 3 routers in this case, and the client sees the provider network as a full mesh between all PEs.

The work which is presented in Chapter 2 fits into the context of VPNs which are established by interconnecting the PEs with an Ethernet network: this can be the case for layer-3 VPNs or layer-2 VPNs (using technologies such as 802.1ad, also known as Q-in-Q [IEE06b] and 802.1ah, also known as MAC-in-MAC [IEE06a]).

For all types of VPNs, one of the most popular ways (the only way actually, for L1-VPNs) for interconnecting the PEs is to establish a full-mesh of (G)MPLS-controlled tunnels [MR07]: in this context, the work on bus-LSPs (see Chapters 3, 4, 5, and 6) will allow such tunnels to be setup at a lesser cost (both in terms of OPEX and CAPEX) for the provider.

1.3 Service Providers and Transport Networks

The architecture of the Internet is organized in what is known as a tier-based approach: its organization is based on Service Providers which inter-operate according to a provider-client model. The end-user accesses the network through a series of aggregation networks which are in turn themselves clients of higher level networks. This range of networks spans from the personal area network (PAN), to the Local Area Network (LAN), to an Internet Service Provider network (ISP). This ISP provides the user with connectivity to the Internet: for businesses, this also means providing interconnection (VPN service) between various sites.

There are multiple tiers of ISPs. Tier-1 ISPs are in the core of the Internet and provide the interconnection of tier-2 ISPs. The tier model is hierarchical in that each tier-2 ISP provides interconnection for tier-3 ISPs and so on. This model also allows for what are known as "peering agreements", in which two Service Providers agree to connect directly and thus bypass a higher level ISP to interconnect them. Such peering agreements are put into effect when the volumes of traffic exchanged between the peers are equivalent, and the agreement is therefore interesting for both peers.

1.3.1 Transport Networks

The ISPs need transport functions, which they may or may not manage internally. This aspect is orthogonal to the service provided. By definition, the mission of the transport network is to provide end-to-end, transparent transport services to the client (which could be a 2 Mb/s circuit or a Gigabit Ethernet flow), with deterministic levels of reliability and availability across a network. The transport function is traditionally divided in three main segments. The first segment is the Access segment, which is often regarded as less profitable. The Access must bring the traffic from and to the end user: this is the copper phone line, FTTx or a leased line for example. The Metro Aggregation segment,

which covers distances between 20 and 50 kilometers, aggregates the traffic from various access segments, and provides metropolitan network coverage: this is an SDH ring for instance. The aggregation function of this network is to multiplex traffic from the Access; the aggregation process usually tends to homogenize the shape of the traffic. The Core segment deals with the largest traffic volumes and long geographical distances; they work at the highest bandwidths and granularities. It is agreed upon that typical distances covered by core networks are in the order of the hundreds of kilometers in general. Core networks are traditionally connection oriented networks: the data is not routed hop-by-hop at each core network node, but rather, connections are established between remote core nodes, and the traffic is tunneled through these connections. Connection-oriented networks are typically used for their traffic-engineering capabilities, the high bandwidth they provide and their strong protection schemes.

As client services are moving toward IP and Ethernet technologies, the transport network has to evolve to speak a similar packet-based language. There is a trend moving toward packet-oriented transport technologies that accommodate increased bandwidth while offering differentiated quality of service (QoS). Current Time-Division Multiplexing (TDM) transport network infrastructures require extensive upgrading – even rebuilding – with new overlays. The upgrades lead to higher revenues, but at greater cost. Where interoperability of TDM networks with packet based networks is a complex issue, Ethernet and IP/MPLS based transport is able to accomplish this interoperability in a smoother fashion. Moreover, packet-based transport has the key capability to aggregate and transport traffic in fine granularity, taking advantage of statistical multiplexing for optimal use of optical bandwidth at the lowest cost per bit. It does this while meeting traditional TDM features in terms of manageability, reliability and availability.

1.3.2 Network Planes

A network makes use of three separate sets of functions to operate, which are regrouped in three different planes: the *data plane*, the *control plane* and the *management plane*. Note that all these planes can be physically distinct networks. For instance, the control plane may be an IP network connecting various nodes and used to control an SDH data plane.

1.3.2.1 Data Plane

The *data plane* is responsible for forwarding data from source to destination, and therefore, in each node, the data plane functions allow the traffic to be received from an incoming interface and re-emitted on one (or more) outgoing interfaces.

In a connection oriented network, the data plane is made up of connections between various equipments. The main candidate for the Next Generation Networks transport architecture

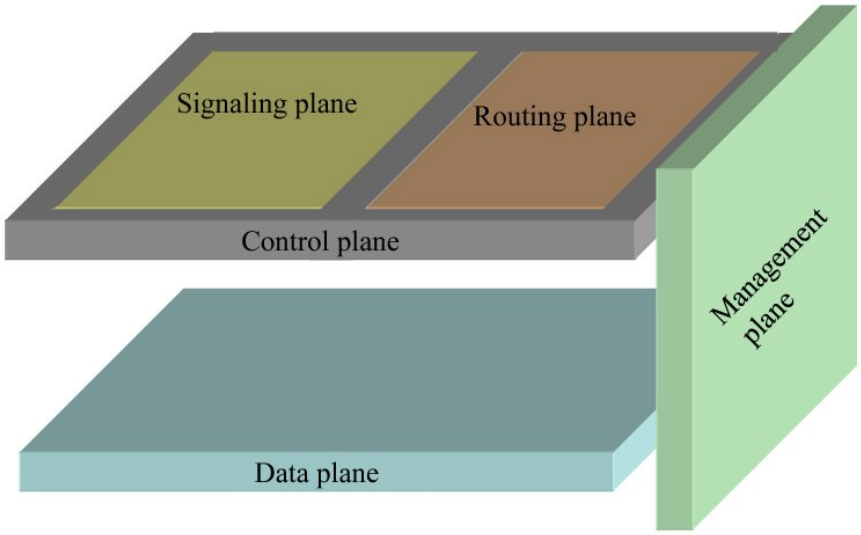


Figure 1.2: Network Planes

is the GMPLS framework [RFC3471], which follows the architecture presented here, and the same plane separation [RFC3945]. GMPLS is built as a generalization of MPLS-TE [RFC3209]. We will make use of the GMPLS terminology in this thesis, which we therefore introduce here. The data plane principle in all MPLS-like protocols is to switch a label. MPLS nodes are therefore Label Switching Routers (LSRs), and a connection is called a Label Switched Path (LSP). Such LSPs may span one or more links (these links may be physical or virtual links created by other connections, this is explained later). An LSP connects two LSRs called *ingress* and *egress* nodes, in such a way that all traffic from the ingress node reaches the egress node. Mapping traffic to a given LSP is part of the adaptation function of the data plane, which allows a given network to provide connectivity to a client layer/network. The adaptation function takes a given type of traffic as input and presents it to the data plane in a standardized form, which is ready for use. The adaptation functions are used at the exit of an LSP when the traffic needs to be passed on to the client layer.

An FA-LSP is a tunnel opened between two equipments, used as a Forwarding Adjacency (FA): this means its use is restricted to the data plane, although this is only a practical restriction. Data which goes into this tunnel will not be specifically treated until arrival at exit. Various categories of traffic can use a same tunnel and will be treated in the same way, because of the MPLS paradigm that a tunnel is agnostic to what it transports. Along an LSP, all packets are switched according to a label (MPLS paradigm). Therefore an FA-LSP is such an LSP used in the data plane.

Establishing such FA-LSPs defines a logical topology for the data plane, consisting in the LSRs (vertices) and the FAs (edges) connecting them: this is the virtual layout (see

Section 1.3.3). Indeed LSPs can be opened using as their segments not only physical links, but also FAs. This recursion is the basis to multi-layer networks, as they are defined in Section 1.3.3.

As has been shown in previous work, such as in [GS95, YKH95, GSM02, GM02, OSK+03, SL03, XST03, ZLYHG02], dynamically changing the virtual layout offers high use-effectiveness of the network resources. GMPLS signaling and routing has been designed to establish and release such LSPs on the fly, rather than provisioning these LSPs statically via management.

1.3.2.2 Control Plane

The *control plane* consists in the association of the *routing plane* and the *signaling plane*.

The *routing plane* allows network nodes to exchange information on the data plane topology: this information will then be used to perform the routing function in the data plane. The responsibility of deciding on the routes the traffic may follow in the network can be assigned to the source node of the traffic, this is *explicit source routing*, or distributed among all nodes which perform *hop-by-hop routing*. When explicit source routing is applied, the *ingress node* can perform:

- *strict routing*: all the intermediate nodes are explicitly specified.
- *loose routing*: some of the intermediate nodes may not be specified, in which case some of the intermediate nodes will have to obtain new routes for the missing segments.

The actual calculation of the route may however be delegated to a different network element, this is for example the case when using the Path Computation Element (PCE) architecture [RFC4655] defined in the IETF *PCE* working group. In all cases, the calculation of the route is not a part of the routing plane, as the latter is only responsible for the exchange of information on the data plane topology.

The *signaling plane* is responsible for the exchange of messages, in connection-oriented networks, between nodes taking part in a connection. These signaling messages control the connection, that is they are responsible for the setup and release of the connection, and, where available, for the association with backup connections, for the deviation of the connection, for the modification of its endpoints, and for resource reservation.

The value of a control plane is inversely proportional to the OPEX it generates: automation of the control plane is of great value since it allows for reducing management. One aspect which is developed in this thesis is the reduction of OPEX by the reduction of the total number of entities which appear in the network. By entities, one must understand the connections established, the nodes, the protection resources, etc. Entities which appear

in the network's control plane must also be monitored by an operations and management department of the network operator: reducing the number of entities is beneficial for the operator as it allows for OPEX reduction. Of course the reduction of the number of entities cannot be separated from the complexity of each entity. The control plane is an enabler for transport network functionalities such as traffic engineering (TE) and SLA-compliance. It is through the control plane that it is possible to establish connections and assign TE characteristics to them, such as QoS constraints. Traffic engineering is also a way to maximize the use of a network, by allowing load sharing, enabling protection scenarios (such as primary and backup connections which do not share "at-risk" links or nodes), etc.

Although there were trials to use CR-LDP [RFC3472], the Resource reSerVation Protocol with Traffic Engineering extensions (RSVP-TE) [RFC3473] is now accepted as the signaling protocol of choice for the GMPLS control plane. The signaling plane is used to establish the LSPs and assign a label to each segment, among other things, such as reserve the resources, plan the protection schemes, etc. As for the routing protocol, GMPLS uses OSPF-TE [RFC3630] for intra-domain routing messages exchange.

1.3.2.3 Management Plane

The *management plane* is the third plane to be considered: this plane is transverse to the two others and allows for management, that is configuration by operators of the two other planes. For example, an operator might want to setup a connection by directly entering its characteristics into network nodes instead of using the control plane (data plane management). Another example is that of an operator who might want to define for a given node the TE policy its control plane must follow (control plane management).

The OAM (Operations, Administration and Maintenance) functions are a part of the management plane, and they are meant to monitor the network's condition. For example, OAM functions allow to test the liveliness of a node in the network (ping operation), check that an LSP is still up, running, and associated to the right Forwarding Equivalence Class (FEC), etc.

The management functions are traditionally divided into 5 main categories, called the FCAPS:

- F: Fault-management,
- C: Configuration,
- A: Accounting,
- P: Performance,

- S: Security.

Fault management's task is to recognize, isolate, correct and log faults that occur in the network. The use of trend analyses is possible to predict, correct or avoid further errors for maximal network availability.

Configuration must gather and store configurations from network devices (locally or remotely), simplify the configuration of the device, track changes which are made to the configuration and statically provision tunnels and LSPs.

Accounting gathers usage statistics from the network users and allows both the billing of clients and the enforcement of usage quotas.

Performance collects usage data in order to monitor the network's condition, adapt its capacities to the clients demands and upgrade the equipments where needed.

Security Management is the process of controlling access to the network's elements, enforcing security policies and maintaining the network free of intruders' attacks.

1.3.3 Multilayer Networks: Peer and Overlay Models

A multilayer network is a network in which two or more networking technologies (and thus, data planes) are superposed. A connection oriented layer, say layer N, is usually used to create a *layout* over layer N-1 to optimize the transport of the traffic coming from layer N+1. The lowest possible layer is the physical layer, which corresponds to the physical interconnections between nodes. In the GMPLS context, the circuits (connections) are the Label Switched Paths (LSPs); thus, at a given layer, a layout is a set of LSPs. The LSPs can be announced as Forwarding Adjacencies (FA) by the routing protocol in an upper layer. The upper layer therefore uses a set of connections provided by its lower layer, this is the *nesting* function, where the lower layer connection, used as an FA by the above layer *nests* an upper layer connection. The traffic demands may then be routed over the uppermost layer. This is the standard multilayer network model, an example of which is presented for the two-layer case in Figure 1.3.

There are several reasons which justify multilayer networks. Historically, different technologies appeared, and operators gradually invested in these, depending on their needs, as each technology satisfied different requirements. For instance, optical switching allows for large bandwidths, but does not allow for fine granularity connections. Packet switched networks have this granularity but do not necessarily provide the same robustness to failures, or the same bandwidths.

When dealing with multilayer networks, two different models are commonly used to describe the way these networks interact. Depending on the type of association between control plane(s) with a multilayer network, one will speak of *peer model* or *overlay model*.

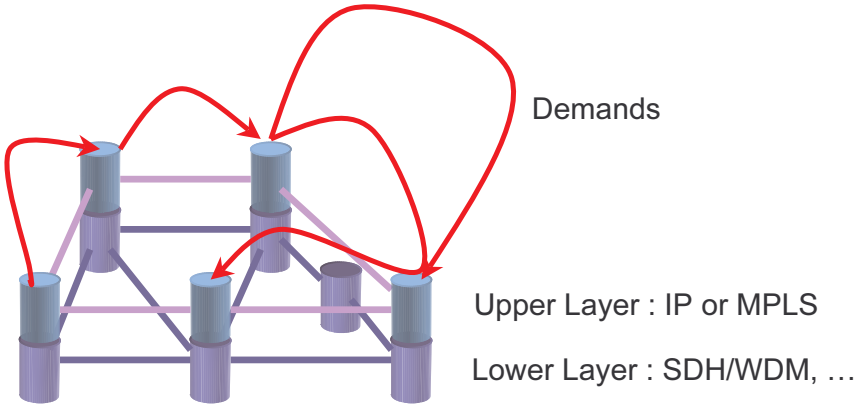


Figure 1.3: Multilayer Network Example

Note that a unified control plane such as the one provided by the GMPLS framework is intended to work in both models.

In the *overlay model*, each layer has its own control plane associated to its data plane. The lower layer only provides connectivity, management and TE information to its upper layer: therefore, the layers are logically separated. The lower layer can decide on which information it will export to the upper layer control plane: this is a client-server model, in which the lower layer is the server answering demands from the client upper layer. For example, an established connection can be associated to its bandwidth, an administrative group, some risk group (*Shared Risk Resource Groups*, SRRG), an end-to-end delay, etc. However, using this model, the upper layer is agnostic to the actual use of the resources and even the topology in the lower layers when these are not exported to the upper layers.

On the contrary, the *peer model* has a single control plane for multiple layers. Associated with this model are the ideas of *unified*, *integrated* or *common* control plane. In the peer model, since there is a single control plane, the upper layers are not independent from the lower layers. The entire information is shared by all layers, and a single connection can span different layers. The route calculation for example, may take into account, at the highest layers, information from the lowest layers, which would not usually be exported. This allows for finer tuning, for instance to perform load balancing at the link level. Indeed, in the overlay model, information concerning the individual physical links is usually not exported from lower layers to upper layers. The main drawback of the peer model compared to the overlay model is its scalability: sharing all network characteristics among all nodes does not scale well. However, since the topology is shared to all nodes, even client nodes, this also poses security issues. The aggregation offered by the overlay model is a way to solve these problems.

1.4 Multilayer Networks Optimization Methodology

Classical optimization problems for network operators are design and resource optimization. These problems are usually based on point-to-point connections (LSPs). The minimum cost multi-objective multi-commodity flow problems resulting from the realistic modeling of the considered networks are known to be NP-complete. This NP-completeness limits the size of the networks that can be treated through exact methods. Available solvers were used to obtain results for the formulated problems for small networks. These results are useful to validate the proposed models, as well as the correctness of the problem formulations. They are also very useful in devising other resolution schemes, since the exact result can be compared to that of the proposed schemes.

To overcome the network size limitation, which quickly makes computation intractable, heuristic methods were developed to approximately solve the described problems. If heuristics use general search techniques, without any knowledge of the nature of the problem, they are meta-heuristics. However, investigation showed that such heuristics were of little use with the considered problems. Heuristics which are adapted to the nature of the particular problem being treated are called Ad-Hoc heuristics. In the problems which we present results for, such heuristics were developed, and their results were compared, on small network instances, to exact solutions, and on larger networks, to other published algorithms.

Part of our work presented here consists in optimizing multi-layer networks. Since we consider only one virtual layer, this consists in choosing two routing sets which will be overall optimal: first, one must route the virtual layout over the physical topology, and second, the demands - i.e., the client traffic - must be routed over this virtual layout. This case is illustrated by Figure 1.3, but may be extended to more than two layers.

Much study has been carried out on this topic. Optimally designing the Virtual Network Topology (the VNT problem) has undergone many studies, in which the optimality criterion has been defined in many ways. It can can be one of the following, a linear combination of a subset of them, or a prioritized selection of them:

- Minimize a delay function (total delay minimized, maximal delay minimized, etc.) [SS03, LGCS02]
- Minimize routing complexity (upper layer number of hops) [XST03]
- Minimize complexity of upper layer (such as the number of LSPs)
- Maximize factor by which the traffic matrix can be increased [YR02]
- Minimize traffic disruption and reconfiguration costs (this implies adding a time dimension). This can be done by simultaneously:

- constraining a new configuration to be within a certain distance of optimality (where "distance" needs defining) [ZLYHG02, SPM01, RR00]
- minimizing the number of operations to go from one configuration to the other [TZJT02, CMP+03, KL01]).

Many approaches are available to (at least partially) solve the VNT problem. It has been proved to be NP-hard [DR00] even for simple ring-topology cases. These problems are far from new and many studies are available in the ATM architecture (in which the layers are the VPs and the VCs), such as in [GS95, FL98, YKH95]. Based on the formal statement of the problem as an optimization problem, such as presented in [PM04], traditional linear optimization techniques may be applied. These include generic network optimization techniques such as Lagrangian relaxation [AMO93], Dantzig-Wolfe decomposition [AMO93] and column generation [AMO93]. However, these techniques have some limitations which arise when precisely modeling networks: scaling issues, such as solving time which grows fast with network size; integer and non-linear-constraints issues which appear when modeling finer constraints, such as delay. This is where heuristic algorithms come into use, they can be divided into two categories: static and dynamic heuristics.

1.4.1 Static Optimization Heuristics

The first set of heuristics, which our bus-LSP virtual layout design algorithm (see Section 5.3.3) belongs to, is the set of static heuristics. These are heuristics which work off-line, and are usually centralized: they necessitate a central computer. This computer will need access to all data entering the optimization process, and it will calculate a layout which will satisfy at best all demands. However, such an algorithm is not able to adapt to changes in the traffic matrix. This implies that the traffic effectively has moderate (or predictable) time evolutions. The solution is calculated in advance, and there will be no way of applying feedback to the solution. This can be used therefore for core networks with high aggregation factors, when the total traffic volumes do not vary much from the average value; or when the evolutions over long intervals of time obey a given periodic variation (for example, hourly variations which are self similar every day).

These static methods focus on the optimization problem while taking advantage of its network characteristics: these are network optimization techniques such as *Branch Exchange* and its variations [LA91, LHA94, YKS+02], *Simulated Annealing* [MBRM96], *Flow Deviation* [MBRM96], *Integer Relaxation* [KS01], *Lagrangian Relaxation* [SL03]. In [ZLYHG02], the authors develop a *Prediction Based Multi-stage Heuristic Reconfiguration* which uses predictions of traffic and a node exchange method. In [TZJT02], the authors compare three heuristics for routing the demands:

- (adding) Longest Path First (LPF) to the VNT

- (adding) Shortest Path First (SPF) to the VNT
- (adding) Minimal Disruptive Path First (MDPF) to the VNT

1.4.2 Dynamic Optimization Heuristics

Dynamic heuristics are used to maintain the network resources at best use, by dynamically adapting the existing layout without recalculating it entirely for every new demand. The algorithm uses the traffic variation as feedback for reconfiguration of the VNT, therefore maintaining a design adapted (within the chosen algorithms capabilities) to the input traffic matrix. When it is possible for each network node to adapt the routing based solely on the information it has at hand, and no centralized intelligence is needed, the algorithm is distributed. This is the case for shortest path routing which can easily be distributed to each network node; however, such cases are rarer with the increasing number of constraints the routing is expected to obey. The advent of centralized routing has led to the development of the PCE architecture [RFC4655], such as defined in the IETF PCE working group.

When a centralized solution is used, the operator must guarantee the distribution of the yielded solution to the nodes, for effective establishment of the solution. For instance, if the operator manually sets up all tunnels, then he will just have to transfer his solution. However, if the algorithm is dynamic and is going to change often, some kind of protocol must be installed to allow the distribution of the configuration to the nodes. When using a distributed algorithm, such a need no longer exists, as each node is only responsible for its configuration. However, the available distributed solutions do not provide results as accurate as centralized ones.

In [OSK+03], the authors study the influence of router ports: this will be one of the major studied criteria in Section 5. In their study, they propose two policies for accepting connection requests. One aims at re-using all established lower layer connections and therefore maximize the number of multi-hop upper-layer connections. The other policy, on the contrary, tries to establish direct single-hop connections in the upper layer (end-to-end lower layer connections). They established that the second policy is better as soon as the number of packet-switching ports in the network is under a given threshold, where the first policy is best in other cases. In [XST03], the Minimum Average Logical Hop (MALH) algorithm searches all possible virtual links and establishes/deletes those in such a way that it minimizes the overall average number of logical hops.

In [GSM02] and [GM02], the authors use periodic measures of traffic for centralized decision making. The setup and deletion of light-paths (upper layer links) is based on thresholds: high and low watermarks, which condition the establishment and deletion of LSPs. In [SOI+03], the authors derive their idea from [GSM02, GM02] to distribute the

algorithm in the nodes.

1.4.3 Dynamic and Static Optimization Heuristics Interworking

Dynamic and static heuristics are, however, often used in conjunction. The dynamic heuristic is indeed often able neither to calculate a first working VNT, nor the initial routing of demands over the VNT. The static heuristic will be used to that effect, the optimization process being done over average long term values for the traffic matrix; and some dynamic, maybe distributed, algorithm will then manage the network layout and routing over time. If the dynamic heuristic diverges too much for the optimal use of the network resources, the process may be reseted by making use of the static algorithm again.

1.5 Thesis organization

The thesis is organized as follows:

In Chapter 2, we present our work on metropolitan Ethernet. This work is not related to GMPLS-controlled networks, but, however, Ethernet is still the technology of choice. We describe the context in which our work is introduced and then present an original formulation of the VPN service on Ethernet spanning trees optimization problem. We then present a greedy heuristic which tries to optimize the layout of these spanning trees and numerically compare its results to concurrent algorithms.

In Chapter 3, we introduce a new object for core networks: the bus-LSP. The bus-LSP is an abstract type of LSP which can be implemented in various data planes, and it induces a complex type of adjacency in the data plane: the bus-FA. We describe the technological context in which this object is introduced, formally define it and present its representation which allows traditional network algorithms to be applied. We also qualitatively evaluate benefits such an object can bring, in terms of OPEX and CAPEX, in single layer networks and multi-layer networks. Though this technology may be applied to GMPLS-controlled Ethernet, its scope is however not limited to Ethernet and is much broader.

In Chapter 4, we evaluate the benefits which bus-LSPs will bring to single layer networks. In this phase, we formulate a first optimization problem, which we solve instances of on two networks using a generic solver. We obtain numerical results demonstrating strong OPEX savings in networks using bus-LSPs.

In Chapter 5, the benefits of introducing bus-LSPs in multi-layer networks are quantitatively evaluated. For this purpose, a network model is presented and exactly solved on a small network. An ad-hoc heuristic is then developed, which is basically a modified and tuned greedy heuristic. We analyze numerical results of our heuristic approach on two large realistic networks, and show strong CAPEX and OPEX savings in multi-layer

networks using bus-LSPs.

In Chapter 6, we tackle the control plane-related issues raised by bus-LSPs. We present how bus-FAs are to be represented in the routing plane. We then show how specific control over bus-LSPs can be accomplished using the standard GMPLS signaling protocol. Indeed, bus-LSPs induce new control needs, which are, as we will demonstrate, not easily fulfilled by standard mechanisms. We therefore introduce a protocol extension which allows for finer and smoother control over bus-LSPs.

Finally, in Chapter 7 we conclude on the interest of the work presented in the thesis, and we discuss some perspectives opened by this work.

Chapter 2

Contribution to Metropolitan Ethernet design

In the metropolitan network segment, Ethernet is already widely deployed and used, in particular to provide clients with the Virtual Private Network service. However, this deployment is based on protocols which are evolving to offer new and improved traffic-engineering functionalities. This chapter explores how the optimal use of Multiple Spanning Trees can improve network scalability and resiliency at the cost of some added complexity.

We first present the Metro Ethernet architecture in Section 2.1 and the VPN services it provides in Section 2.2. Related work to this subject is detailed in Section 2.3. We then formulate the problem of mapping multiple VPNs to various spanning trees as an original Mixed Integer Non Linear Program in Section 2.4. However, this problem is not computationally tractable. Therefore, in Section 2.5, we propose a heuristic algorithm to help the operator first configure his Ethernet switches, that is, build the spanning trees of MSTP, and then map client VPNs to STs in an optimal way for load sharing considerations. In Section 2.6, we provide numerical results for our heuristic approach and compare it to other published algorithms to prove its relevance. Finally, we will present some concluding remarks in Section 2.7.

2.1 Towards Metro Ethernet

2.1.1 Using VLANs to provide VPN services

Ethernet has been a very successful technology and has been widely accepted in local networks. Network operators have been inclined to push this technology further into their network, following a cost-effective desire of unification. Therefore, Ethernet is now becoming the technology of choice when building Metropolitan Area Networks (MANs).

Ethernet MANs will support VPNs (Virtual Private Networks), using the concept of VLANs (Virtual Local Area Networks), such as defined by [IEE03]. VPNs are distributed LANs which are connected using a provider-provisioned network. Once connected, nodes in VPNs are able to communicate as if they were physically directly connected.

VLANs allow a single network to interconnect several groups of Ethernet nodes - groups which, in a VPN context, will be the VPNs - and maintain a separation between those groups, isolating one from another.

2.1.2 Ethernet's control plane

There are many advantages to the Ethernet technology. It has been around long enough and deployed widely enough to benefit from substantial cost reductions. It is now a mature technology which is very cost effective, easy to deploy and easily inter-operable. However it lacks a decent control plane. For the network operator, this means there are no resource reservation schemes available and no load-sharing mechanisms. For the network user, this means there are no QoS (Quality of Service) guarantees and no protection mechanisms for failure resiliency. These are essential drawbacks which need to be addressed when deploying MAN-grade Ethernet. These major issues are therefore a focus of great interest. While the introduction of a GMPLS control plane [RFC3471] for Ethernet [Pap04] is under consideration and would solve many if not all of these issues, networks are already deployed and need solutions now; moreover the GMPLS control plane is not fully automated, therefore operation expense (OPEX) is not negligible.

For the protection issues, the Spanning Tree Protocol [IEE98] (STP) has evolved to a Rapid Spanning Tree Protocol [IEE01] (RSTP). These protocols are in charge of building the Spanning Tree (ST) which will support the Ethernet traffic. The RSTP version accelerates reconvergence of the tree when links or nodes of the current tree fail, bringing the convergence time from 30 seconds to 3 seconds in most failure scenarios. While this is still no protection mechanism, the recovery times are at least more acceptable.

As far as load balancing is concerned, the introduction of the Multiple Spanning Tree Protocol [IEE02] (MSTP) allows for multiple trees to coexist on a single network. Each tree has an identifier and traffic is mapped to various trees so as to allow some sort of traffic engineering. For instance, using multiple trees, it is possible to achieve load balancing in the network by spreading the load across trees which use different links. The work introduced in this chapter considers the issue of optimally mapping VLANs to STs, using MSTP to provide the VPN services.

2.1.3 Requirements for a Metro Access Ethernet

The Metro Ethernet Forum (MEF) has defined so-called Ethernet services for metro transport networks. The MEF, created in June 2001, has become, during the past years, the main forum, widely accepted by the telecom industry, to specify and accelerate worldwide adoption of Carrier-class Ethernet networks and services.

The up to date Ethernet Service Definition framework is currently defined in the MEF 6, "Ethernet Services Definition" and MEF 10, "Ethernet Services Attributes Phase 1". The service framework intends to be very generic in order to serve as a common basis for a wide range of application-based services. It thus defines two major Service types, each begin associated to certain service attributes, which finally ends with some specific Ethernet Service Attribute Parameters. One of the main attributes which is defined by the MEF, and which is not available in legacy Ethernet, is scalability: the provider's network should accept up to the order of 100,000 customers. According to [Mor06], a poll done amongst numerous ISPs revealed that VPN services should support:

- From 1 to thousands of VPNs in the network
- 5 to hundreds or even thousands of VPNs per Provider Edge (PE)
- tens to hundreds of Customer Edges (CEs) per VPN per PE
- hundreds or even thousands of PEs per VPN.

The two major Ethernet Service types are the Ethernet Line (E-Line), corresponding to a point-to-point Ethernet Virtual Connection (EVC) and the Ethernet LAN (E-LAN) corresponding to a multipoint-to-multipoint Ethernet Virtual Connection (EVC), which correspond respectively to VPWS and VPLS defined in Section 1.2. Two main proposals are considered at the MEF for these services: 802.1ad [IEE06b] – "Q-in-Q" – and 802.1ah [IEE06a] – "MAC-in-MAC". Both of these introduce new headers in the Ethernet header, so as to allow the customer's Ethernet and the provider's Ethernet to be separated. [IEE06b] allows for what is called "VLAN stacking", which is having more than one VLAN ID (up to three) per frame: this allows the customer to have his own private VLANs used for internal client switching, and the provider to use another VLAN ID based on the customer, to isolate the customers' traffic sets one from another. However, the service provider is limited to only 4096 VLANs, which is clearly insufficient based on the previous scalability requirements. [IEE06a], which is still at drafting stage, allows for MAC stacking, enabling the service provider to manage up to 16 million customers.

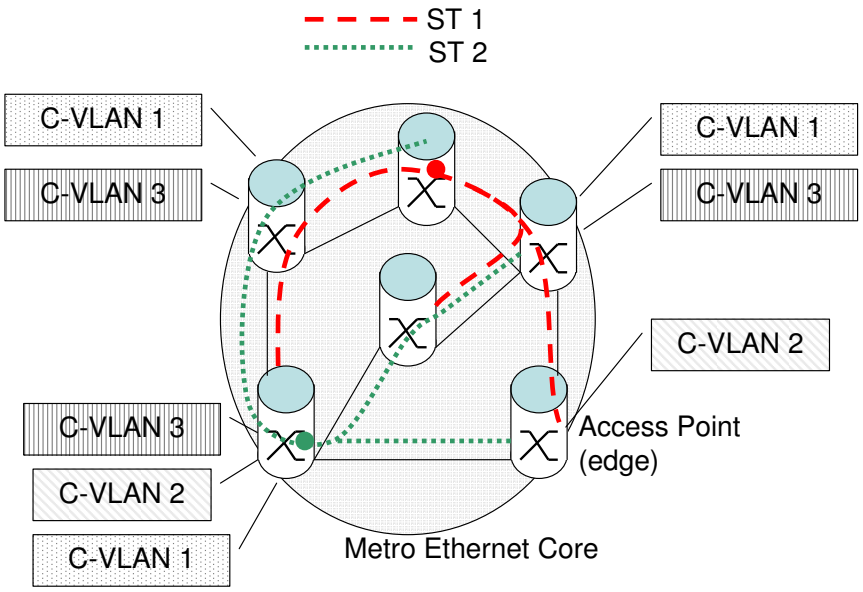


Figure 2.1: Metro Ethernet Architecture

2.2 Metro Ethernet for Ethernet VPNs

The Metro Ethernet architecture (see Figure 2.1) consists of a provider's metropolitan network which relies on Ethernet to transparently connect distant nodes of the network's clients.

A customer may want to interconnect various sites and operate them as a single LAN: he wants to setup a Virtual Private Network (VPN). When this VPN uses a Metro Ethernet provider, it appears as a C-VLAN (Customer VLAN) for the provider. Each of his sites is connected to the provider's network via a gateway: the Provider Edge (PE). The network provider is in charge of transparently interconnecting these PEs according to some Service Level Agreement (SLA).

The notion of VLAN identifier (VID) was introduced for this purpose. The use of VLAN IDs enables the logical separation between groups of Ethernet clients (which would otherwise be impossible due to the flat addressing used in Ethernet). When a frame belonging to a given client enters the provider's network, it is tagged with a VID. Each port of the Ethernet switches in the core is assigned a VID. The switch will forward the tagged frame to and from a port only if it is part of the corresponding C-VLAN. This mechanism allows the different C-VLANs to use the same provider network and yet, C-VLANs are isolated one from another.

Ethernet uses a spanning tree protocol which prevents loops in the forwarding. This protocol constructs a shortest-paths tree given a root and link weights. The root is automatically selected based on an administrator-set identifer for each node. The link weights

are also set by management. Frames are then sent on this tree, which guarantees a single path between any two nodes, and the switches just need to learn which port gives access to which Ethernet destination machine. It is possible to build all possible tree topologies over a given network by properly choosing the root node and link weights.

However, using a tree structure means many links are not used, since they are left out of the structure. This means the network resources are wasted. One way to overcome this issue is to use the Multiple Spanning Tree Protocol (MSTP). This protocol allows, among other things, to have several Spanning Trees (STs) on the same network. Each tree has a single identifier. The C-VLANs are mapped to spanning trees, such that each C-VLAN uses only one of the available STs. However, one ST can carry multiple C-VLANs. The question is to build the STs and have them carry the C-VLANs in a way which is optimal: in Figure 2.1, there are many ways to map C-VLANs 1, 2 and 3 to the two STs. The optimality can be defined in many ways.

One may want to minimize total network resources usage, minimize the maximally loaded link (that is improve load sharing) or have distinct trees (or at least, trees which share the minimum amount of links) for failure resiliency. This chapter focuses on the load-sharing issue; however, other issues such as protection (measured in the number of VPNs threatened by a single link failure) and VLAN ID space (e.g., the number of VLANs needed, which directly impacts OPEX) will also be considered.

2.3 Related work

Many papers have studied how to do traffic engineering using MSTP [ACG05, PNM+05, HZC06]. [ACG05] shows the tradeoff between the number of spanning trees deployed, the alternate routing and the load-balancing. It also offers an algorithm to group various VPN clients onto fewer spanning trees. This grouping strategy however does not show how to initially build the spanning trees which the VPNs are then mapped to. [PNM+05] proposes a heuristic algorithm to build a spanning tree for each VPN. In the end, there are as many trees deployed as there are VPNs in the network: this raises both management and scaling issues. Finally, [HZC06] proposes a control admission scheme to make the best use of deployed spanning trees. This last paper therefore is adapted for optimal dynamic use of spanning trees.

2.4 Formulation as a MINLP

In this section, we will provide a mathematical formulation of the above described problem in the form of a MINLP (Mixed Integer Non Linear Program). Given a physical topology, which STs and which assignment of VPNs to STs make the optimal use of the

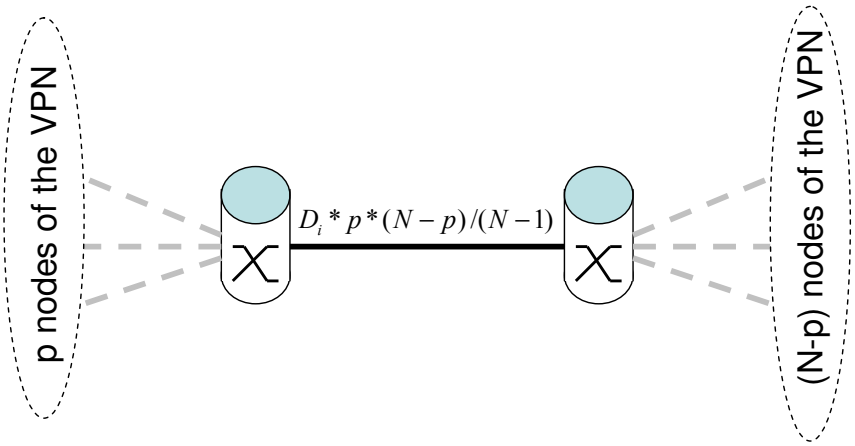


Figure 2.2: Link Bandwidth Calculation

network resources? We first make preliminary observations which are used in the MINLP formulation.

2.4.1 Preliminary observations

For simplicity of the formulation in Section 2.4.2, we will consider here that nodes in a given VPN i are all given the same access bandwidth (D_i). We also assume there is a uniform traffic matrix. This implies that the bandwidth is reserved in such a way that along any tree supporting a given VPN, there must be enough bandwidth to satisfy traffic of volume $D_i/(N_i - 1)$ between all nodes in the VPN, where N_i is the number of nodes belonging to VPN i. However, simple modifications will allow nodes to have differentiated accesses, by adding weights to the nodes.

The calculation of the bandwidth needed on a single link is very simple. Consider two nodes on a tree such as those depicted in Figure 2.2. One node gives access to p nodes of the VPN, the other node to $N_i - p$. Then, for the above bandwidth reservation hypothesis to apply, we need to reserve $D_i * p * (N_i - p)/(N_i - 1)$ units of bandwidth on the link. The key element to understanding the formulation is therefore the calculation of how many nodes are accessed through a given node.

2.4.2 Optimization Problem Formulation

Let us assume that:

- $v = 1, 2, .., n$ indices network nodes
- $e = 1, 2, .., p$ indices links of the network

- $i = 1, 2, .., V$ indices the VPNs we are trying to setup where V is the total number of VPNs to be supported.
- $j = 1, 2, .., S$ indices spanning trees, where S is the maximum amount of Spanning Trees which can be setup in the network.

Additionnally we will assume that the following constants are known by the operator:

- D_i is the bandwidth allocated to each node of VPN i
- P_v^i is the binary constant indicating that node v is a part of VPN i. Note that a node may only be a part of a single VPN: one node must be created for each VPN connected to a given AP (Access Point). This is not a restriction, it is simply for clarity of the formulation. A simple way to ponder some nodes in regards of others is therefore to connect more nodes of the same VPN to a given AP.
- $N_i = \sum_v P_v^i$ is the number of clients of VPN i
- a_{ev} is the binary constant indicating that link e has an extremity at v
- C_e is the capacity of link e
- M_e is the module of link e (integer number of trees which the link can support).

Finally, the optimization problem will manipulate these variables:

- x_{eij} is the flow allocated to link e for VPN i on tree j
- x_{ej} is the total flow allocated to link e on tree j
- x_e is the total flow allocated to link e
- m_{ij} is the binary variable set to 1 if VPN i is mapped to tree j
- ι_{veij} is the integer variable indicating how many nodes of VPN i are available behind node v when coming from link e, using tree j.
- δ_{ejv} is the binary variable indicating that link e is being used in VPN j such that v must go through e to get to the root. If this variable is null, then either the link is not being used in the tree's active topology or it is being used in the other direction.
- u_j is the binary variable indicating that ST j has at least one VPN mapped to it.
- $1/\Lambda$ is the amount by which the bandwidths assigned to each VPN can be multiplied. $1/\Lambda$ should be greater or equal to 1 if the initial bandwidth is to be assigned.

- $1/\Theta$ is the amount by which the number of trees assigned to each VPN can be multiplied. $1/\Theta$ should be greater or equal to 1 if the modules are significant.

We can then easily formulate a Mixed Integer Non-Linear Program with a set of constraints:

- Domain definition constraints:
  - The δ_{ejv} variables are only different from zero for (e, v) couples such that link e is connected to node v

$$\delta_{ejv} \leq a_{ev} \quad \forall j, v, e \tag{2.4.1}$$

  - The ι_{veij} variables are only different from zero for (e, v) couples such that link e is connected to node v. Also, no interface may present an access for a VPN for more nodes than there are clients(this prevents loops to appear, considered as being part of the spanning tree (wrongly) by the program, and containing at least one VPN member mapped to the ST: it forces the graphs generated by the algorithm to be connex, and because of the n-1 active links (see constraint (2.4.8)), a spanning tree)

$$\iota_{veij} \leq N_i * a_{ev} \quad \forall e, i, j, v \tag{2.4.2}$$

- Mapping constraint:
  - Only one ST (Spanning Tree) can be used for a given VPN:

$$\sum_j m_{ij} = 1 \quad \forall i \tag{2.4.3}$$

  - ST use:

$$u_j \geq \frac{\sum_i m_{ij}}{V} \quad \forall j \tag{2.4.4}$$

- Spanning Tree Definition:
  - In a spanning tree, there are exactly (number of nodes - 1) links used (one link going to the root per node, the *uplink*). We use this property to define our spanning tree:

$$\sum_{e,v} \delta_{ejv} = n - 1 \quad \forall j \tag{2.4.5}$$

  - One uplink per non-Root node and per ST:

$$\sum_e \delta_{ejv} \leq 1 \quad \forall v, j \tag{2.4.6}$$

– A given link may only be oriented once at most:

$$\sum_{v} \delta_{ejv} \leq 1 \quad \forall j, e \tag{2.4.7}$$

- We must define, for each node, the number of VPN nodes it gives access to when coming from a given link. These are the interface definitions constraints:

  – For ST nodes which are part of the VPN (and are thus, by hypothesis, leaves of the network graph), the interface is set to 1.

$$\sum_{e/a_{ev}=1} \iota_{veij} = m_{ij} \quad \forall j, i, v/P_v^i = 1 \tag{2.4.8}$$

  – For ST nodes which are not part of the VPN, the interface gives access to the sum of the values of the interfaces its other interfaces are linked to.

$$\iota_{veij} \geq \sum_{\substack{e' \neq e/ \\ a_{e'v}=1}} \sum_{\substack{v' \neq v/ \\ a_{e'v'}=1}} \iota_{v'eij} - (1 - \sum_{v'} \delta_{ejv'}) * N \quad \forall j, i, v/P_v^i = 0, e/a_{ev} = 1 \tag{2.4.9}$$

  where N is the largest VPN count ($= max_i N_i$). The second term insures that the inequality is only applied to links in the active topology (see constraint 2.4.10).

  – For all ST nodes, interface must only be set to a positive number if the link is part of the active topology:

$$\iota_{veij} \leq \sum_{v'} \delta_{ejv'} * N \quad \forall e, i, j, v \tag{2.4.10}$$

- Based on the number of clients of a VPN on accessed behind every node, it is easy to define a set of constraints dictating the flow assignation and related conditions.

  – Flow on links:

$$\sum_{i} x_{eij} \leq x_{ej} \quad \forall e, j \tag{2.4.11}$$

$$\sum_{i} x_{ej} \leq x_{e} \quad \forall e \tag{2.4.12}$$

  – Flow can only be set if the VPN is mapped to a tree, and only if the link is used in the tree's active topology

$$x_{eij} \leq \sum_{v} \delta_{ejv} * M \quad \forall j, e, i \tag{2.4.13}$$

$$x_{eij} \leq m_{ij} * M \quad \forall j, e, i \tag{2.4.14}$$

where M is a constant large enough not to be a constraint.

– Flow assignation: a link must provide enough flow for every node on one side to be able to setup a connection with a node on the other side (this is based on 2.4.1)

$$x_{eij} \geq D_i/(N_i - 1) * \sum_{\substack{v' \neq v/ \\ a_{ev'}=1}} \iota_{v'eij} * (N_i - \sum_{\substack{v' \neq v/ \\ a_{ev'}=1}} \iota_{v'eij}) \quad \forall e, i, j, v/a_{ev} = 1 \quad (2.4.15)$$

– Capacity constraint: this constraint can be used in two ways. First, intuitively, in a resource optimization problem for a given network. Second, by setting all link capacities equal, maximizing $1/\Lambda$ in this constraint will actually be a way to minimize the maximally loaded link, i.e. achieve load balancing.

$$x_e \leq \Lambda.C_e \quad \forall e \quad (2.4.16)$$

– Number of trees per link constraint: this constraint can also be used in two ways. First, intuitively, in a resource optimization problem for a given network, one may want to limit the number of trees on a link based on some discrete resource (such as wavelengths). Second, by setting all link modules equal, maximizing $1/\Theta$ in this constraint will actually be a way to minimize the maximal number of trees on any single link, which means that any link failure will impact a minimal number of trees.

$$\sum_{j,v} \delta_{ejv} \leq \Theta.M_e \quad \forall e \quad (2.4.17)$$

The objective is now:

$$\text{Minimize } \alpha \sum_j u_j + \beta.\Lambda + \gamma.\Theta \quad (2.4.18)$$

Where α, β, γ are the weights assigned according to relative importance of respectively, the cost of the number of STs (management cost), load balancing and protection needs.

Note that constraint (2.4.15) is quadratic, semi definite negative, which means that the solution space is concave. This means it is not possible to solve exactly without exploring the entire solution space. So as to solve this problem, we tried linearizing this constraint (see Appendix A.1), but calculations were still intractable (using the commercial solver CPLEX [CPLEX]), even for instances of the problem implying very small networks. This is why the following heuristic method was conceived and analyzed.

2.5 Greedy Heuristic Mapping of VPNs to Spanning Trees

We will present here an original heuristic algorithm for mapping VPNs to STs: the *Diversified Forest with Greedy Mapping*. This method is based on two stages. First, we build a set of STs which aim at having few links in common: the Spanning Forest. Second, we assign VPNs to STs in a greedy fashion.

The rationale behind this process is that this way, the network operator only builds a Spanning Forest once and then maps his various VPNs to it. This method is therefore very convenient for operators, while providing good results (see Section 2.6).

2.5.1 Building the Spanning Forest

For this stage of the algorithm, the weights of each link will be changed so as to represent how often the link appears in spanning trees. After building an ST, the weights of its links are increased of a fixed value Δ (influence of this *feedback* parameter is studied in Section 2.6). Initially, all links have a same fixed weight of 1.

When building the spanning trees, an important factor is the choice of the root. The root is typically the node of the tree which will have the most traffic running along it. Therefore, it is important that the roots be as varied as possible, which is why we introduce a variable associated to each node which counts how many times a node has been a root in any ST. The root is then chosen as the network node which minimizes the quantity (with W the weight and C the capacity):

$$\frac{\sum_{\text{all adjacent links}} W_{link}}{\sum_{\text{all adjacent links}} C_{link}} * (\text{number of times root}) \tag{2.5.1}$$

Once the root is chosen, the ST is built by using a method such as Dijkstra's shortest path [Dij59]. The weights are then updated and the process is repeated a fixed number of times S.

Algorithm 1: Spanning Forest Generation

Given: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
Given: C_l the capacity of link $l, \forall l \in \mathcal{A}$
Given: W_l the weight of link $l, \forall l \in \mathcal{A}$
Find: A diversified Spanning Forest

1: Set $W_l = 1 \quad \forall l \in \mathcal{A}$
2: **for all** $i \in [1, ..., S]$ **do**
3: Select a root minimizing the quantity of (2.5.1)
4: Build a spanning tree ST_i rooted at the selected node
5: $W_l \leftarrow W_l + \Delta, \forall l \in \mathcal{S}T_i$
6: **end for**

2.5.2 Mapping the VPNs on the Spanning Forest

Once the spanning forest has been generated, the VPNs have to be mapped onto it. For this part, a simple "greedy" method is applied.

Algorithm 2: Mapping VPNs to Spanning Forest

Given: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
Given: C_l the capacity of link $l, \forall l \in \mathcal{A}$
Given: A Spanning Forest: $\{ST_1, ST_2, ..., ST_S\}$
Find: A mapping of VPNs to STs

```
1: Sort VPNs decreasingly according to the quantity D_i * N_i
2: for all VPN do
3:     Select ST which minimizes maximally loaded link when mapping the current
       VPN on it.
4:     if More than one ST are candidates then
5:         Select ST which minimizes total bandwidth added to network when
           mapping the current VPN on it.
6:     end if
7:     Map VPN to selected ST (update link loads)
8: end for
```

The VPNs are sorted by their volume (ie: by decreasing order of the quantity $D_i * N_i$ using the same notations as in section 2.4).

Next, the largest VPN is mapped onto the ST of the Forest which minimizes the load of the maximally loaded link. If more than one ST have the same minimum value for the maximally loaded link, the ST which satisfies the VPN while using the total minimum amount of bandwidth in the network is selected.

2.6 Performance Analysis

In this section, we applied the heuristic method described in the previous section to three different networks. The first one is a 11 node- and 14 (bidirectional) link-network, based on the French research network VTHD [VTHD], illustrated in Figure 2.3. The second network is a 21 node- and 36 link-network, based on the Italian high-speed network [SIL+98] illustrated in Figure 2.4 . The third network is a highly connected 12 node- and 40 link-network, such as the one described in [PNM+05]. We considered the first two core networks all along this thesis (except in Chapter 4 due to computational issues) as they are good large scale, realistic networks. Their topology can be assumed close to one of a small Metro Ethernet network. The last network was chosen due to its large size, which makes it more realistic as a Metro Ethernet topology. However, the last network was also chosen to provide a comparable study frame with the algorithms described in [PNM+05], as their algorithms were designed to perform well on such a network.

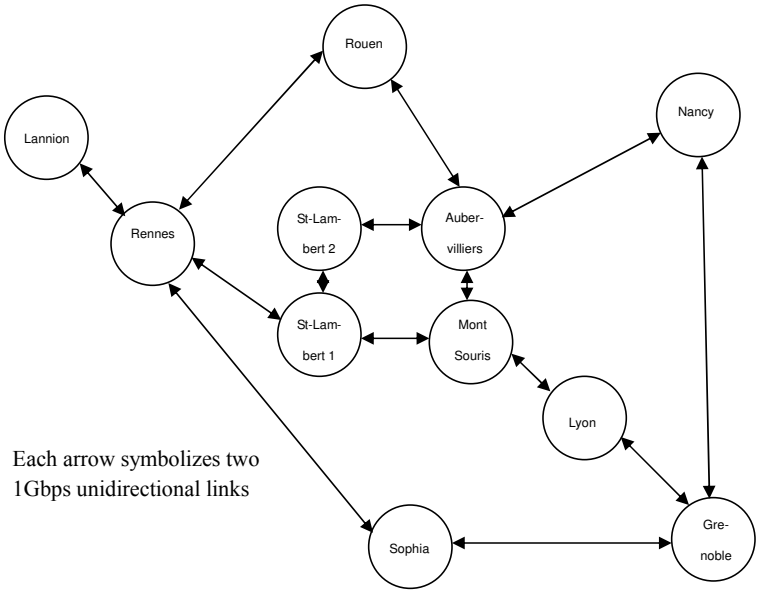


Figure 2.3: VTHD Network

Table 2.1: Relative Algorithm Performances : VTHD Network

		VTHD Network				
	Algorithm →	Single ST	MST without traffic update	MST with traffic update	Enhanced MST	**Diversified Forest w/ Greedy Mapping**
30 VPNs	**Max.load**	1678	1084	930	862	**794**
	Avg.load	664	672	676	**634**	682
	Max.nb of trees/link	1	30	30	30	**10.4**
	Total nb of trees	1	30	30	30	**10.4**
10 VPNs	**Max.load**	535	390	354	349	**270**
	Avg.load	212	212	215	**199**	214
	Max.nb of trees/link	1	10	10	10	**6.72**
	Total nb of trees	1	10	10	10	**6.72**

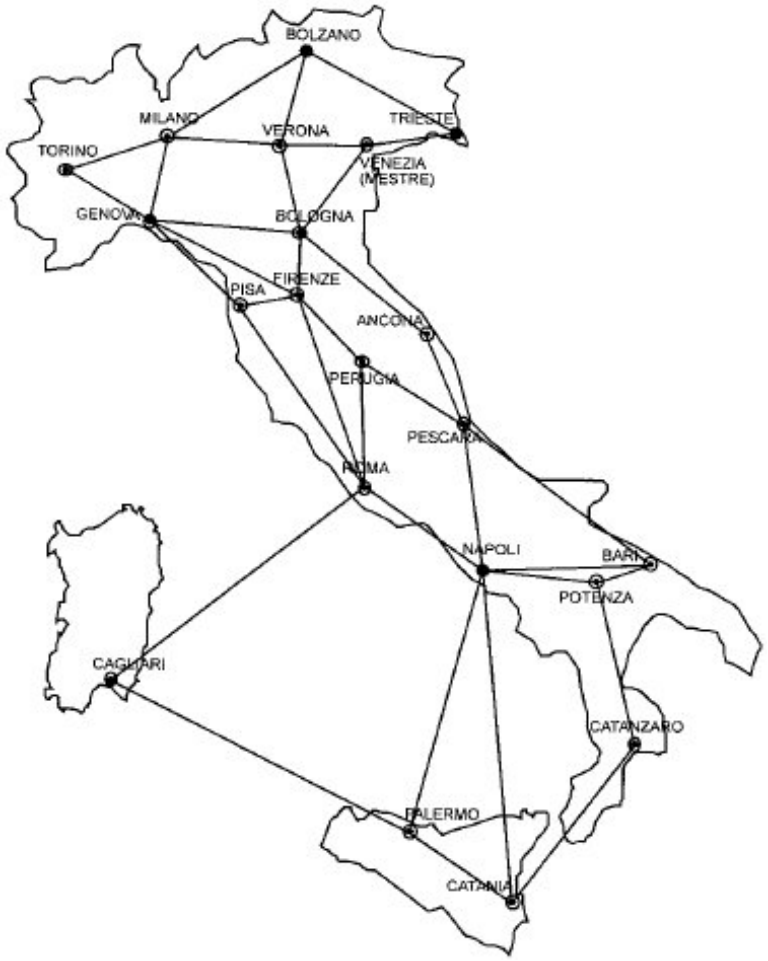


Figure 2.4: Italian High-Speed Network

We compare the results of our heuristic to those provided by the 3 algorithms described in [PNM+05]. The first one constructs one ST associated to each VPN. This leads to as many STs as there are VPNs. The idea is to apply weights to links which increase with the load supported, and construct trees based on these weights. This technique is referred to as "MST with traffic update". They further enhance their tree by mapping some links of some shortest paths to the constructed trees: this will be referred to as the "enhanced MST" technique. When the link weights are not updated after constructing a tree, that is, each ST is constructed for a VPN by simply selecting a root randomly among the VPN's AP nodes, the technique is referred to as "MST without traffic update". We compared the results of our heuristic on the 3 above described networks with those provided by the "MST with traffic update", the "MST without traffic update", the "enhanced MST" and the single spanning tree strategy, by implementing all algorithms[1].

[1]The source code for Python implementations as well as auxiliary programs are freely available at http://perso.enst.fr/~brehon/

Table 2.2: Relative Algorithm Performances : Italian Network

		Italian Network				
	Algorithm →	Single ST	MST without traffic update	MST with traffic update	Enhanced MST	**Diversified Forest w/ Greedy Mapping**
30 VPNs	**Max.load**	5915	3658	2807	2872	**2252**
	Avg.load	1504	1257	1288	**1131**	1255
	Max.nb of trees/link	1	29.0	27.4	27.1	**12.7**
	Total nb of trees	1	30	30	30	**13.8**
10 VPNs	**Max.load**	1826	1267	1055	1221	**785**
	Avg.load	467	387	396	**341**	379
	Max.nb of trees/link	1	9.93	9.55	9.82	**7.19**
	Total nb of trees	1	10	10	10	**7.29**

For simplicity, networks have links which are all of equal capacity. We generate a fixed number of VPNs, having each a random set of APs in the network. Each VPN asks for a random amount of bandwidth, uniformly chosen betwen 0 and 10 MBps. This amount must be available between any 2 nodes of the VPN. For our heuristic, we chose to generate as many STs as there are nodes, in the Spanning Forest Generation phase. Δ was set to 1 in this chart. The results presented here are average values of 100 simulations. The results are presented in tables 2.1, 2.2, 2.3 and give the statistics for maximum load, average load, maximum number of trees for the links and the total number of trees setup in the network.

As the results show, our algorithm outperforms the other 4 as far as load balancing is concerned. For the VTHD network, the maximum load on a single link is 7.9% lower on average than the best concurrent algorithm (which is here the "enhanced MST"). For the Italian network, the improvement is of 19.8% over the "MST with traffic update". We can also notice that for this large network which is not highly connected, the "enhanced MST" algorithm does not improve on the "MST with traffic update". Finally, for the highly connected network, our algorithm improves on the "enhanced MST" algorithm by 25%.

The average link load is higher with our approach (by 3 to 9%) than the best algorithm in this regard, which is always the "enhanced MST". Considering how that algorithm builds

Table 2.3: Relative Algorithm Performances : Dense Network

		Dense Network				
	Algorithm →	Single ST	MST without traffic update	MST with traffic update	Enhanced MST	**Diversified Forest w/ Greedy Mapping**
30 VPNs	**Max.load**	1581	650	403	382	**286**
	Avg.load	230	216	215	**208**	215
	Max.nb of trees/link	1	18.3	13.9	14	**5.0**
	Total nb of trees	1	30	30	30	**11.5**
10 VPNs	**Max.load**	522	280	202	219	**139**
	Avg.load	74	70	70	**66**	68
	Max.nb of trees/link	1	6.89	5.56	5.73	**3.36**
	Total nb of trees	1	10	10	10	**6.71**

the STs, this is however expected behavior, since it aims at minimizing an overall cost of the tree. In contrast the *Diversified Forest with Greedy Mapping* aims at achieving load-balancing: this comes at the cost of using less direct paths, impacting the average link load. The network operator will therefore have to ponder whether increasing the average load a little is worth the tradeoff to reduce the highest link utilization.

When there are 30 VPNs (respectively 10), the "enhanced MST" and "MST with/without traffic updates" all use 30 (resp. 10) trees. The single ST uses a single tree to support all VPNs. However, our algorithm makes use of only 30% to 50% that number of trees. This does not have any meaning regarding traffic distribution, but it does mean a considerable label space savings (less VLAN-IDs). It also means there will be less signaling-related traffic, since there are fewer trees to maintain.

In tables 2.4, 2.5, 2.6, the influence of the parameter Δ is analyzed. This parameter, which is defined in Section 2.5.1, corresponds to the value of the feedback which is brought to the link weight after it has been incorporated into a ST. The conclusion which may be drawn is that there is no optimal value for the parameter, and that this is network-dependent optimal. Here, for the purpose of load balancing, it is better to have high feedback for the Italian network and low feedback for the dense network (any feedback is good for

Table 2.4: Influence of Δ for the VTHD Network

		VTHD Network			
	value of $\Delta \rightarrow$	0	0.4	1	5
30 VPNs	**Max.load**	794	**783**	**783**	**783**
	Avg.load	686	**682**	**682**	**682**
	Max.nb of trees	**9.36**	10.43	10.43	10.43
	Avg.nb of trees	**9.36**	10.43	10.43	10.43
10 VPNs	**Max.load**	274	**270**	**270**	**270**
	Avg.load	**213**	214	214	214
	Max.nb of trees/link	**6.32**	6.72	6.72	6.72
	Total nb of trees	**6.32**	6.72	6.72	6.72

Table 2.5: Influence of Δ for the Italian Network

		Italian Network			
	value of $\Delta \rightarrow$	0	0.4	1	5
30 VPNs	**Max.load**	2264	2280	2251	**2218**
	Avg.load	**1216**	1253	1255	1252
	Max.nb of trees	**10.24**	12.08	12.72	12.48
	Avg.nb of trees	**10.26**	13.31	13.81	13.92
10 VPNs	**Max.load**	776	788	785	**775**
	Avg.load	**372**	380	379	378
	Max.nb of trees/link	**6.32**	7.19	7.26	7.47
	Total nb of trees	**6.32**	7.55	7.29	7.38

VTHD). For overall load, it is better to have *no* feedback however, in all cases except VTHD with few VPNs. As for the maximal number of trees per link and the total number of trees in the network, it is often better to have *no* feedback. These conclusions have been observed regularly on most of the averaged experiments. The value of the parameter Δ must therefore be adapted to the underlying network for optimal results, even though setting the value to 1 proves to be on average a good choice.

Table 2.6: Influence of Δ for the Dense Network

		Dense Network			
	value of Δ →	0	0.4	1	5
30 VPNs	**Max.load**	317	**280**	286	302
	Avg.load	**211**	217	216	221
	Max.nb of trees	6.03	**4.8**	4.95	4.94
	Avg.nb of trees	**10.6**	11.4	11.5	11.4
10 VPNs	**Max.load**	144	140	**139**	149
	Avg.load	**66.7**	67.5	67.5	69.2
	Max.nb of trees/link	3.54	3.36	3.5	**3.15**
	Total nb of trees	**6.16**	6.36	6.71	6.69

2.7 Concluding Remarks

In this chapter, we tackled the problem of mapping VPNs to Spanning Trees in Metro Ethernet networks. We formulated this problem as a novel MINLP problem. However, due to tractability issues, we were brought to conceive a heuristic algorithm to solve the problem of optimally balancing the load of the VPNs accross the network. This algorithm is quite simple to implement by an operator, since the two stages of building the Spanning Forest and then mapping the VPNs are totally independent. We compared the results of our heuristic to the other known methods of mapping VPNs to Spanning Trees which are available, and very good results were obtained.

This research could, provided proper adaptation, be used to determine the best *multicast* distribution trees, in the context of the current trend of multicast research, both for VPN services or content-distribution (such as IP television). Further research on the subject is targeting the resiliency of the generated forests, to minimize the impact of a single failure.

For Carrier Ethernet services, providing load-balanced Ethernet transport of client Ethernet is not sufficient, as other requirements are stated at the MEF for "Carrier Ethernet", such as:

- Protection: this is a traditional transport requirement, which translates into the well known "five 9's" requirement, or 99.999% of network availability
- Quality of Service (QoS) guarantees, such as hard bandwidth, delay, jitter, etc. guarantees
- Service management, which allows for simple way of enabling, managing, measuring and upgrading services (refer to Section 1.3.2.3).

All these additional requirements necessitate advanced networking functions, such as the ones enabled by the GMPLS framework: transport of Ethernet in GMPLS-controlled networks will benefit from the bus-LSP structure presented in the following chapters.

Chapter 3

Bus-LSPs and Bus-FAs

In present GMPLS networks, the layouts are based on point to point Label Switched Paths (LSPs). In this chapter, we introduce an unexplored type of LSP (and more precisely of Traffic-Engineered link), based on a unidirectional bus concept, allowing reducing the cost of the network as well as the layout complexity. The benefits of the concept are qualitatively evaluated. The graph representation of this concept is also introduced, so as to allow the application of traditional graph-based studies to networks implementing this new concept.

3.1 Technological context

3.1.1 Opaque optical network architecture

An opaque optical network consists of multiple equipments as described in Figure 3.1, which we will call multilayer cross-connects (MLXC), interconnected by multiple optical fibers. Use of (D)WDM (Dense Wavelength Division Multiplexing) allows for multiplexing more than one wavelength on a given fiber, and hence, it is possible to use the great potential of a fiber's bandwidth. Indeed, while no current electronic device can deliver data at the rate which utilizes a fiber efficiently, by multiplexing wavelengths with reduced bandwidth, it is possible for electronics to use the wavelengths efficiently.

The MLXC has two main functions: a routing function, and a switching (or forwarding) function. The routing function and the switching function may belong to a same piece of equipment, or they may be physically separated and need interfaces to be connected. The routing function is used at the packet level, to determine where to send each packet based on its header. This is layer 3 in the standard OSI model [Zim80]. The switching function is used to forward all data coming from a given direction to a same destination direction. This is layer 2 in the standard OSI model [Zim80]. This switching function may

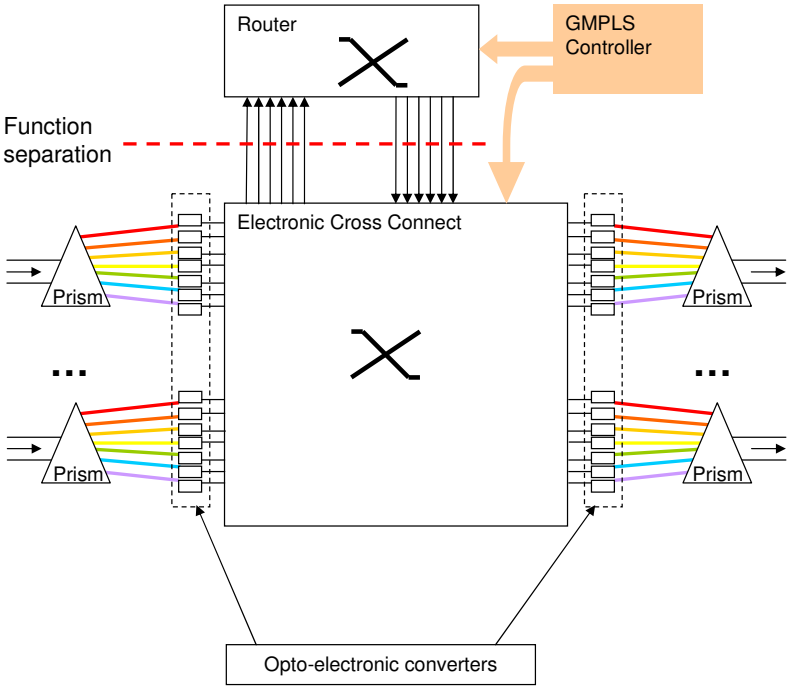


Figure 3.1: The MLXC architecture

be dynamically changed so as to modify the switching patterns, and therefore, virtual layouts.

The routing function may be realized by standard OSPF (or IS-IS), in which case every packet is routed based on its destination field using hierarchical routing tables [RFC2328]. It may also implement a label-based packet switching technique such as MPLS [RFC3031], or MPLS-TE [RFC2702], which are often placed at a layer 2.5 in the OSI model.

The MLXC consists in various elements. On the path of transit packets, the first element met is a transceiver at the end of a fiber optic which demultiplexes the various wavelengths and sends each one of them to an opto-electronic converter. These converters are attached to input ports of the electronic cross connect, and so are the output ports of the router. The output ports of the cross-connect are themselves attached, one to one, either to input ports of the router or to outgoing converters. Outgoing converters are multiplexed onto a single fiber using a transceiver again (basically, a prism). This means that a packet going through the MLXC and which is routed at layer 3 will be switched through the electronic cross-connect twice: once to go from an incoming converter to the router, and once from the router to the appropriate outgoing converter.

A network consisting of such equipments is called opaque in opposition to transparent optical networks. In transparent optical networks, wavelengths are switched directly at the optical level, without use for opto-electronic and electro-optical conversions. In opaque

networks this is not the case. Although transparent optical networks have been announced for a while, they still are not deployed, and opaque networks prevail in operator networks. Therefore the circuits which are established are not light paths but instead circuits of a transport technology. These technologies can be *layer 2 LSPs* [Pap04] (Ethernet circuits), SDH, etc.

A very complex subproblem [NTLM02] is often linked to the multilayer routing problem described in Section 1.4. When applied to transparent optical networks, a wavelength cannot be changed along a path, unless expensive devices called *Wavelength Converters* are used. Since the optical bandwidth space is limited, this means that wavelength availability can become quite fast an important constraint. This yields a subproblem called the wavelength assignation problem which when unfeasible for a given VNT may cause the only possible solutions to be suboptimal. When considering opaque (or equivalently, transparent with unlimited wavelength converters) optical networks, this problem does not arise.

3.1.2 GMPLS Unified Control Plane and MLXCs

The unified multilayer control plane offered by the GMPLS architecture has been designed so as to take advantage of such networks, and is intended to pilot equipments such as the MLXC.

In the exposed network architecture, two types of FA-LSPs may coexist: layer 2 FA-LSPs and layer 3 FA-LSPs:

- At layer 2, an FA-LSP which only transits through a node will be switched directly from an incoming port on the electronic cross-connect to an outgoing port. An FA-LSP originating in an MLXC will have its layer 3 packets routed towards an ingoing port of the cross-connect. The cross-connect has been configured so as to switch this port to a given outgoing port by the GMPLS control plane. Finally, an FA-LSP terminating in an MLXC will have incoming packets, once out of the opto-electronic converters, switched to an outgoing port going to an incoming port of the router.

- At layer 3, an FA-LSP is simply a set of successive layer 2 FA-LSPs which are concatenated to form a longer fixed path for packets. At the merger of each of these layer 2 FA-LSPs, the packets must be forwarded at the router level of the joining MLXC.

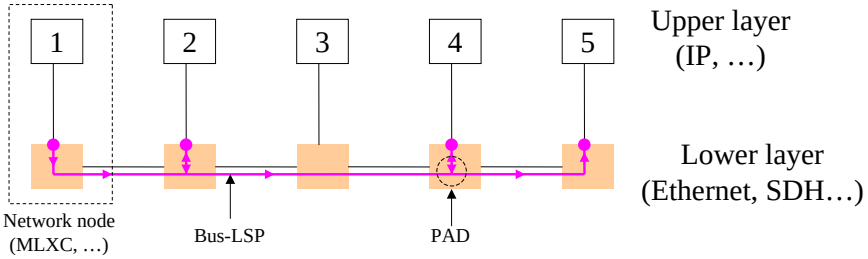


Figure 3.2: Bus-LSP Terminology

3.2 Bus-LSPs

3.2.1 Definition

In a GMPLS-controlled multilayer network, a Label Switched Path (LSP) is a connection between ingress and egress LSRs (Label Switching Routers), spanning one or more links. The links can be physical links such as fibers, or lower-layer LSPs nesting the LSP, as long as the hierarchy defined in [RFC4206] is respected. A Forwarding Adjacency (FA) is a representation, at the Data Plane, of the connectivity that one or several LSP offer in the data plane between the ingress and egress. It can therefore be used in upper layers as a physical link is used by the bottom layer.

What we call here *bus-LSP* and *bus-FA* are extensions of these concepts of LSP and FA. In a bus-LSP, such as the one depicted in Figure 3.2, intermediate LSRs in the LSP have the possibility of adding and dropping data on the bus-LSP. We will call these nodes add-drop points. A bus-LSP provides a forwarding connectivity between ingress, add-drop points and egress in the data plane. We call the representation of this connectivity in the Data Plane a bus-FA. A mechanism allows the ingress of the bus-FA and any add point to specify which single node on the bus-FA the data sent is addressed to. This solution is neither a multicast one nor a point-to-(multi)point one. At the moment, the bus-LSP supporting the bus-FA is unidirectional. Hence, data may only be sent to a drop point (or the egress) which is downstream from the sending add point (or ingress). When traffic arrives in a node which must inject it onto a bus-LSP, the node checks where the traffic must exit, based on the Forwarding Equivalency Class the packet belongs to [RFC3031] as for any (G)MPLS packet, and addresses the specific drop point (or egress node) it is intended for. Note that bus-LSPs can nest other bus-LSPs at various layers, just as LSPs can nest LSPs.

This structure is only one step away from offering some degraded point-to-multipoint functionality, by allowing packets to be both dropped and forwarded in the same MLXC. This needs further signaling and routing protocols research, but it could be useful for VPN or content distribution, as will be shown in Section 3.2.4.3.

3.2.2 The Packet Add-Drop (PAD)

The equipment which is considered at the moment consists in a PAD (Packet Add-Drop) which is a relatively light piece of software or hardware added in the electronic cross-connect part of the MLXC. The PAD is able to read a small integer in the incoming packets on-the-fly, and:

- Either decrement it if it is strictly positive and forward the packet to the cross-connect for regular FA switching,
- or forward the packet to the router if it is null (possibly bypassing the cross-connect).

This is the Drop functionality of the PAD. The integer therefore serves as a local addressing along the bus-LSP, and any equipment sending packets on the bus can set this integer to a given value which corresponds to the number of PADs the packet is supposed to go through before being dropped. The PAD also has an Add functionality to it, which allows an intermediate router to inject traffic on a bus. This relies on a MAC protocol, of which the details are not disclosed here.

Some details of this PAD structure are unknown, and will be decided upon full implementation. Some of these specific details are:

- The number of incoming electronic ports which will be mapped to a PAD function.
- The PAD position in the MLXC architecture:
  - at input of the electronic cross-connect : this implies the router must add the packets to the LSP which is then switched by the cross connect
  - at output : it means uselessly switching packets through the electronic fabric before dropping them
  - partly input and partly output, dropping packets before going through the cross-connect and adding packets after. This is obviously the most expensive solution as the PAD is split in two parts, but it is the most efficient solution as the electronic fabric use is minimized.
- The number of interfaces going from PADs to the routing function. Given that on one bus-LSP, incoming packets in a transit MLXC will not all be dropped, it would appear that having all PADs have their own outgoing port on the cross-connect and their own input port on the router is inefficient. Therefore, all PADs could share one or several ports.

In the case where bus-LSPs are nested in other bus-LSPs, there must be a PAD system for every layer using bus-LSPs. Each layer has its own counter system, and these cannot interact, in the same way as GMPLS labels do not interact.

3.2.3 Related work

3.2.3.1 Light trails

Light-trails [GC03] are a neighbor concept to bus-LSPs. A light trail is setup as a path in the network. A dedicated control channel is then in charge of setting up connections between nodes belonging to the trail. While these two nodes are conversing, no other node may use the trail.

Bus-LSPs are however very different from light trails. In a light-trail architecture, a dedicated control channel must give clearance before data is sent on trail. Therefore, the specific control channel requires resources for the exchange of the control messages it uses. Also, data must be aggregated before sending, for control channel usage sparing, and for resource utilization maximization. Indeed, if the data is not aggregated, the control channel spends its time swapping which node may use the data channel(s), and therefore, data channel resources are wasted. Finally, the shared optical bus cannot support two overlapping connections, since, at any moment, each segment of the trail is dedicated to a given connection. With the bus-LSP concept, the shared bus is accessible at all time for all nodes. This means two small overlapping connections may simultaneously use the bus.

Yet, studies can be easily applied from light-trails to bus-LSPs and conversely. Some studies [BSK05, FHS04, BHS05] have focused on the design of virtual topologies using light-trails. [FHS04, BHS05] focus on the optical CAPEX optimization (namely, number of transceivers) in the case of all-optical transparent networks. [BSK05] concentrates on reducing the number of connection grooming nodes (*hub* nodes).

3.2.3.2 Distributed Aggregation

In [BPD+04], the authors use the possibility of adding traffic onto an LSP at an intermediate node in an all-optical transparent network to improve throughput and reduce network cost. This is the equivalent of a bus-LSP on which only add nodes are available, but in a transparent optical network. However, we show that in a routed environment, the throughput increase and cost reduction can be achieved by using the nesting technique, at the cost of router processing load.

3.2.4 Benefits of the Bus-LSPs

By offering more adjacencies with fewer LSPs, the bus-LSP concept allows reducing the number of LSPs to be setup and therefore the operation and maintenance costs, regardless of the environment it is deployed in.

However, this is not the only benefit of bus-LSPs, as shown below. We analyze two cases. In the first one we consider a network of just two layers, the physical network layer and

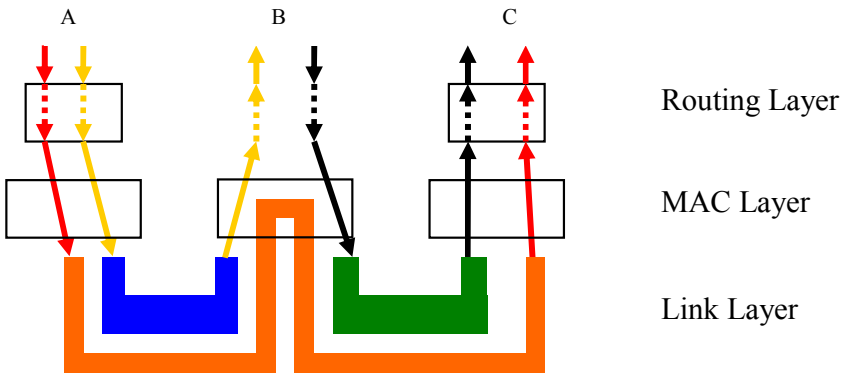


Figure 3.3: Direct Point to Point Architecture

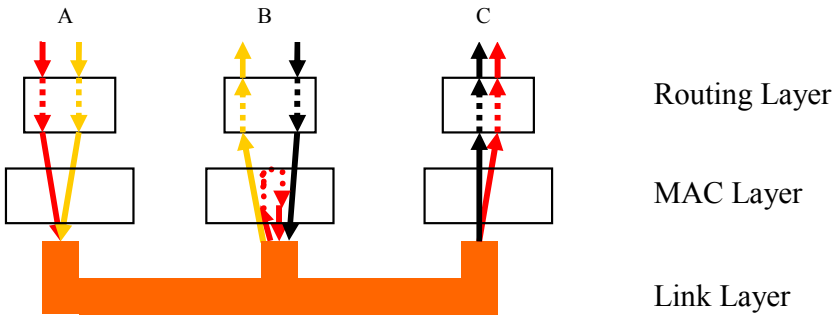


Figure 3.4: Bus-LSP Architecture

a second one allowing the deployment of a layout that will support directly the traffic. There is no routing over the layout allowing the transport of traffic over more than one FA (we call the corresponding network architecture "non-routed architecture"). This requires the establishment of connections between all pair of nodes which have a demand to satisfy from one to the other, and can lead to a full-mesh of connections. In the second case, there is a third layer that can route the traffic over several FAs (there is no need here for a full mesh). We will also consider the additional services the bus-LSPs facilitate to provide.

3.2.4.1 Benefits of the Bus-LSP in Non-Routed Architectures

In a direct point to point architecture (see Figure 3.3), each traffic demand between two nodes is transported over a single point to point FA between the nodes (no bus-FAs are used). We will call this architecture DP2P (direct point-to-point). In the following we consider that the FAs are supported by lambda connections.

As an example, consider a physical topology composed of three nodes A,B and C which are connected by two links, one from A to B and the other from B to C (as shown in figures 3.3 and 3.4). When comparing the DP2P solution with the bus-LSP one represented

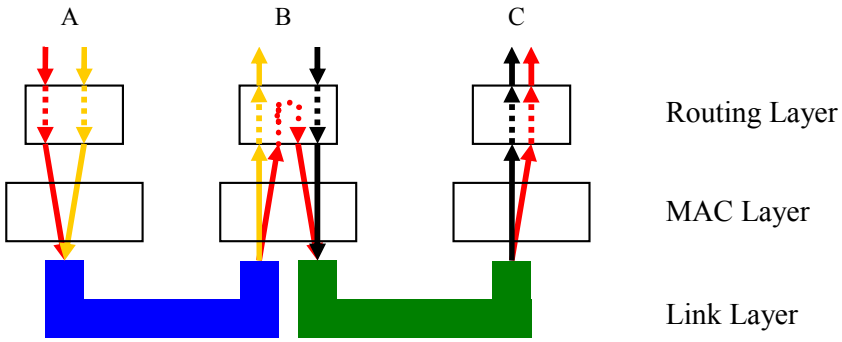


Figure 3.5: Multi-Hopped Architecture

in Figure 3.4, we observe that:

1. The DP2P approach requires maintaining two LSPs on each link, instead of only one for the bus-LSP (see Figure 3.4).

2. Because of the discrete bandwidth of lambda connections, no statistical multiplexing is allowed in the DP2P case between the A to B and A to C traffic, neither between the A to C and B to C traffic. Therefore, more resources are usually consumed. This idea has been partially explored in [BPD+04] using a more restrictive, yet related to the bus-LSP, technology.

3. Twice as many wavelengths and total opto-electronic converters are required.

The DP2P approach is therefore more resource-consuming and implies a greater management complexity since the number of required LSPs is larger. The gain increases with the size of the network.

3.2.4.2 Benefits of the Bus-LSP in Routed Architectures

In a multilayer environment, it is possible to route demands over multiple FAs or bus-FAs. We compare here a bus-LSP based architecture to a situation (see Figure 3.5) where all traffic demands are supported by LSPs which are setup only between physically adjacent nodes that can switch traffic. We will call this situation MH (multi-hopped).

As an example, consider the same topology as in the previous section. When comparing the MH solution with the bus-LSP one, we observe that:

1. Where the MH architecture has to setup and maintain two or more LSPs, only one bus-LSP is needed.

2. The bus-LSP structure is less demanding in terms of router resources than the MH structure; indeed, transit packets in a node are not routed at layer 3 but just forwarded at layer 2. This can be done very easily if layer 2 frames are tagged with a mark indicating the corresponding dropping node. This means lower load for the router, and processing power is spared, which can directly be converted to cost reduction for the operator.

3. If the layer 3 routing function is provided by a separate equipment, the bus-LSP will save resources at the interface between layer 3 and lower layers equipment, this can even reduce the number of required physical interfaces in large networks.

4. Finally, the bus-LSP and MH architectures are strictly equivalent in terms of statistical multiplexing since both can multiplex multiple flows onto a single LSP.

3.2.4.3 Benefits of the Bus-LSP as a Service Enabler

This section will focus on the functional interest a tool such as the bus-LSP can provide. From the section presented beforehand, its interest can be understood on a pure optimization level. Bus-LSPs basically reduce the load on the upper-layer router, save some opto-electronic converters and provide management facilities. With this in hand, it is natural to consider bus-LSPs as a tool to reduce costs and optimize the network design. However, they can also play an interesting part for providers wishing to furnish given services.

First of all, due to the particular design of the bus-LSP, flows inside it share the bandwidth. If a provider is using it to send traffic, dimensioning nested LSPs is important. However, a bus-FA could be used to provide a Layer 2 VPN service to customers. By running a bus-FA through all CEs (Customer Edges), which is dedicated to the customer, a bus-FA is a very interesting way of offering layer-2 connectivity. A customer may use the bandwidth reserved as he wishes, and the bandwidth sharing between CEs is not to be decided by the provider. Additionally, if bus-FAs were enhanced by a multipoint functionality, there would be some additional gain by natively offering broadcast instead of relying on frame duplication at CEs. Finally, if a bus-LSP is allowed to be circular, any CE can broadcast or target a message on the Layer-2 VPN. In this case, one unidirectional ring-shaped bus-LSP would replace the two unidirectional bus-LSPs needed to provide the proper connectivity. In the VPN perspective, obviously, a bus-FA is used not as an optimization technique, but as a functionality-serving architecture. This service remains to be detailed (particularly the CEs behavior, how to establish the bus/cycle running between CEs, etc.).

Second, if bus-FAs are given point-to-multipoint capabilities, then they can be used as a simple way of doing content delivery (among other services which remain to be defined). A central server could send its content via a bus-FA to distributed servers. This can be

useful for television and video applications for instance, and more generally, for all one-way content distribution services. Using a bus-FA for two-way content application can also be considered, either by using two parallel bus-LSPs or by a single circular bus-LSP, since this is basically the establishment of a temporary VPN session.

3.3 Representation in multi-layer networks

It is important to be able to represent the connectivity a bus-FA offers in a graph. This can be useful for example, once a layout containing bus-LSPs has been decided on, so as to optimize the above layer. This does not apply when optimizing resources in multi-layer optimization, i.e., where the virtual layout *and* the routing on the layout are both optimized at once. However, for single layer optimization over a pre-set virtual layout, it is important to represent bus-LSPs in a usable way. This section brings an answer to this problem.

In this section, we will show a graph representation of bus-LSPs. This representation will be used to initialize bus-LSPs of length 1 over a link in Algorithm 5.

Let us assume we are considering a bus-LSP as depicted in Figure 3.6, running through 5 different nodes. Node 1 is ingress, 5 is egress, node 2 has add/drop capability, node 4 has only drop capability, and node 3 is just a transit node. We can transform this configuration into the graph depicted in Figure 3.7.

To do this we first create a virtual node associated with each true node connected to the bus-LSP: we create 5 virtual nodes $1'$,$2'$,$3'$,$4'$, and $5'$. The virtual node is then connected to its parent node with up to two directed links, depending on its access type to the bus-LSP: these are called *fork* links and they connect physical nodes to to virtual nodes (and conversely). In our example, this leads to the following fork links setup:

- (1,1'),
- (2,2') and (2',2),
- no links between nodes 3 and 3',
- (4',4),
- (5',5).

Then, all virtual nodes are connected (here 1',2',3',4',5'); they represent the bus-LSP itself, while the fork links represent router access. Capacities are set in the following way on the representation. The fork links capacities are set to the router treatment capacity. For example, if router 2 has interfaces between the routing layer (upper layer) and the optical

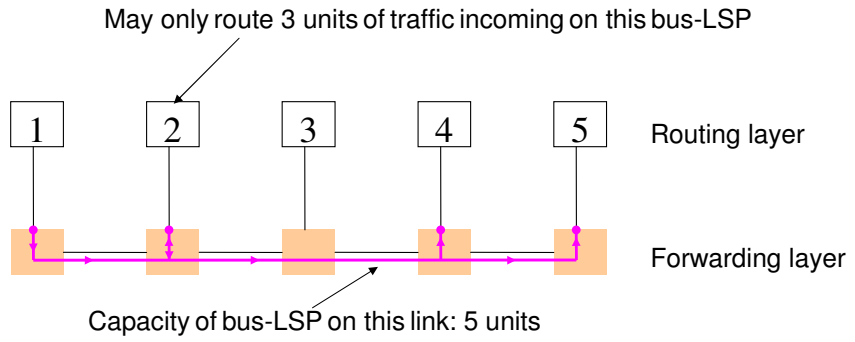


Figure 3.6: Bus-LSP

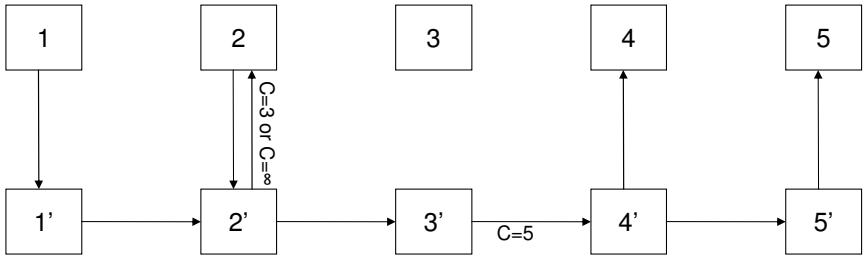


Figure 3.7: Bus-LSP equivalent representation

layer (lower layer) to send on this bus-LSP a volume of 3 units, the capacity is set to 3 on link (2,2') (one may set the capacity to ∞ if routing is not a constraint, as in Section 5.3). The capacity on the link between two virtual nodes is set to the capacity (Z_{gq}) available in the bus-LSP segment between the two true nodes associated to those virtual nodes. For example, if the bus-LSP has a capacity of 5 units of traffic between nodes 3 and 4, the capacity of link (3',4') is set to 5.

Note that a virtual node is associated with a given bus-LSP. If multiple bus-LSPs run through the same node, multiple virtual nodes are formed.

3.4 Concluding Remarks

In this chapter, we have introduced the novel bus-LSP structure. This structure is introduced in the GMPLS framework as a way for network operators to reduce both CAPEX and OPEX, in single- and multi-layer networks. This was qualitatively demonstrated, and will be quantitatively explored in following Chapters 4 and 5. Additionally, the representation of a bus-LSP in a graph allows integration of this complex structure in traditional graph-based studies.

Chapter 4

Bus-LSPs in Single Layer Environments

As has been shown in Chapter 3, there are two types of gain possible in single layer environments. First, if the layer's resource is available only in discrete increments, it is possible to multiplex more flows on a single link using bus-LSPs. This provides the ability to spare link bandwidth. The reader may refer to the work in [BPD+04] for some results linked to this strategy. However those results are based on *Distributed Aggregation* which only allow packet adding and not packet dropping: using bus-LSPs will provide CAPEX savings which will necessarily be greater or equal.

Second, it has been shown that making use of bus-FAs should reduce the overall number of FAs setup. This can directly be translated into OPEX savings. Indeed, a network's operation costs are directly related to the number of network entities to supervise and manage (FCAPS costs). The more network entities there are, the more people are needed for management: therefore reducing the number of said entities will result in lower operations costs. This chapter focuses on this aspect of management simplification.

4.1 Optimization Model for a Network Using bus-LSPs

In the following sections we will quantitatively analyze the benefits of bus-LSPs in terms of layout complexity reduction. In the present section we introduce a single layer network model and the various related notations that we use in the following sections.

4.1.1 Network and Paths

The physical network is represented by a directed graph $G = (V, A)$, where V is the set of vertices indexed by subscript $n = 1, 2, \ldots, N$, and A is the set of edges indexed by subscript

$e = 1, 2, \ldots, E$. The vertices represent here the GMPLS Label Switching Routers (LSRs) and each edge represents a unidirectional connecting link between them. Additionally, c_e denotes the nominal capacity of link e.

Since our target is to show the benefits of bus-LSPs in terms of reduction of the number of objects to be set-up and maintained, a path formulation of the network optimization problem is better adapted than a node-link formulation. Let subscript $p = 1, 2, \ldots, P$ index all considered paths in the network. Note of course that in practical cases, the network operator can select the list of paths that can be used; this is taken into account in our model. A path is characterized by the ordered succession of links it uses. In the bus-LSP context, a path and a bus-LSP will be considered as equivalent, that is, all nodes along a path will be able to add or remove packets on the bus. We suppose in this thesis that bus-LSPs do not cross a given router more than once.

4.1.2 Demands and Flows

Let subscript $d = 1, 2, \ldots, D$ index all network demands, that is the various end-to-end commodities. Demand d requires bandwidth h_d.

Since we consider a non-routed (single-layer) architecture, demands have to use a single (bus-)FA to go from source to destination.

Let δ_{edp} be the binary incidence coefficient that takes value 1 when path (or bus-LSP) p requires bandwidth on link e when satisfying demand d, and value 0 otherwise. We assume here that the signaling and routing protocols allow the residual bandwidth to vary along a given bus-LSP, depending on the initial nominal capacities of the links composing the bus-LSP and the bandwidth reservations made on each segment of the bus-LSP.

Let γ_{dp} be the binary incidence coefficient indicating if path (or bus-LSP) p can effectively transport demand d. Again, the operator can restrict the use of a path for the transport of a given flow so as to guarantee delay constraints for example.

Let x_{dp} be the continuous variable indicating how much flow from demand d is routed on path p.

Let u_p be the binary variable associated with path p indicating whether path p is activated or not.

4.2 Optimization Problems Formulation

Given the physical network configuration and a traffic matrix, the objective is to minimize the total number of LSPs a network operator has to manage. Below are formulated two problems that need to be solved for this purpose. The first consists in minimizing the

number of active LSPs with the constraints of transporting all the traffic without overloading any link, and the second has an additional target of keeping the overall bandwidth consumption minimal.

4.2.1 Reduced Complexity Layout With Bus-LSPs

MCL 1
Minimal Complexity Layout Problem

Given:
The topology, i.e., the set of links $\{1, 2, \ldots, E\}$; the sets of demands $\{1, 2, \ldots, D\}$ and of paths $\{1, 2, \ldots, P\}$ as well as:

- h_d for $d = 1, 2, \ldots, D$ (demand volume)
- c_e for $e = 1, 2, \ldots, E$ (link capacity)
- δ_{edp} for $d = 1, 2, \ldots, D$; $e = 1, 2, \ldots, E$; $p = 1, 2, \ldots, P$
- γ_{dp} for $d = 1, 2, \ldots, D$; $p = 1, 2, \ldots, P$

Minimize:

$$\sum_{p=1}^{P} u_p \tag{4.2.1}$$

Subject to:

$$\sum_{d=1}^{D} \sum_{p=1}^{P} \delta_{edp} x_{dp} \leq c_e \quad \text{for } e = 1, 2, \ldots, E \tag{4.2.2}$$

$$\sum_{p=1}^{P} x_{dp} \gamma_{dp} = h_d \quad \text{for } d = 1, 2, \ldots, D \tag{4.2.3}$$

$$x_{dp} \leq h_d u_p \quad \text{for } d = 1, 2, \ldots, D,\ p = 1, 2, \ldots, P \tag{4.2.4}$$

The operational expense for a network operator is related to the layout complexity. In the thesis, we define complexity as the total number of LSPs which are simultaneously active. In this section, we formulate a multi-commodity flow problem with the objective of minimizing this number of bus-LSPs. The MCL (Minimal Complexity Layout) problem is formulated as Mixed Integer Linear Program (MILP), as some of the variables (like the u_p) are integers and other (like the x_{dp}) are continuous. The paths which may be used in this formulation are predefined, that is a set of paths is used as an input to the problem. Constraint (4.2.2) ensures that no link carries more flow than its nominal capacity. Con-

straint (4.2.3) expresses the fact that all the demand is satisfied and that only paths which connect the endpoints of a demand are used. Constraint (4.2.4) ensures that there is no flow on inactive bus-LSPs (for which u_p=0). We allow here for load-balancing, or flow bifurcation, since we do not impose for a commodity to be satisfied by a single path.

4.2.2 Reduced Complexity, Minimal Overall Bandwidth Layout With Bus-LSPs

As we will show in the numerical study in Section 4.3, minimizing the number of bus-LSPs can lead to wasting a lot of bandwidth, as demands may be multiplexed on too long bus-LSPs. Therefore, we propose another formulation, where we have the additional constraint that the overall used bandwidth does not exceed the minimal overall bandwidth required to support the demand in a network that does not implement bus-LSPs.

This MCL-BC (Minimal Complexity Layout with Bandwidth Constraint) problem is solved in two steps. The first one consists in determining the minimal link bandwidth consumption (B^*) of a layout not using bus-LSPs. This is obtained by solving a MCL problem similar to the one formulated in Section 4.2.1 for which only objective (4.2.1) is changed and becomes:

$$minimize : B = \sum_{e=1}^{E}(\sum_{p=1}^{P}\sum_{d=1}^{D}\delta_{edp}x_{dp}) \tag{4.2.5}$$

This leads to a minimal network utilization B^*. As the u_p variables are no longer constrained, they may all be set to 1, which means that flows can be routed on all paths. However, only those using minimal amount of bandwidth will be selected, regardless of how many are used and whether load is balanced or not. We then use this to solve a variation of the MCL problem with this optimal bandwidth utilization as an additional constraint (4.2.10).

MCL-BC 1
Minimal Complexity Layout with Bandwidth Constraint Problem

Given:
The sets of links $\{1, 2, \ldots, E\}$, of demands $\{1, 2, \ldots, D\}$ and of paths $\{1, 2, \ldots, P\}$ as well as:

- h_d for $d = 1, 2, \ldots, D$
- c_e for $e = 1, 2, \ldots, E$
- δ_{edp} for $d = 1, 2, \ldots, D$; $e = 1, 2, \ldots, E$; $p = 1, 2, \ldots, P$
- γ_{dp} for $d = 1, 2, \ldots, D$; $p = 1, 2, \ldots, P$

Minimize:

$$\sum_{p=1}^{P} u_p \tag{4.2.6}$$

Subject to:

$$\sum_{d=1}^{D}\sum_{p=1}^{P} \delta_{edp} x_{dp} \leq c_e \quad \text{for } e = 1, 2, \ldots, E \tag{4.2.7}$$

$$\sum_{p=1}^{P} x_{dp}\gamma_{dp} = h_d \quad \text{for } d = 1, 2, \ldots, D \tag{4.2.8}$$

$$x_{dp} \leq h_d u_p \quad \text{for } d = 1, 2, \ldots, D,\ p = 1, 2, \ldots, P \tag{4.2.9}$$

$$\sum_{e=1}^{E}(\sum_{p=1}^{P}\sum_{d=1}^{D} \delta_{edp}.x_{dp}) \leq B^* + \epsilon \tag{4.2.10}$$

where ϵ is a tolerance allowing a slightly higher use of bandwidth as compared to B^*.

Objective (4.2.6) and constraints (4.2.7), (4.2.8), (4.2.9) are respectively the same as (4.2.1), (4.2.2), (4.2.3) and (4.2.4).

4.2.3 Problem Formulation Improvements and Additional Constraints

An operator might want to ensure that all data flows associated with a given commodity use the same path. This can be imposed in the model by adding a binary variable u_{dp} which is set to 1 if demand d uses path p, and two constraints:

$$\sum_{p=1}^{P} u_{dp} = 1 \quad \text{for } d = 1, 2, \ldots, D \tag{4.2.11}$$

$$x_{dp} \leq h_d u_{dp} \quad \text{for } d = 1, 2, \ldots, D; p = 1, 2, \ldots, P \tag{4.2.12}$$

Constraint (4.2.11) guarantees that only one path will be used for a given demand. Constraint (4.2.12) ensures that flow will be assigned only to the path which is designated for routing the demand.

Another additional constraint one might want to consider is the possibility of weighting bandwidth utilization on different links, to take into account such factors as the length or the technology of the link. By introducing an additional cost coefficient g_e for each link e, one may replace (4.2.5) by:

$$minimize : \sum_{e=1}^{E} (\sum_{p=1}^{P} \sum_{d=1}^{D} \delta_{edp} g_e x_{dp}) \tag{4.2.13}$$

The new objective (4.2.13) will yield a different overall network cost $B^{*\prime}$, which is used to replace constraint (4.2.10) by:

$$\sum_{e=1}^{E} (\sum_{p=1}^{P} \sum_{d=1}^{D} \delta_{edp} g_e x_{dp}) \leq B^{*\prime} \tag{4.2.14}$$

Finally, if the results produced by this formulation are to produce an average point of operation which will be subject to traffic matrix variations in time, it can be important to change constraints (4.2.2) and (4.2.7) to:

$$\sum_{p=1}^{P} \sum_{d=1}^{D} \delta_{edp} g_e x_{dp} \leq c_e . \Theta \quad \text{for } e = 1, 2, \ldots, E \tag{4.2.15}$$

where Θ is a coefficient ensuring that no link in the resulting solution will be excessively loaded, thus leaving some margin for traffic variations.

As a side note, we would like to point out that the formulations given here do not ensure that the bus-LSPs used in the solution will effectively transport traffic on their entire length. It may be possible to shorten the bus-LSPs which are used as some segments may not transport any traffic at all. Also some bus-LSPs may transport demands which do not overlap, that is, these bus-LSPs could be cut in two (or more) without any impact on demand satisfaction. However, there is no way to avoid this. We do take these aspects into consideration in the heuristic solving method described in Section 5.3 of Chapter 5, which does not provide solutions using such "degenerate bus-LSPs" (that is, bus-LSPs

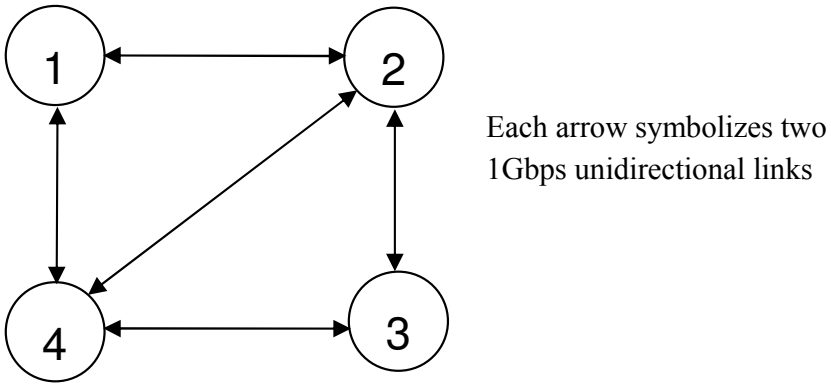


Figure 4.1: NET1 Network

which have segments without any flow on them, and could thus be split).

4.3 Numerical Results

We present here a numerical study of the problems MCL and MCL-BC. We compare the complexity of the layouts when bus-LSPs are allowed and not allowed and show that the bus-LSP allows for drastically reducing this complexity. Since we consider the single layer case, the number of LSPs required in the non-bus LSP case is at least the total number of demands. To simplify the presentation, we compare the number of required bus-LSPs with the total number of demands, which corresponds to a worst case of the gain.

The results are obtained by submitting the formulated problem to the numerical solver ILOG CPLEX. We present results for two networks, noted NET1 and VTHD. NET1 (Figure 4.1) is a 4 node network, with 10 unidirectional links (this very simple network topology allows to develop some intuition on the benefits of bus-LSPs). VTHD (Figure 2.3) is the 11 node and 28 links network, based on a French National Research Network, deployed for the project VTHD (Vraiment Très Haut Débit) [VTHD]. Due to the complexity of the CPLEX program associated to the problem formulated in Section 5.2.2 (see AMPL files in Appendix A.2), we were not able to perform tests on large networks, and could not provide results for the Italian network.

We generate traffic demands for all pairs of nodes in the networks by using independent uniform random variables distributed in the interval [0, maxload]. We chose values for maxload such that the results show the network under light load, where all links are relatively unloaded; and under high load, where many links are loaded at their nominal capacity.

It is interesting to note that the solutions are not unique; more than one LSP layout may yield the same bus-LSP count and optimal bandwidth use. Even when dealing with

Table 4.1: MCL Results

		Network			
		NET1 (12 demands)		VTHD (110 demands)	
Maxload (Mbps) →		200	500	20	180
Number of bus-LSPs		2	3	5	7
Bandwidth used (Gbps)		2.02	6.21	3.75	25.83
Length of bus-LSP (links)	Avg	3	3	7.8	7.57
	Max	3	3	9	9
Demands per bus-LSP	Avg	6	4.33	22	17.29
	Max	6	5	34	23
Demands per link	Avg	2	1.9	14.32	11.93
	Max	4	4	32	16

homogeneous symmetric traffic matrixes, the layout provided by the optimal solution is not always symmetric: there is not necessarily a reverse matching bus-LSP for every bus-LSP.

Table 1 provides numerical results for the MCL case. The main result here is the reduction of the number of required LSPs which, under high traffic loads, is brought down from 12 to 3 for NET1 and from 110 to 7 for VTHD. Such a significant reduction in the complexity of the layout is quite important considering that the related operational costs will also decrease in an important amount.

We observe in Table 4.1 that the total number of flows transported by the bus-LSPs is larger than the number of commodities. The reason is that, as indicated before, load sharing is allowed between bus-LSPs.

Note that the bus-LSPs go through an important number of nodes: for NET1, they all visit all nodes, and for VTHD, they visit between 6 and 10 of the 11 nodes. It is finally interesting to see that the established bus-LSPs carry a great number of demands, proving the fact that there is a good reuse factor of a bus-FAs; the average number of demands running on a link actually proves this point as well. This reuse factor however decreases as the load increases, because the links along the bus saturate and cannot convey any more demands.

We now will discuss the MCL-BC problem. For the simulations of this problem, we set the value of ϵ (see constraint (4.2.10)) to 1/10,000 of the value of the optimal bandwidth utilization. We used here the same traffic matrixes as those generated for the study of MCL so as to compare results between both methods.

Table 4.2: MCL-BC Results

		Network			
		NET1 (12 demands)		VTHD (110 demands)	
Maxload (Mbps) →		200	500	20	180
Number of bus-LSPs		4	4	14	14
Bandwidth used (Gbps)		1.48	4.72	2.14	19.80
Length of bus-LSP (links)	Avg	3	2.75	6.6	6.5
	Max	3	3	8	9
Demands per bus-LSP	Avg	3	2	7.86	8
	Max	3	3	11	14
Demands per link	Avg	1.4	1.4	8.21	8.43
	Max	2	2	11	12

First of all, the number of bus-LSPs required here is notably larger than in the previous case. However, the gain is still important, reducing the number of LSPs from 12 to 4 for NET1 and from 110 to 14 for VTHD.

The second important result is the bandwidth utilization. When solving the MCL-BC, the bandwidth utilization is at least 23% less than when solving the MCL. This therefore may justify the use of the MCL-BC as a traffic engineering tool, as it simultaneously gives a good utilization of the network resources and reduces the operational costs by simplifying the layout.

It is interesting to remark that in our examples the number of required bus-LSPs does not depend on the load. This is of course not always the case, but happens quasi-systematically when the demand matrixes are generated uniformly. As with the MCL problem, the average length of bus-LSPs is reduced when the load increases, as some very important demands will fill a link to nominal capacity, preventing any other demand from using it: this generates shorter bus-FAs. On average, the number of demands running on each bus-LSP and on each link has reduced, which comes from the fact that the load is spread out more evenly so as to allow better bandwidth utilization.

4.4 Concluding Remarks

In this chapter we analyzed the benefits of introducing bus-LSPs in a GMPLS network. We show that this concept allows for reducing both the OPEX and CAPEX and we

evaluated quantitatively this gain for two network topologies in the single layer case. For this purpose, we have formulated and analyzed two mixed-integer linear programs which are realistic models of the analyzed networks.

We will present in the following chapter an optimal design for the multi-layer case. As these bus-LSPs are to be integrated in GMPLS controlled routed multilayer networks, we will evaluate both the layout complexity reduction and the resource savings bus-LSPs can provide in such networks.

Chapter 5

Bus-LSPs in Multi Layer Environments

In this chapter, we study the use of bus-LSPs in multi-layer environments. As has been shown in the previous chapters, bus-LSPs allow the operator to reduce both the CAPEX and the OPEX, by lowering the number of routing equipment and limiting the complexity of the virtual layouts. Our contribution is the design and evaluation of a virtual layout in a bus-LSP-capable network. The goal is to maximize the utilization of such a network by operators, through both maximizing the traffic scaling and minimizing upper-layer routing. We show how bus-LSPs can be used to reduce CAPEX and OPEX of a multi-layer network. Concerning CAPEX, we show that resources (like bandwidth and switching capacity) and components (like interfaces and electro-optical converters) can be saved. Concerning OPEX, we show that bus-LSPs allow reducing the layout complexity. We measure this complexity by the number of LSPs that have to be established and maintained.

5.1 An Introductory Example

We will demonstrate the value of bus-LSPs using the sample network shown in Figure 5.1. This example was designed on purpose, to prove the value of bus-LSPs in a simple way in the multi-layer network case. Let us assume that every link supports a single wavelength

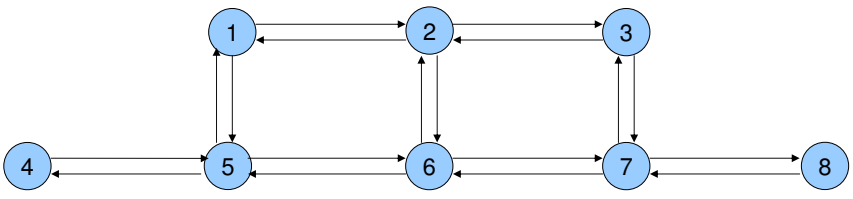


Figure 5.1: Sample 8-node Network

in each direction. Each wavelength may carry 1 unit of traffic. The demands are:

- $1 \rightarrow 3$: volume 0.3
- $1 \rightarrow 6$: volume 0.1
- $2 \rightarrow 7$: volume 0.1
- $3 \rightarrow 8$: volume 0.1
- $4 \rightarrow 8$: volume 0.3.

Nodes 5,6 and 7 can route 0.1 units of traffic. This can satisfy, at best, only 25% of each demand without using bus-LSPs, this means saturating routers 5 and 7, and setting up at least 7 LSPs, for instance:

- 1-2-3
- 3-7
- 2-1-5
- 3-2-6
- 4-5
- 5-6-7
- 7-8.

This exact result was obtained using the CPLEX MIP solver. Using bus-LSPs, 100% of demands can be satisfied, by simply opening 2 bus-LSPs (1-5-6-7-3 and 4-5-1-2-3-7-8).

Now, suppose nodes 5, 6 and 7 cannot route any traffic. Without using bus-LSPs, it is impossible to route all demands, even partially, whereas when using bus-LSP-aware nodes, all demands can still be fully satisfied, using the same layout as before.

5.2 Optimization Problem

5.2.1 Optimization Model for a Multi-Layer Network Using Bus-LSPs

The physical network (lower layer network, PN) is represented by a directed graph $\mathcal{N} = (\mathcal{V}, \mathcal{G})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{G}$ is the set of links. Nodes $v \in \mathcal{V}$ represent GMPLS Label Switching Routers (LSRs), and links $g \in \mathcal{G}$ represent physical directed connecting links between two nodes. Quantity c_g denotes the capacity of link g, and M is a discrete module of bandwidth allocated to the lower layer resources, for example, the capacity of

one wavelength in WDM. O_v is the maximum outgoing bandwidth capacity of the router at node v, and I_v—its maximum incoming bandwidth capacity.

The virtual network (upper layer network, VN) is represented by a directed graph $\mathcal{M} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (the same as in PN—we assume that all nodes have full multi-layer capabilities), and $\mathcal{E}$ is the set of virtual links. Capacity of link $e \in \mathcal{E}$ is denoted by y_e (it will be a variable). Each virtual link $e \in \mathcal{E}$ represents a forwarding adjacency (FA) such as defined in Section 3. Each bus-LSP will therefore generate multiple FAs in the virtual topology. In the sequel we will use notation $\delta^+(v)$ and $\delta^-(v)$ for the set of all virtual links $e \in \mathcal{E}$ outgoing from and incoming to node $v \in \mathcal{V}$, respectively. Finally, set $\mathcal{D}$ will represent all network demands, that is the collection of end-to-end commodities. Demand $d \in \mathcal{D}$ requires bandwidth h_d.

Because of the strong path-oriented nature of this problem, path formulation of the network optimization problem is more suitable than node-link formulation. The set of all paths in PN (i.e., sequences of physical links) formed to realize links $e \in \mathcal{E}$ is denoted by $\boldsymbol{\mathcal{Q}}$. Then $\mathcal{Q}_e \subseteq \boldsymbol{\mathcal{Q}}$ is the set of all physical paths that can be used to realize capacity y_e of virtual link $e \in \mathcal{E}$. Clearly, $\boldsymbol{\mathcal{Q}} = \bigcup_{e \in \mathcal{E}} \mathcal{Q}_e$. Moreover, since paths $q \in \boldsymbol{\mathcal{Q}}$ are supported by bus-LSPs, we need to distinguish a subset $\hat{\boldsymbol{\mathcal{Q}}}$ of all maximal paths (i.e., the paths representing bus-LSPs) in the set of all paths $\boldsymbol{\mathcal{Q}}$ ($\hat{\boldsymbol{\mathcal{Q}}} \subseteq \boldsymbol{\mathcal{Q}}$). We assume that $q \in \hat{\boldsymbol{\mathcal{Q}}}$ if, and only if, $q \in \boldsymbol{\mathcal{Q}}$ and q is not a subpath of any other path $q' \in \boldsymbol{\mathcal{Q}}$, i.e., $\forall q' \in \boldsymbol{\mathcal{Q}},\ q' \not\supseteq q$. Next, with each path $q \in \hat{\boldsymbol{\mathcal{Q}}}$ we associate set $Q(q)$ which is the set of all subpaths of the maximal path q that connect the end nodes of some virtual link $e \in \mathcal{E}$, i.e., $Q(q) = \{q' : q' \in \boldsymbol{\mathcal{Q}} \wedge q' \subseteq q\}$. Finally, with each $q \in \mathcal{Q}_e$ we associate the quantity $\Delta_q = \max\{c_g : g \in q\}$ which bounds the capacity of virtual link $e \in \mathcal{E}$ that can be realized on q.

The set of all paths in the VN is denoted by $\boldsymbol{\mathcal{P}}$, and $\mathcal{P}_d \subseteq \boldsymbol{\mathcal{P}}$ is the set of all (virtual) paths $p \in \boldsymbol{\mathcal{P}}$ that can be used to realize demand $d \in \mathcal{D}$. Certainly, $\boldsymbol{\mathcal{P}} = \bigcup_{d \in \mathcal{D}} \mathcal{P}_d$.

We note that in practice the network operator can select the list of paths $\mathcal{Q}_e$ to be used for realizing virtual link $e \in \mathcal{E}$; this is taken into account in our model. Note also that path $q \in \mathcal{Q}_e$ is characterized by the sequence of links it uses. In the bus-LSP context, a bus-LSP is equivalent to a maximal path, that is, all nodes along a maximal path will be able to add or remove packets on the bus. We assume that bus-LSPs do not cross a given router more than once and that signaling and routing protocols allow the bandwidth to vary along a given bus-LSP, depending on the capacities of the links composing the bus-LSP and the bandwidth reservations made on each segment of the bus-LSP. We also note that the operator can restrict the length of paths in the VN for the transport of a given demand to guarantee a maximum number of hops. He may also restrict the number of physical links to appear in a given bus-LSP. This allows to bound the overall delay (under the assumption that delay is independent of the virtual and physical link loads).

Let x_{dp} be a continuous variable (flow) indicating what portion of volume of demand d is

routed on path $p \in \mathcal{P}_d$. Similarly, let z_{eq} be a continuous flow allocated to path $q \in \mathcal{Q}_e$ (a bus-LSP or its subpath) realizing the capacity of virtual link e. We will also use an auxiliary continuous variable Z_{gq} expressing, for each bus-LSP $q \in \hat{\mathcal{Q}}$ such that $g \in q$, the load of g induced by all flows on q using link g.

We will also use two sets of binary variables. First, U_q will denote a binary variable associated with each maximal path (bus-LSP) $q \in \hat{\mathcal{Q}}$, indicating whether path q is activated or not. (Variable S will express the total number of active bus-LSPs.) Second, u_{eq} will be the binary variable indicating whether or not link e is actually using path $q \in \mathcal{Q}_e$.

5.2.2 Optimization Problem Formulation

An important part of the network operator's capital expense (CAPEX) optimization consists in maximizing the amount of traffic a given initial investment will bring him. Therefore, increasing this amount is a direct source of revenue. So is the reduction of the total routing equipment needed. The operational expense for a network operator is related to the layout complexity. In this thesis, we define complexity as the total number of active LSPs.

Given a physical network configuration and a traffic matrix, the objective is to use bus-LSPs to maximize the amount of traffic the operator could transport, while minimizing routing costs and total complexity of the layout.

First, we design the virtual topology, using bus-LSPs to transport the maximal amount of traffic, represented by variable H. Once this optimal amount is found, we next minimize the costs in routing equipment in the upper layer, which in a first approximation is equivalent to reducing the overall amount of routed traffic. Finally, we minimize the total number of LSPs a network operator has to manage, which directly impacts OPEX.

In this section, we formulate a multi-commodity flow problem with these objectives. The VTD (Virtual Topology Design) problem is formulated as Mixed Integer Linear Program (MILP), as it uses both integer and continuous variables [PM04]. The hypothesis is that all nodes are able to both route IP traffic and switch optical connections.

VTD 1
Virtual Topology Design Problem

Given:
set of nodes $\mathcal{V}$, set of physical links $\mathcal{G}$, set of virtual links $\mathcal{E}$, set of demands $\mathcal{D}$, set of physical paths $\boldsymbol{\mathcal{Q}}$, set of virtual paths $\boldsymbol{\mathcal{P}}$, and

- h_d, volume of demand $d \in \mathcal{D}$
- c_g, capacity of physical link $g \in \mathcal{E}$
- I_v, O_v, bounds on incoming and outgoing capacity of node $v \in \mathcal{V}$
- M, capacity module

$$\textbf{lexmax} \quad (H, -R, -S) \tag{5.2.1}$$

Subject to:

$$\sum_{p \in \mathcal{P}_d} x_{dp} \geq H h_d, \quad d \in \mathcal{D} \tag{5.2.2}$$

$$\sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d:\, e \in p} x_{dp} = y_e, \quad e \in \mathcal{E} \tag{5.2.3}$$

$$\sum_{q \in \mathcal{Q}_e} z_{eq} = y_e, \quad e \in \mathcal{E} \tag{5.2.4}$$

$$R = \sum_{e \in \mathcal{E}} y_e \tag{5.2.5}$$

$$\sum_{e \in \delta^-(v)} y_e \leq I_v, \quad v \in \mathcal{V} \tag{5.2.6}$$

$$\sum_{e \in \delta^+(v)} y_e \leq O_v, \quad v \in \mathcal{V} \tag{5.2.7}$$

$$S = \sum_{q \in \hat{\boldsymbol{\mathcal{Q}}}} U_q \tag{5.2.8}$$

$$u_{eq'} \leq U_q, \quad q \in \hat{\boldsymbol{\mathcal{Q}}} \quad e \in \mathcal{E} \quad q' \in Q(q) \cap \mathcal{Q}_e \tag{5.2.9}$$

$$\sum_{q \in \mathcal{Q}_e} u_{eq} = 1, \quad e \in \mathcal{E} \tag{5.2.10}$$

$$z_{eq} \leq \Delta_q u_{eq}, \quad e \in \mathcal{E} \quad q \in \mathcal{Q}_e \tag{5.2.11}$$

$$Z_{gq} = \sum_{e \in \mathcal{E}} \sum_{q' \in Q(q) \cap \mathcal{Q}_e:\, g \in q'} z_{eq'}, \quad g \in \mathcal{G} \quad q \in \hat{\boldsymbol{\mathcal{Q}}} \tag{5.2.12}$$

$$\sum_{q \in \hat{\boldsymbol{\mathcal{Q}}}} \lceil \frac{Z_{gq}}{M} \rceil \leq c_g, \quad g \in \mathcal{G}. \tag{5.2.13}$$

Constraint (5.2.2) ensures that all demands are satisfied when multiplied by factor H, which is the first objective in (5.2.1). Equalities (5.2.3) define the capacity of each virtual link to be equal to its load. Constraint (5.2.4) ensures that link capacities will actually be realized by means of PN flows. (5.2.5) defines the overall amount of virtual capacity in the network—this is the second objective minimized by (5.2.1).

Constraints (5.2.6) and (5.2.7) bound the amount of incoming and outgoing traffic, respectively, routed through an upper layer node v. Minimizing the total number of active bus-LSPs, defined in (5.2.8), is the third objective in (5.2.1). Constraint (5.2.9) forbids assigning flows to inactive bus-LSPs. In the formulation we do not assume load-balancing (flow bifurcation) in the lower layer since virtual links are supported by single bus-LSP's—this is forced by (5.2.10) and (5.2.11). Equalities (5.2.12) specify the load of a PN link g induced by bus-LSP q.

Finally, constraint (5.2.13) accounts for the fact that total flow on a particular bus-LSP is allocated to PN links in discrete modules M. This is a critical constraint which actually reveals the benefit from using bus-LSPs instead of regular LSPs.

Notice that although (5.2.13) is not written in the linear form, it is easy to transform this inequality into a set of linear constraints using integer variables Y_{gq}:

$$Z_{gq} \leq MY_{gq}, \quad g \in \mathcal{G} \quad q \in \hat{\boldsymbol{\mathcal{Q}}} \tag{5.2.14}$$

$$\sum_{q \in \hat{\boldsymbol{\mathcal{Q}}}} Y_{gq} \leq c_g, \quad g \in \mathcal{G}. \tag{5.2.15}$$

5.2.2.1 Problem Formulation Improvements and Additional Constraints

An additional constraint one might want to consider is the possibility of weighting some demands compared to others. This is particularly useful if one knows the relative growth speeds of the various demands, i.e., the traffic matrix may not scale uniformly. This could also be useful if one has a premium client whose demands he wants to prioritize and let grow more. In this case, it is useful to introduce an additional coefficient s_d for each demand d, which represents the growth speed of demand d. For example, demand d_1 may grow twice as fast as demand d_2: $s_{d_1} = s_{d_2} * 2$. In this case, (5.2.2) can be replaced by:

$$\sum_{p \in \mathcal{P}_d} x_{dp} \geq H s_d h_d, \quad d \in \mathcal{D} \tag{5.2.16}$$

In Chapter 4, a coefficient had been introduced to allow for some margin around the operation point. This is not something which is needed here, since the traffic multiplication is maximized: this guarantees that the traffic matrix will have as much margin for growth as possible.

To be completely exact, the costs in routing equipment are actually equal to the number of routing ports on each router. This quantity increases by discrete amounts and not continuously, therefore the overall amount of routed traffic is not strictly equal to the routing equipment costs. This implies transforming constraint (5.2.5) into:

$$R = \sum_{e \in \mathcal{E}} P\lceil y_e / P \rceil \tag{5.2.17}$$

where P is the capacity of a router port. Although this is not in the linear form, a transformation similar to the one presented for (5.2.13) can be applied to make it linear.

5.3 Heuristic Algorithm

We describe here a set of algorithms, which, when combined, will approximatively solve problem VTD 1, as described in Section 5.2.

This heuristic approach is not distributed in its nature: knowledge of all demands is necessary, and thus, a centralized implementation is best adapted. Our algorithms are meant to be used statically, however it runs fast enough for a semi-dynamic approach to be considered. However, since the optimization routines reroute established demands, this introduces traffic disruption which is not desired in dynamic contexts.

5.3.1 Forbidden Turns Shortest Path First Algorithm

We first specify a constrained shortest path algorithm: the Forbidden Turns Shortest Path First (FT-SPF) algorithm. The goal of this algorithm is to find a shortest path between a source node and a destination node without using some specific "turns".

Let us first define a *turn* as a pair of adjacent links. Turn (g_1, g_2) is forbidden if a shortest path yielded by our FT-SPF algorithm cannot use g_1 and g_2 successively. However the SP may use g_1 or g_2 alone, or even both, as long as they do not appear immediately one after another in the solution.

Algorithm 3, FT-SPF, defines a procedure to find a shortest path without any of the forbidden turns which are given in a list $\mathcal{L}$. This algorithm is based on the Dijkstra algorithm [Dij59]. The notations in FT-SPF are as follow:

- $from_v^d \quad , v \in \mathcal{V}$, is the node from which one must come when in v to reach the destination d if no other specific route is specified (*default* route)
- $from_v^{v'} \quad , v \in \mathcal{V}$, is the node from which one must come to reach v, when v' is included in the route: this is a *specific* route
- x can be either a superscript for the default route or a specific route

- $distance_v^x \quad , v \in \mathcal{V}$, is the sum of the weights needed to reach v when on a route (specific or default).
- $M_v, v \in \mathcal{V}$, is a list of nodes which cannot be reached by the default route flowing through v.

Algorithm 3: Forbidden Turns - Shortest Path First (FT-SPF)

Given: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
Given: c_a the capacity of link $l, \forall l \in \mathcal{A}$
Given: $w(a)$ the weight of link $l, \forall l \in \mathcal{A}$
Given: A source node s and a sink node d.
Given: A list of forbidden turns $\mathcal{F}$
Find: A shortest path between s and d in which no forbidden turns appear, if such a path exists

Create a queue Q and push s in Q
Initialize $M_v = \emptyset \quad \forall v$
Initialize $from_v^x = -1 \quad \forall v, x$
Initialize $distance_v^x = \infty \quad \forall v, x$
while $Q \neq \emptyset$ **do**
    $where = extract(Q)$
    $M_{where} \leftarrow M_{where} \bigcup \{v \in \mathcal{V} / (from^d(where) - where - v) \in \mathcal{F}$ AND $(from^v(where) - where - v) \in \mathcal{F}\}$
    $M_{where} \leftarrow M_{where} \setminus \{where\}$
    for all node $next$ adjacent to $where$ such that ($next \notin M_{where}$ OR $distance_{where}^{next} < \infty$) **do**
        if $distance_{next}^x > distance_{where}^x + w(where, next)$ **then**
            $from^x(next) = where$
            $distance^x(next) = distance_{where}^x + w(where, next)$
            $M_{next} \leftarrow M_{next} \bigcup M_{where}$
            push $next$ to Q
        end if
    end for
end while
Shortest path without forbidden turns is simply obtained by reversely following the default route from destination to source, branching onto specific routes when needed.

When $\mathcal{L} = \emptyset$, FT-SPF is equivalent to Dijkstra's algorithm. When there is one or more forbidden turns, the full search guarantees that if there is an alternate path, it will be found. Indeed, the marking of *missed nodes* guarantees the propagation of the algorithm to nodes which have been explored when some forbidden turn has blocked the algorithm. The algorithm therefore performs better than $(|\mathcal{L}| + 1)$ times Dijkstra, since parts of the default route are not recalculated for every forbidden turn. If Q is the queue managing the nodes to be considered by the algorithm, complexity depends on how nodes are popped from Q. In our implementation, we extract from Q the node with the least default cost

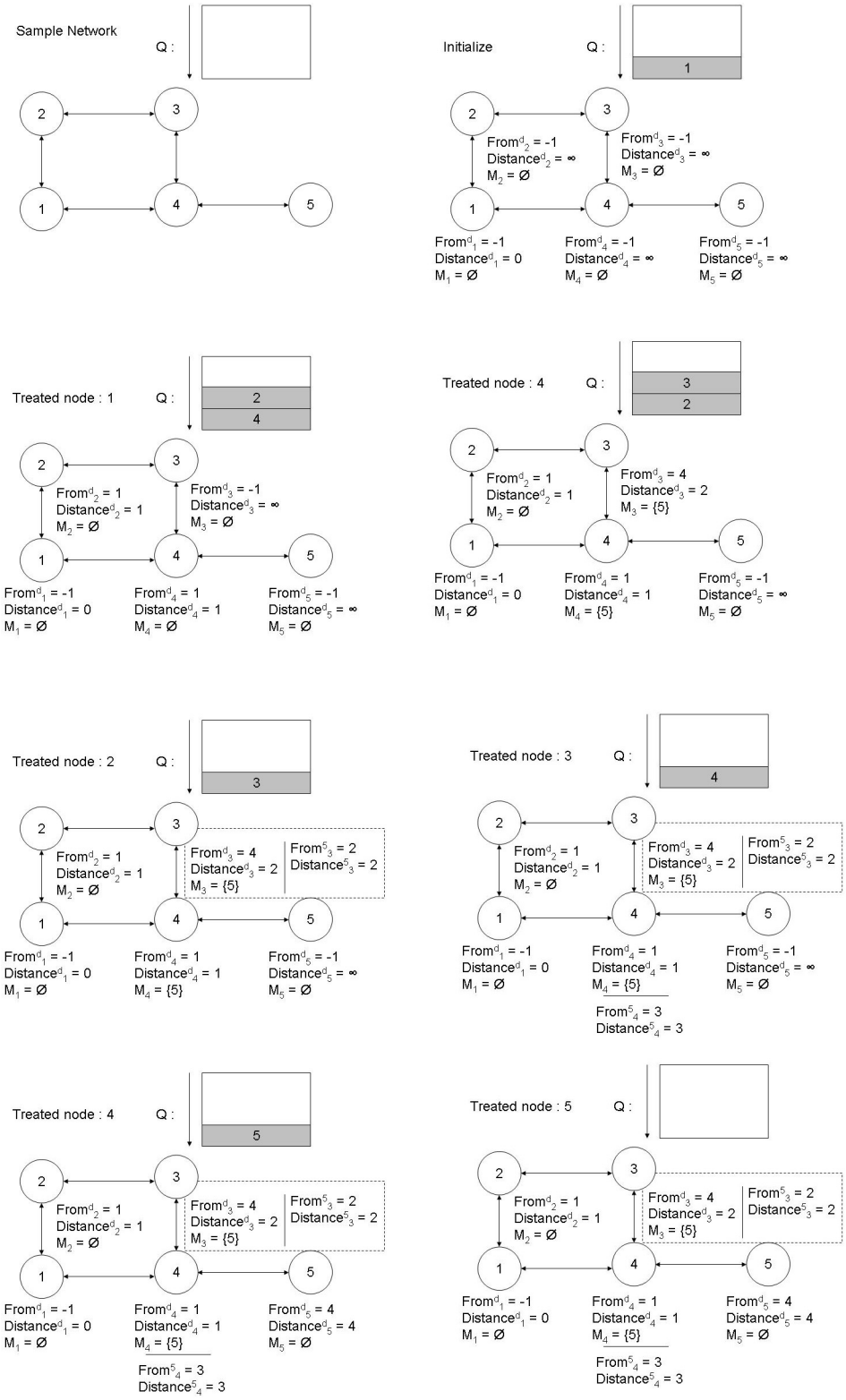


Figure 5.2: FT-SPF applied to a sample network

(cost on the default route) from linear storage: the complexity is therefore: $O((|\mathcal{L}|+1).n^2)$. An example of this algorithm on a simple network is demonstrated in Figure 5.2.

5.3.2 Potential and Interference-Based Routing Algorithm for Lexicographically Maximizing a Traffic Matrix

In this Section, we propose an extension to the Minimum Interference Routing Algorithm (MIRA) [KKL00] to route the demands over the network. Our Potential- and Interference-Based Routing Algorithm (PIBRA) is designed to achieve MinMax optimization on a weighted set of demands (that is maximize scaling of the traffic matrix). While MIRA is intended for environments in which future request is anticipated, we will use PIBRA for static traffic matrices. This is to allow re-routing of demands separately, and partial matrix rerouting. PIBRA is an algorithm which routes demands one at a time, so is suited for this task. Moreover, such an approach would allow PIBRA to be used, were the heuristic algorithm to be adapted for dynamic traffic matrices.

We will define here the *potential* of a demand as its ratio of *maxflow* and volume, where the maxflow is the maximal amount of flow which can be sent on the network, without bifurcation. For a given demand d, the higher its potential, the more its volume h_d can be increased (without having to change the routing of demand d or the network capacities).

We define here the WLEX-MAX problem. Given a network and a current routing of $D-1$ demands, we wish to route a new demand d with volume h_d. Given $\mathcal{F}$, the set of feasible $(D-1)$-vectors of *potential* values for the $D-1$ other demands after d is routed, the objective of WLEX-MAX is to find $\tilde{\Theta} \in \mathcal{F}$ such that $I(\tilde{\Theta}) \geq I(\Theta)$ for all $\Theta \in \mathcal{F}$, where $I(\Theta)$ represents a nondecreasing arrangement of Θ.

One of the approximations on which this algorithm relies is that the ordering of potentials will remain the same after demand d is routed. This is clearly not always true, but is more likely to happen if the volume h_d of demand d is small. This is why we will apply PIBRA to the list of demands, sorted by decreasing volumes.

Intuitively, the weights we are looking for must be such that the path of the demand which has the smallest potential is protected the most, then the second lowest potential, etc. Therefore, by applying a similar reasoning as in [KKL00] we define a new set of weights for the links. If we are able to number the demands which are already routed such that:

$$\tilde{\Theta}_{d_1} \leq \tilde{\Theta}_{d_2} \leq \ldots \leq \tilde{\Theta}_{d_{D-1}} \tag{5.3.1}$$

then we can obtain as in [KKL00], with m the number of links in the network, the following set of weights:

$$\beta_i = \begin{cases} 1 & i = (D-1) \\ mh_d(1+mh_d)^{D-i-2} & i = (D-2), \ldots, 1 \end{cases} \tag{5.3.2}$$

with the desired property:

$$\beta_i > (\beta_{i+1} + \beta_{i+2} + \ldots + \beta_{D-1}) \quad i = (D-2), \ldots, 1 \tag{5.3.3}$$

Algorithm 4: Potential- and Interference-Based Routing Algorithm (single demand)

Given: A graph $\mathcal{N} = (\mathcal{V}, \mathcal{G})$
Given: A set $\mathcal{B}$ of residual capacities on all the links.
Given: An ingress node a and an egress node b between which a demand d of volume h_d must be setup.
Find: A route between a and b having necessary capacity to satisfy d

1: Eliminate all links which have residual bandwidth less than h_d and form a reduced network
2: Compute the maxflow values and the critical link sets $C_{d'} \quad \forall d' \in \mathcal{D} \backslash \{d\}$
3: Sort the demands according to the potential and calculate their coefficients $\beta_{d'}$ based on formula (5.3.2).
4: Compute the link weights

$$w(g) = \sum_{d' : l \in C_{d'}} \beta_{d'} \quad \forall g \in \mathcal{G}. \tag{5.3.4}$$

5: Using FT-SPF, compute the shortest path in the reduced network with $w(g)$ as weight of link $g \in \mathcal{G}$.
6: Route d along the obtained path, update residual capacities.

Notice that PIBRA uses FT-SPF instead of Dijkstra which is suggested by [KKL00]. When FT-SPF is run without any forbidden turns it is equivalent to Dijkstra. However, using FT-SPF forces the flow to be deviated around the shortest path: this will be useful in Section 5.3.3.

Results for PIBRA are presented in Section 5.4. Still, it is not fundamentally different from the original MIRA, for which more complete results are available in [KKL00].

5.3.3 Layout Design Algorithm for Bus-LSPs

In this section, we present a heuristic algorithm for designing the virtual network when bus-LSPs are allowed. We will be using PIBRA as explained in previous Section 5.3.2 (which uses for the routing stage the FT-SPF introduced in Section 5.3.1), and the graph representation of bus-LSPs detailed in 3.3.

When applying PIBRA to more than one demand, we will always do it in decreasing order of demand volume (see Section 5.3.2). PIBRA is chosen because it can both provide an initial routing of the traffic and re-route a single demand. Because PIBRA uses FT-SPF and not Dijkstra, the flow is able to be inserted in a bus-LSP at a node where there is more routing capacity available. Indeed, flow is deviated from the shortest path when routing

capacities are overloaded along it.

Algorithm 5: Optimization Procedure Initialization

Given: A graph $\mathcal{N} = (\mathcal{V}, \mathcal{G})$
Given: A set $\mathcal{C}$ of capacities on all the links.
Given: A set $\mathcal{R}$ of routing capacities on all nodes.
Given: A traffic matrix (set of demands) $\mathcal{D}$.
1: Transform each link into a bus-LSP of length 1, as described in Section 3.3, using *infinite* capacities on the fork links.
2: Apply PIBRA on all demands, sorted decreasingly by volume (and not by potential, see Section 5.3.2).
3: Create an initially empty list $\mathcal{F}_d$ of forbidden turns for each demand d
4: Create an initially empty queue of forbidden pairs of bus-LSPs to merge $\mathcal{L}$

The idea behind this heuristic algorithm is to:

- route the demands to maximize potential, without taking into account routing constraints. This is why, in Algorithm 5, we use *infinite* capacities on the fork links.
- rearrange layout to reduce amount of routed traffic (that is, traffic which goes from one bus-LSP to another), by deviating flow (i.e., forbidding turns) around heavily loaded nodes

The procedures of rearrangement of the layout can be either:

- Merging two adjacent bus-LSPs into a single one, provided that the merging does not introduce a loop. This reduces routing by allowing some routed demands to flow along a single bus-LSP.
- Cutting a bus-LSP into two smaller ones around a pivot node. This will increase routing if a demand now has to be routed around a node.

In Algorithm 6, function GetLinkPair() is used. This function returns a pair of fork links. One is a (virtual node,node) link of a bus-LSP, the other is a (node,virtual node) link of another bus-LSP. Let us associate for each pair of links g_1, g_2 around v the vector of values (where d is the largest demand routed in v):

- a binary variable indicating there is excess routing (compared to routing capacity of v)
- a binary variable indicating v is egress for the bus-LSP using g_1 and ingress for the bus-LSP using g_2
- the amount of excessive routed traffic due to d, in v, going from g_1 to g_2

Algorithm 6: Bus-LSP Network Optimization Algorithm

```
1: while pair = GetLinkPair() do
2:     Cut all bus-LSPs around nodes where cutting will not increase routing (such as
       at the end of empty segments of bus-LSPs).
3:     Let node be the node in common to both links of pair
4:     Let busPair be the pair of bus-LSPs using the links of pair
5:     if busPair can be merged into a single bus-LSP and pair ∉ 𝓛 then
6:         Merge busPair
7:     else
8:         Let d be the largest volume using successively both links in pair.
9:         Push pair to 𝓕_d
10:        Unroute d, and reroute d using PIBRA
11:        if d cannot be rerouted then
12:            repeat
13:                v = Pop(𝓕_d)
14:                Cut all bus-LSPs using v, around v into 2 bus-LSPs starting and
                   ending at v
15:                Unroute all demands using the cut bus-LSPs
16:            until d is able to be rerouted using PIBRA
17:            for all point v which was popped from 𝓕_d, and which d now uses in its
               route do
18:                if bus-LSPs, that d uses, around v, may be merged and pair ∉ 𝓛
                   then
19:                    merge bus-LSPs
20:                else
21:                    Push the pair of links around v in 𝓛
22:                end if
23:            end for
24:            Sort unrouted demands decreasingly by volume
25:            Route unrouted demands
26:            if Not all unrouted demands were able to be routed then
27:                Exit algorithm;
28:            end if
29:        end if
30:    end if
31: end while
```

- the amount of total routed traffic due to d, in v, going from g_1 to g_2 (that is, volume of d)

The function will return the pair of links which lexicographically maximize this vector. Therefore, because of the structure of this function, we can categorize our algorithm as greedy. Indeed, at every step, we try to get rid of the routed traffic – by either forwarding it directly in the lower layer or deviating the route around the overloaded router – which is the most problematic: obviously this is suboptimal in some cases. However, we will show

results proving that this algorithm functions quite well and fast.

5.4 Numerical Results

We present here results of our heuristic implementation. As there is not, at least to our knowledge, any exact or approximate method to solve the problem presented in Section 5.2.2, we will provide the reader with numerical comparisons evaluating the algorithm.

5.4.1 Testbed

For the exact results, we have used the commercial software CPLEX 9.1 [CPLEX] to solve the problem presented in Section 5.2.2. For our simulations, we have implemented the heuristic presented in Section 5.3.3 in C++[1], using Boost Graph Libraries. This compiled language was chosen because of speed and efficiency.

We have run our tests on a Debian system, running on a 3.4 GHz Pentium with 4GB of RAM.

5.4.2 Results

5.4.2.1 Sample 8-node network

First, we ran the heuristic on the 8-node sample network presented in Figure 5.1. It took just 16 steps (i.e., Algorithm 6 went 16 times through step 1) to perform, and took less than a second. Output was a 2-bus-LSP virtual layout which transported the 5 demands without any routing at all. All demand volumes could be multiplied by 1.42 and *link* capacity constraints would still hold.

5.4.2.2 VTHD network

Next we ran the heuristic on the VTHD [VTHD] network (Figure 2.3)). It is an 11 node and 28 links network, based on a French National Research Network, deployed for the project VTHD (Vraiment Très Haut Débit). Each link has capacity of 1 GBps, and we limited each node's routing power to process also only 1GBps. Under heavy load, this constraint may not be feasible, but we wish to limit the amount of forbidden routed traffic, that is, limit the amount of extra routing equipment the network operator would have to install to satisfy the demands.

We generate traffic demands for all pairs of nodes in the networks by using independent uniform random variables distributed in the interval [0, maxload]. We chose values for

[1]The source code and auxiliary programs are available at `http://perso.enst.fr/~brehon/`

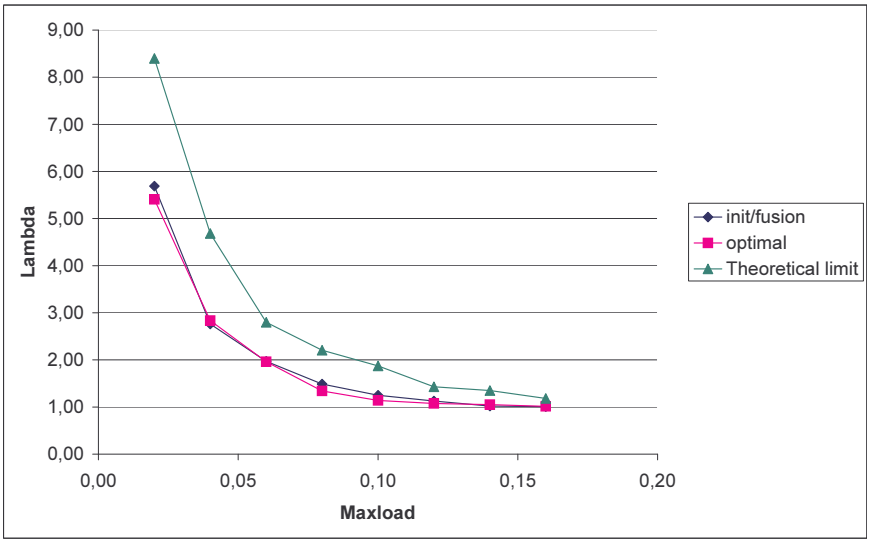


Figure 5.3: Average Value of H (traffic multiplication)

maxload such that the results show the network under light load and under high load. We generated 10 traffic matrixes for each value of maxload (0.02 to 0.2 by increments of 0.02). Results were obtained with a running time of under one minute for every iteration.

The following results were obtained by running the heuristic. We denote by *initial* the results obtained after running the initial PIBRA (Algorithm 5) and by *fusion* the result obtained by applying only bus-LSP fusions to the initial layout: this corresponds to running the heuristic as long as possible without branching in the *else* block (line 7) of Algorithm 6. Finally, we call *optimal* the results obtained by running Algorithm 6 completely.

Results of Figure 5.3 show that PIBRA runs below optimality (it is on average at 71% of optimality), yet it reacts well to the increase of load, and provides a routing of demands in reasonable time and reasonably close to optimality. Theoretical optimum was calculated by an exact resolution using CPLEX with the same traffic conditions, but in a single-layer environment, and allowing flow splitting. *Initial* and *fusion* results are the same since routing is not changed between both steps. Traffic multiplication is calculated without taking into account *routing constraints* (constraints 5.2.6 and 5.2.7 of VTD problem defined in 5.2.2), only *link capacity constraints* are taken into account (constraint 5.2.13). Routing constraints are separated and evaluated in the following results.

Figures 5.4 and 5.5 are the most important ones. These show that our heuristic performs well when considering the initial goal of saving in CAPEX (routing equipments). Our heuristic is able to provide a virtual layout in which the routing power constraints are upheld until *maxload* is equal to 0.1. Total routing is also maintained low, which can translate into some savings by removing extra routing equipments (such as cross layer interfaces). The *average minimal* value is the amount of traffic which is routed at ingress

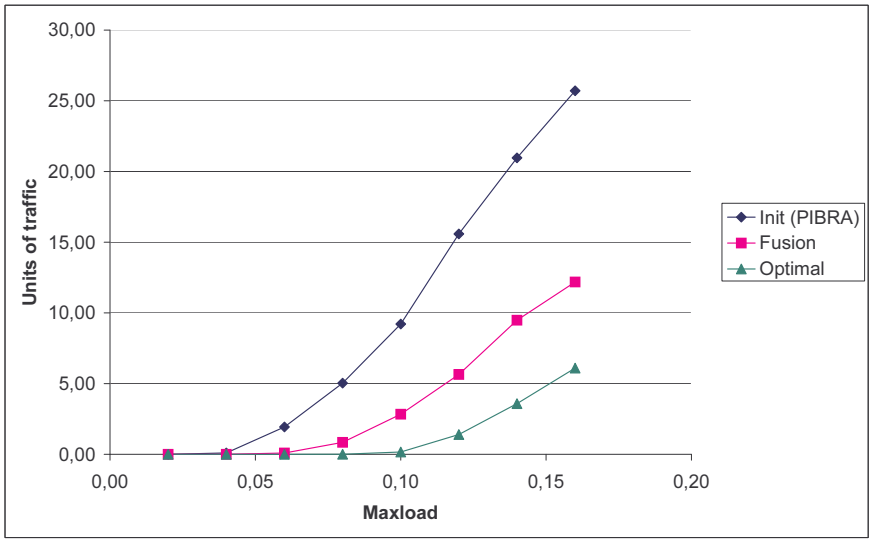


Figure 5.4: Total Forbidden Routed Traffic

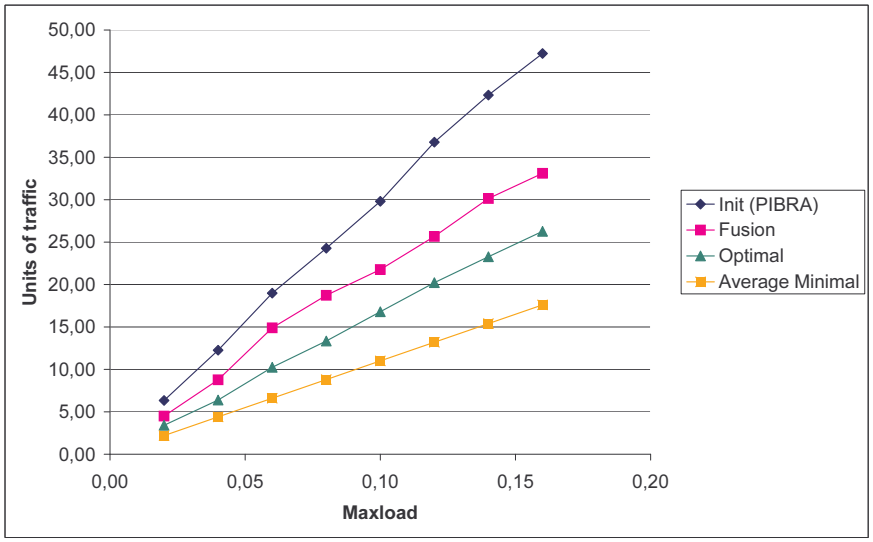


Figure 5.5: Total Amount of Routing in Network

and egress, i.e., the total routing if all traffic could be sent from source to destination in a single bus-LSP.

Results of Figure 5.6 show that OPEX is also quite low. 110 demands are satisfied in this 28-link network setting up only a few bus-LSPs (average is between 6 and 7 bus-LSPs). This very low number means that the network operator's work is quite reduced as maintenance operations concern only a few bus-LSPs.

The delay parameters are studied in figures 5.7 and 5.8 which present respectively the routing delay (average number of upper-layer hops to destination) and the propagation delay (average number of physical links from source to destination). Because of the goal

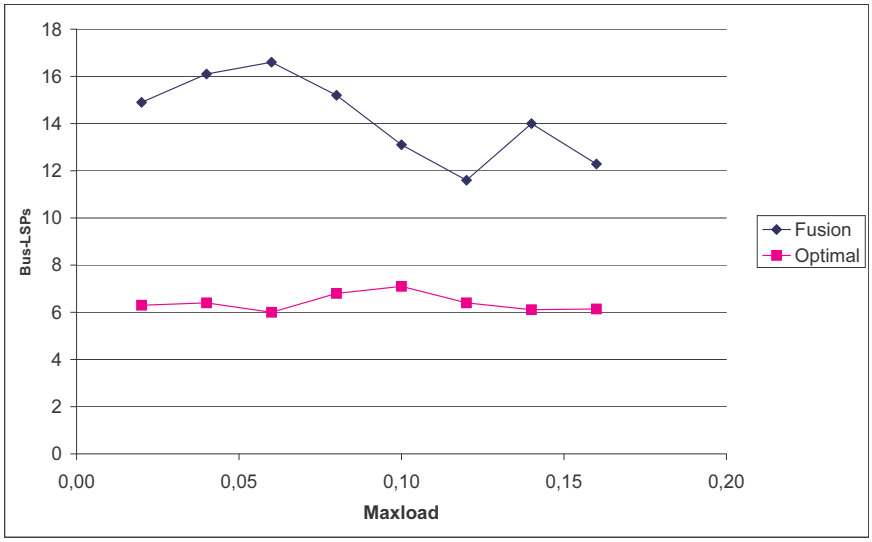


Figure 5.6: Total Number of Bus-LSPs

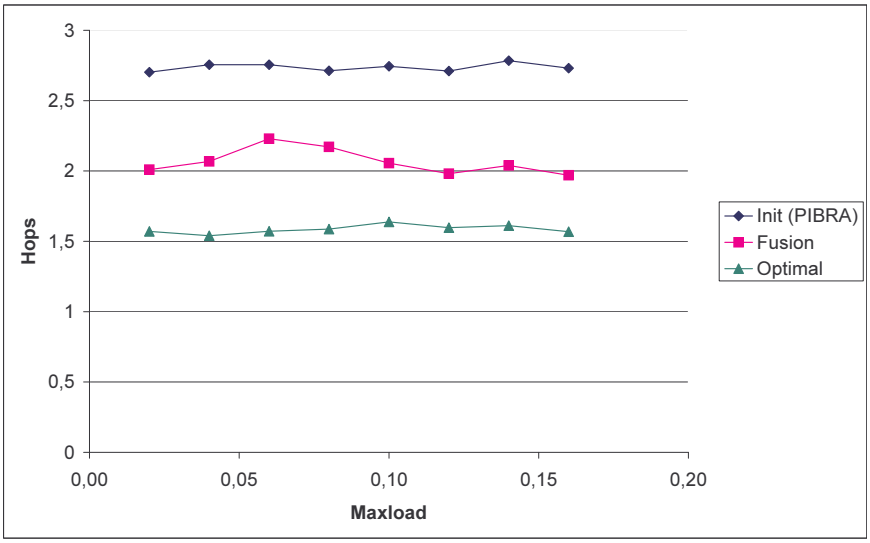


Figure 5.7: Average Number of Hops (routing delay)

of reducing the routing, the routing delay is low as expected. Interestingly however, the overall propagation delay is also quite low, meaning that the average length from source to destination is reduced by our heuristic, yet neither at the cost of traffic multiplier nor at the cost of routing equipment: our heuristic allows the end-to-end delay to be reduced.

Results of Figure 5.9 depict the fact that the average length of bus-LSPs is higher after running the heuristic. This is expected behavior, considering less bus-LSPs are used after running.

Finally, results in Figure 5.10 show the average number of steps our algorithm runs in (the number of times Algorithm 6 goes through step 1). This number is significantly high

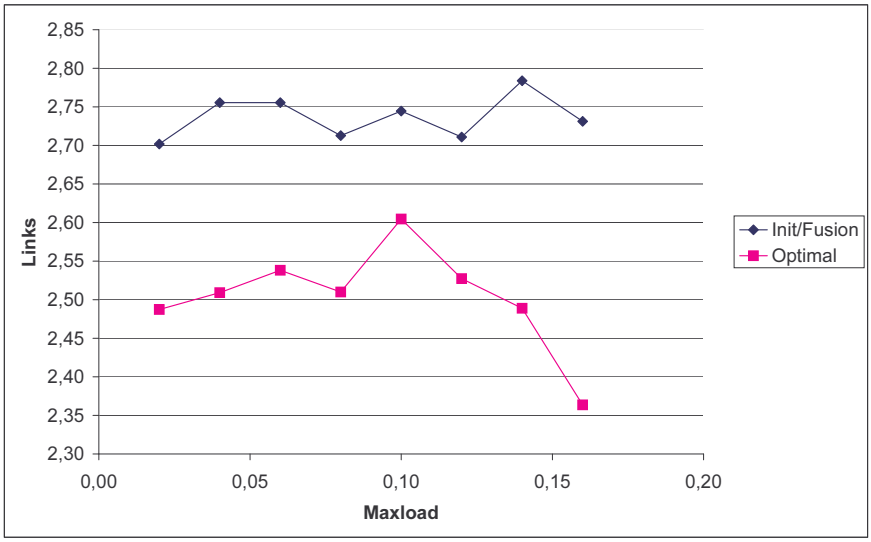


Figure 5.8: Average Number of Links for each Demand (propagation delay)

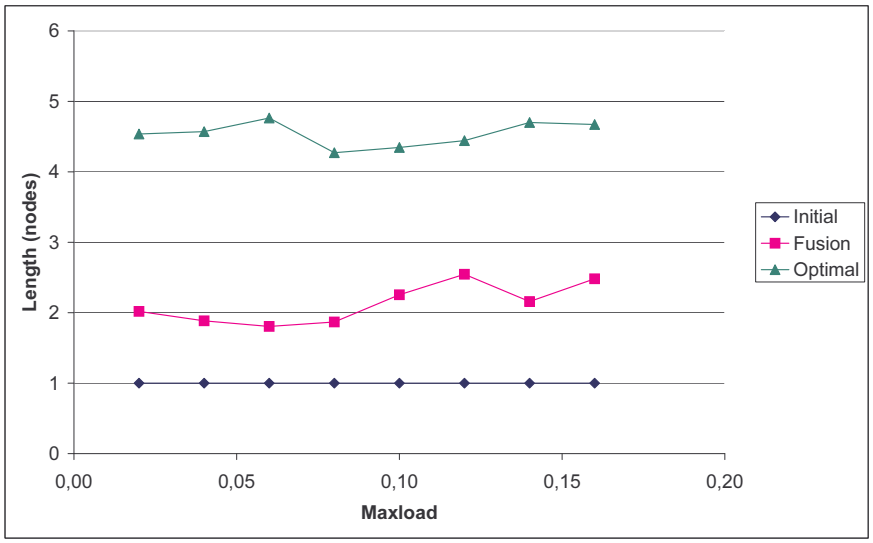


Figure 5.9: Average Bus Length

when running algorithm to end (instead of just stopping at the *fusion* step). However, an entire virtual layout for a given instance of this problem is found within a minute, whereas CPLEX is unable to find a feasible solution (let alone the optimal one) after two weeks of running time when given the formulation of Section 5.2.2 and an instance of the problem for this VTHD network.

5.4.2.3 Italian network

Finally we ran the heuristic on the Italian high-speed network described in [SIL+98] and depicted in Figure 2.4. This is a 21 node- and 72 link-network. Each link was set with a

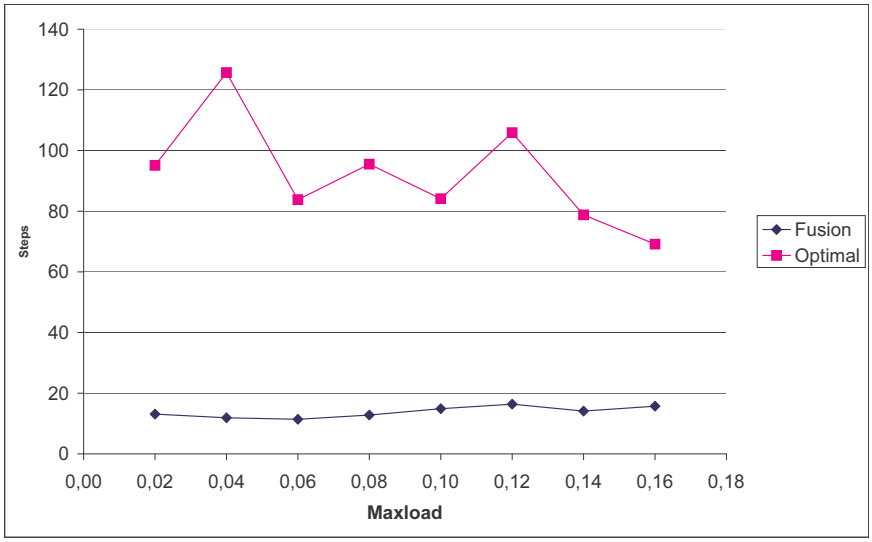


Figure 5.10: Number of Steps to obtain Best Result

capacity of 1 GBps, and we again limited each node's routing power to process also only 1GBps.

We generate traffic demands for all pairs of nodes in the networks by using independent uniform random variables distributed in the interval [0, maxload]. This amounts to 420 demands. We chose values for maxload such that the results show the network under light load and under high load. We generated 10 traffic matrixes for each value of maxload (0.002 to 0.02 by increments of 0.002). Results were obtained with a running time of 65 minutes for each iteration on average. This is not as fast as the previous test, due to the larger network and the more numerous demands.

We present here the results obtained by running the heuristic. We denote by *initial* the results obtained after running the initial PIBRA (algorithm 5) and by *fusion* the result obtained by applying only bus-LSP fusions to the initial layout: this corresponds to running the heuristic as long as possible without branching in the *else* block (line 7) of algorithm 6. Finally, we call *optimal* the results obtained by running algorithm 6 completely.

Results of Figure 5.11 show that the PIBRA runs below optimality: it is on average at 36% of optimality. This can likely be attributed to the fact that the initial routing of the demands uses PIBRA which is not designed for static routing. Theoretical optimal was calculated by an exact resolution using CPLEX with the same traffic conditions, but in a single-layer environment, and allowing flow splitting. The flow splitting may also be partially responsible for theoretical higher values for the multiplication factor. *Initial* and *fusion* results are the same since routing is not changed between both steps. Traffic multiplication is calculated without taking into account *routing constraints* (5.2.6)

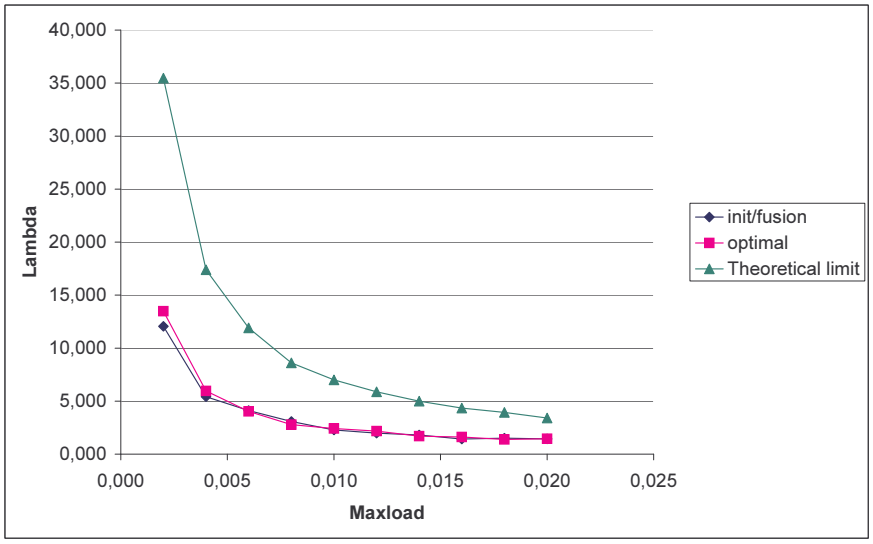


Figure 5.11: Average Value of H (traffic multiplication)

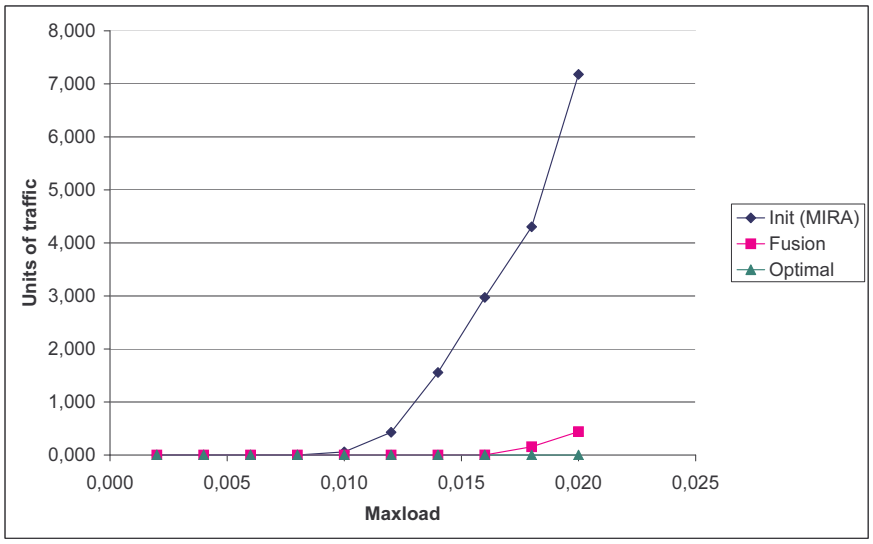


Figure 5.12: Total Forbidden Routed Traffic

and (5.2.7) of VTD problem defined in 5.3.3, only *link capacity constraints* (5.2.13) are taken into account. Routing constraints are separated and evaluated in the following results.

Figures 5.12 and 5.13 are the most important ones. These show that our heuristic performs well when considering the initial goal of saving in CAPEX (routing equipments). Our heuristic is able to provide a virtual layout in which the routing power constraints are *always* upheld. Total routing is also maintained low, which can translate into some savings by removing extra routing equipments (such as cross layer interfaces). The *average minimal* value is the amount of traffic which is routed at ingress and egress, i.e., the total

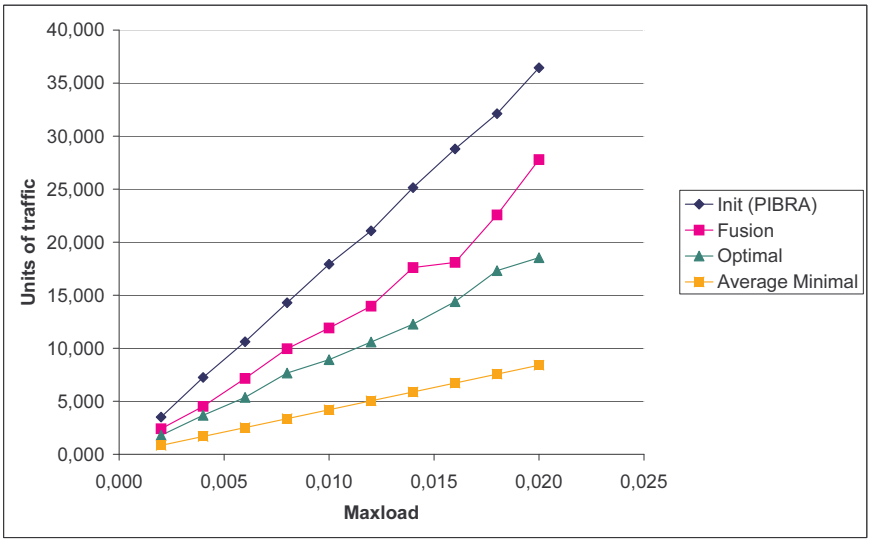


Figure 5.13: Total Amount of Routing in Network

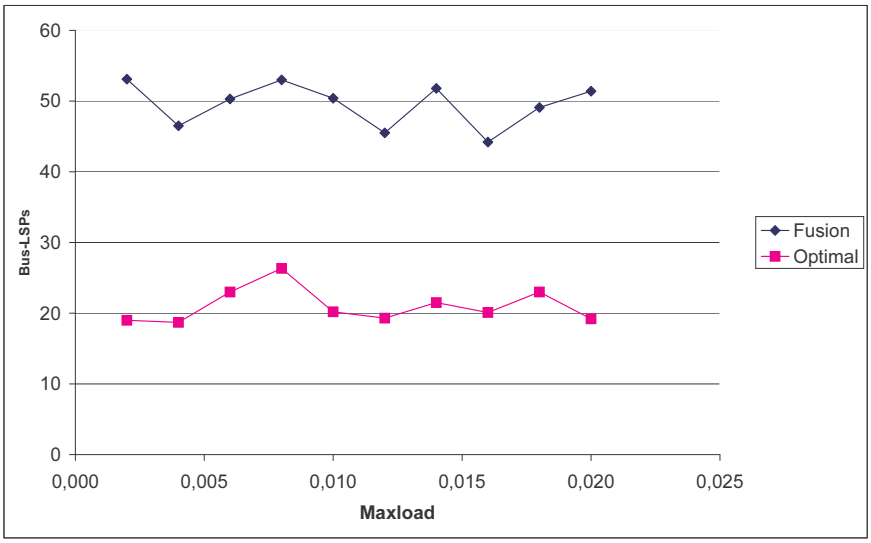


Figure 5.14: Total Number of Bus-LSPs

routing if all traffic could be sent from source to destination in a single bus-LSP.

Results of Figure 5.14 show that OPEX is also quite low. 420 demands are satisfied in this 72-link network setting up only a few bus-LSPs (between 18 and 27 bus-LSPs). This very low number means that the network operator's work is quite reduced as maintenance operations concern only a few bus-LSPs.

The delay parameters are studied in Figures 5.15 and 5.16 which present respectively the routing delay (average number of upper-layer hops to destination) and the propagation delay (average number of physical links from source to destination). Because of the goal of reducing the routing, the routing delay is low as expected. As in the previous section

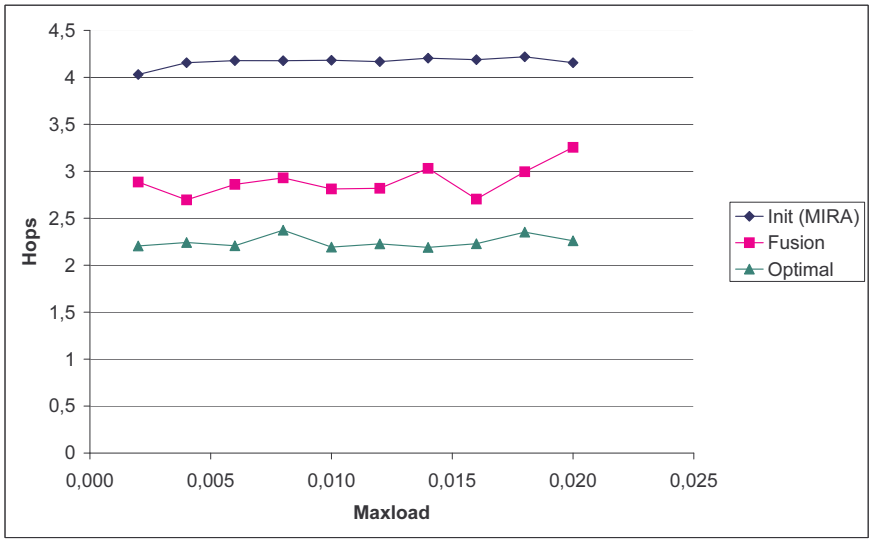


Figure 5.15: Average Number of Hops (routing delay)

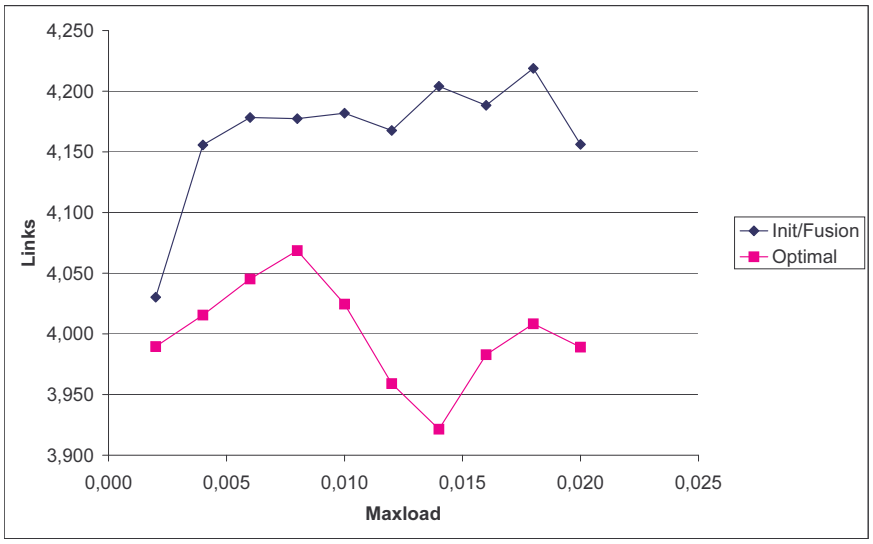


Figure 5.16: Average Number of Links for each Demand (propagation delay)

however, the overall propagation delay is also quite low, meaning that the average length from source to destination is reduced by our heuristic, yet neither at the cost of traffic multiplier nor at the cost of routing equipment.

Results of Figure 5.17 depict the fact that the average length of bus-LSPs is higher after running the heuristic. This is expected behavior, considering less bus-LSPs are used after running.

Finally, results in Figure 5.18 show the average number of steps our algorithm runs in (the number of times algorithm 6 goes through step 1). The number of steps is even higher than with the VTHD network, as the solution space explored is larger.

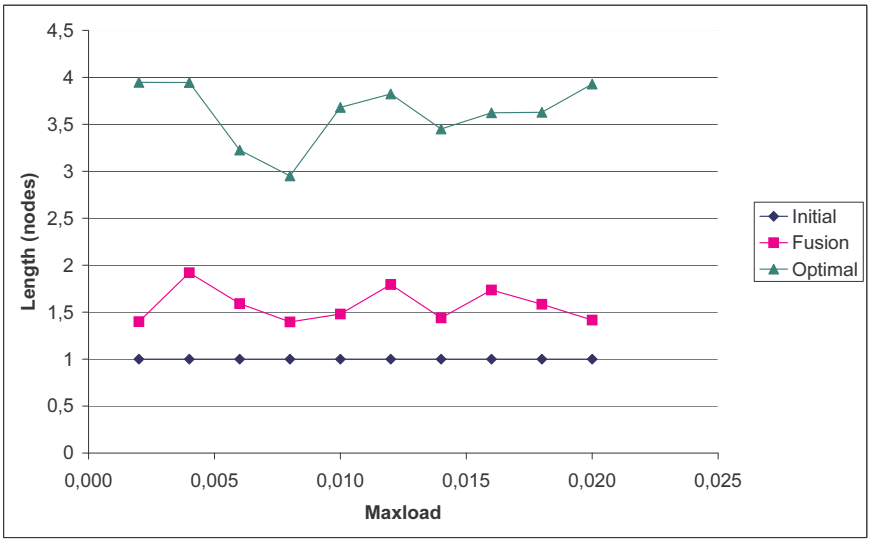


Figure 5.17: Average Bus Length

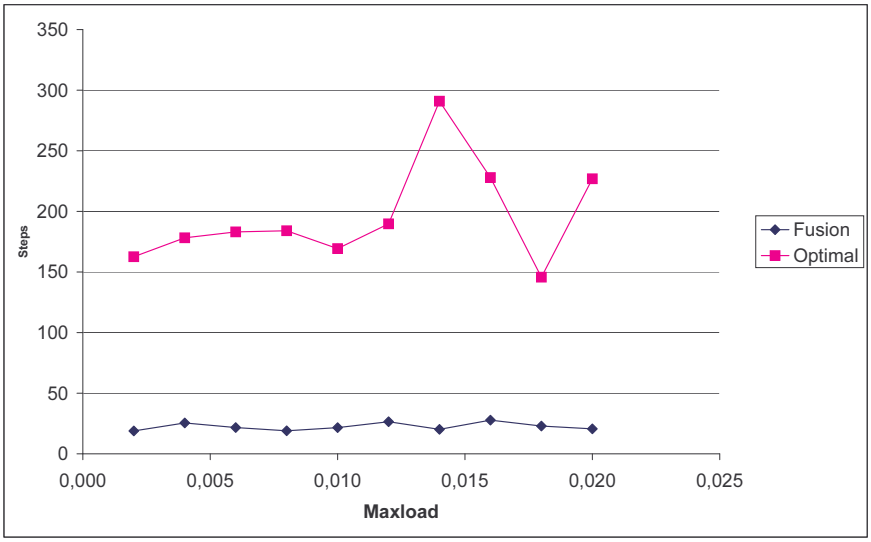


Figure 5.18: Number of Steps to obtain Best Result

5.5 Heuristic Algorithm Improvements

In this section, we consider various sets of improvements which may be made to the proposed heuristic algorithm. Though the results we obtained were quite good, some limitations were encountered. In particular, the heuristic did not scale very well in terms of traffic matrix multiplication. Also, a link can only carry a single bus-LSP in this version, which means ”overlapping” bus-LSPs are not possible.

5.5.1 Improving the initial routing

In our algorithm, we wished that the routing procedure be the same at all stages, using our PIBRA. This is good for implementation simplicity, and performs quite well. However, since this heuristic is meant for offline optimization, it is possible to consider different approaches.

In particular, one could use an exact resolution maximizing the load multiplication, obtained by an ILP solver such as the commercial CPLEX [CPLEX] solver for instance. Since initially, the virtual topology matches exactly the physical topology, the problem can be seen, at least at the beginning, as a single layer problem. Additionally, the fusion process starts with an initial routing of the flows which does not necessarily obey the maximum amount of routing per node constraints (5.2.6) and (5.2.7). Therefore, formulating the maximization of the multiplier is quite easy, and can even be done in a link-node formulation such as described in [PM04]. These problems, which are solved to obtain the *theoretical limit* in the previous section 5.4, are extremely easy and fast to solve. Flow separation should be avoided however, this adds some complexity to the formulation since it means the problem must use integer variables which slow the calculation process further.

However this approach needs further testing to be validated. The approach with PIBRA actually lexicographically maximizes all multipliers, not simply the first coefficient of the proportional fairness. It is not clear whether simply maximizing the first coefficient will actually provide better overall results, though initial intuition may suggest so.

Also, this approach does make the heuristic a little more complicated, since it must first solve the ILP, either using a solver or by some other approach. It must then, if this is not provided by the solving function, correlate the maximum values of flows on each link to the demand paths so as to obtain routing.

5.5.2 Allowing overlapping bus-LSPs

Some scenarios are not possible with the current heuristic. Consider the bus-LSPs depicted in figure 5.19. A link can carry two units of traffic between A and B, and B can carry one unit to C and one unit to D. If demands are from A to C and from A to D, there will necessarily be routing in B using our heuristic, since the fusion process will entirely merge link AB to either BC or BD.

However, this can be avoided by allowing the link AB to carry two separate bus-LSPs. This can be done brutally, by simply considering AB as two one-unit links, but in large networks, this solution is not scalable. Therefore, it would be interesting to design an algorithm which takes this into account by design, at the merging phase for instance. This can probably be done by considering a bus-LSP segment as having a used capacity (the sum of all demands already assigned to it) and a potential capacity (the size it could grow

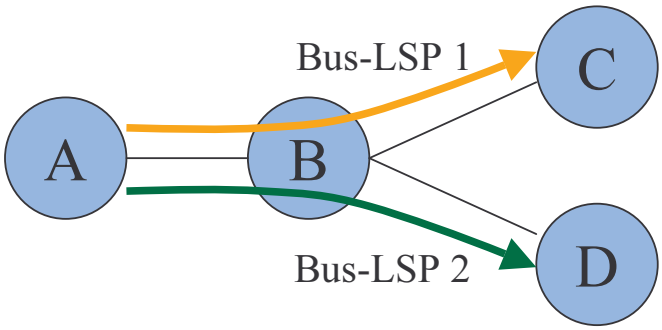


Figure 5.19: 2 Overlapping un-mergeable bus-LSPs

to without the sum of all used capacities exceeding the link nominal capacity). With these two variables, the heuristic might be adapted for flow separation.

Some problems are not solved yet however: when and where does one decide to open a parallel bus-LSP? What happens when two bus-LSPs sharing a segment are used to route a single demand (the potential value is not correct)?

This is an improvement which may be very interesting, but a solution to the above stated problems has not been found yet.

5.5.3 Minor modification of Algorithm 6

Algorithm 6, when tested, performed better as presented above. However, it is possible to modify it slightly and get results which are a little different. On average the following modification gives lesser results (when tested on the examples of Section 5.4), however, on some individual instances it allows the algorithm to perform better. This performance measure is based on the traffic multiplication factor which is found. Note that this is a very marginal increase or decrease of performance in all cases anyway. We don't suggest the use of this modification, however, it can be considered in some cases.

Function *GetLinkPair()* can sometimes return a *pair* such that *busPair* contains two bus-LSPs with more than one node in common. In this case, normal execution would branch into statement of line 8. This will try to reroute demands running around the common node of *pair*. Alternative behavior would be to simply add, before line 5, an instruction saying that when a *busPair* has two or more nodes in common, *pair* should be added to $\mathcal{L}$. This alternate behavior provides results which are, on average and on the networks of Section 5.4, 2% below the standard traffic multiplication value(H). On some individual experiments, however, one can note an increase of up to 5% of traffic multiplication H.

5.5.4 Dynamic use of the heuristics

As was mentioned in Section 5.3, our approach is not ideal for dynamic environments. Since the optimization process may reroute many demands, traffic disruption is likely to happen. However, by considering that the established demands may not be rerouted, one can introduce in the network a new set of demands and apply our heuristic approach with minor modifications. Removing a set of demands can also be done easily in a similar fashion. However, this approach requires further study to evaluate how the solution given evolves compared to optimality and to a complete static re-optimization of the new traffic matrix.

5.6 Concluding Remarks

In this chapter we have analyzed benefits from introducing bus-LSPs in a multi-layer GMPLS network. We have shown that this concept allows for reducing both OPEX and CAPEX and we have evaluated quantitatively this gain for a sample network topology. For this purpose, we have formulated and analyzed a mixed-integer linear program which is a realistic model.

We have then proposed a Virtual Topology Design heuristic, based on a new type of Constrained Shortest Path First Algorithm and a variant of the Minimum Interference Routing Algorithm. Results provided by the heuristic have shown that the layouts provided by our heuristic are good: they allow for significant traffic scaling and important routing reduction, that is, they are meaningful CAPEX-wise. They also are relatively simple which directly translates into a reduction of OPEX. Finally, these results are obtained in a very short computing time, proving they are usable in dimensioning of real networks within the management process.

While the benefits to be gained from the bus-LSP have been studied and quantified, it is still necessary to specify the mechanisms to actually use this structure in GMPLS-controlled networks. The following chapter therefore presents extensions to both the routing and signaling protocols of GMPLS for bus-LSP support.

Chapter 6

Extensions to GMPLS Routing and Signaling Protocols for Bus-LSPs

When integrating a new object such as the bus-LSP into a GMPLS architecture, extensions to the standard protocols are needed for inter-operability. Two of the most important protocols of GMPLS are OSPF-TE [RFC3630] and RSVP-TE [RFC3473], used respectively for the exchange of routing information [RFC4202] and for signaling the establishment, modification, deletion and protection of LSPs in the data plane.

6.1 Routing

6.1.1 OSPF-TE: the GMPLS routing protocol

OSPF-TE is a protocol based on OSPF [RFC2328]. OSPF was designed to allow *link state* routing. Historically, routing started off as *distance vector* routing, with protocols such as RIP [RFC1058]. In such a routing protocol, each router exchanges with its neighbor only distance vectors to other routers and attached hosts. A router which receives such a distance vector updates its own routing tables and forward any modified information to its own neighbors. This is basically a practical implementation of the Dijkstra's algorithm [Dij59]. However, limitations quickly occurred with the use of such protocols, namely convergence time, big distance vector exchanges (routing tables) and a limited maximal size of the routing area, due to what is called "horizon splitting" [RFC1058]. *Link state* routing was introduced to face these limitations.

Link-state routing protocols are based on the contrary on the flooding in the network of announcements of the existence of links (Link State Advertisement, LSA) and routers

(Router Advertisement, RA). Each router is then responsible for calculating a shortest path to any destination. The algorithm used to calculate this shortest path is not standardized, but Dijkstra can be used for example.

However, OSPF was not sufficient to do traffic engineered routing, since the calculations of routes did not take into account multiple criterions and only calculate shortest paths. It was therefore necessary to introduce a protocol able to manage traffic-engineered links, which have multiple characteristics, such as the available and used bandwidths, administrative status colors, etc. OSPF-TE allows this by introducing into an OSPF routing area new type of announcements: *opaque* LSAs and RAs [RFC2370] which carry TE information. In OSPF-TE, two databases are used: one for "regular" links, and one for TE-links, the TE-database. The TE-database takes into account all links which may be used to transport traffic, including LSPs announced as FAs.

6.1.2 Bus-LSPs related requirements of OSPF-TE

For bus-LSPs to be integrated in GMPLS networks, it is important that the IGP (Internal Gateway Protocol) be able to announce these objects so routers know that they should include them in their TE-databases for path calculations; it is also important that the node add-drop capabilities be announced in the network.

However, this must not be done at the cost of other functionalities. The first requirement is therefore backward compatibility: new objects introduced in the routing protocol (and the same will be true of the signaling protocol) should not disrupt the protocol for nodes working with older versions of the protocol. Of course, such nodes may not be able to use all of the functionalities offered by bus-LSPs.

The bus-LSP object is complex, since it creates many Forwarding Adjacencies (FAs) but the capacity of each of these FAs depends on the other FAs. This complexity may translate into an increased number of messages exchanged in the routing plane. This number of messages will have to be monitored and kept as low as possible, for scalability issues. One of the ways to achieve this is by infering attributes, given that many of the characteristics of the bus-LSP segments are linked.

6.1.3 Various representations

Three different ways of representing bus-LSPs in the routing plane were considered. These representations are necessary in order to perform route calculations over graph structures. These representations are:

- The *linear* representation
- The *virtual node* representation

- The *half mesh* representation

In the following, we represent a bus-FA connecting an arbitrary number of nodes denoted $R_1, R_2, \ldots, R_N$. In Figure 6.1, the upper layer is represented by a traditional cylinder with arrows (usually representing IP routers) while the underlying switch is represented by a plain box. This may represent two linked equipments or a single multi-layer equipment, with packet and WDM capabilities for example.

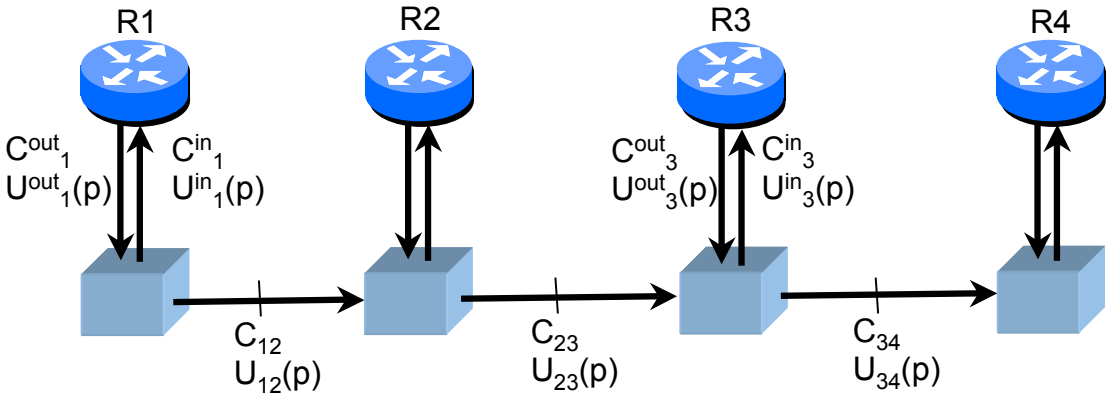


Figure 6.1: Parameters of a bus-LSP (with 4 nodes)

For node n, let C_n^{out} and C_n^{in} denote the capacity available between the upper layer and the lower layer and between the lower layer and the upper layer, respectively. Other parameters will be denoted with the same in/out convention, in particular $\{U_n(p)\}_p$ which will represent the set of unreserved bandwidth (for the different priorities p). At the lower layer, the switches n and $n+1$ are connected by a specific TE link. Let $C_{n,n+1}$ denote its capacity, and $\{U_{n,n+1}(p)\}_p$ denote the set of unreserved bandwidths for different priorities. The unidirectional nature of a bus-LSP implies that:

$$\begin{aligned} C_N^{out} &= 0 \\ C_1^{in} &= 0 \\ U_N^{out}(p) &= 0 \ \forall p \\ U_1^{in}(p) &= 0 \ \forall p \end{aligned}$$

Furthermore, flow conservation implies the following recursion formula, which holds for all prority p :

$$C_{1,2} - U_{1,2}(p) = \left(C_1^{out} - U_1^{out}(p)\right) \quad (6.1.1)$$

$$\begin{aligned} C_{n,n+1} - U_{n,n+1}(p) &= C_{n-1,n} - U_{n-1,n}(p) + \left(C_n^{out} - U_n^{out}(p)\right) \\ &\quad - \left(C_n^{in} - U_n^{in}(p)\right) \end{aligned} \quad (6.1.2)$$

$$C_{N-1,N} - U_{N-1,N}(p) = \left(C_N^{in} - U_N^{in}(p)\right) \quad (6.1.3)$$

From these equations, we can see that all the proposed variables are not needed. We shall see that some proposed routing representations will use this property, omitting the sets $\{U_{n,n+1}(p)\}_p$ for instance, as they can be easily derived.

Note that with this simple model, we focused on parameters related to bandwidth. In this section the cases of other metrics and attributes (SRRG, etc.) are not considered.

6.1.3.1 Linear Representation

This model aims at representing the bus-FA with a set of N specific TE links (where the bus-LSP connects N nodes $R_1, R_2, \ldots, R_N$). Each TE link will summarize the properties of a segment of the bus-LSP. We shall see that this model does not allow non-compatible equipments to use bus-LSPs (yet, bus-LSPs will not disrupt the routing plane). However, the linear decomposition allows us to reach scalability and the lowest overhead in terms of the number of LSAs generated, particularly in case of changes in the bus capacity: whenever a TE-LSP is set-up from node i toward node j across the bus-FA, only two LSAs are generated.

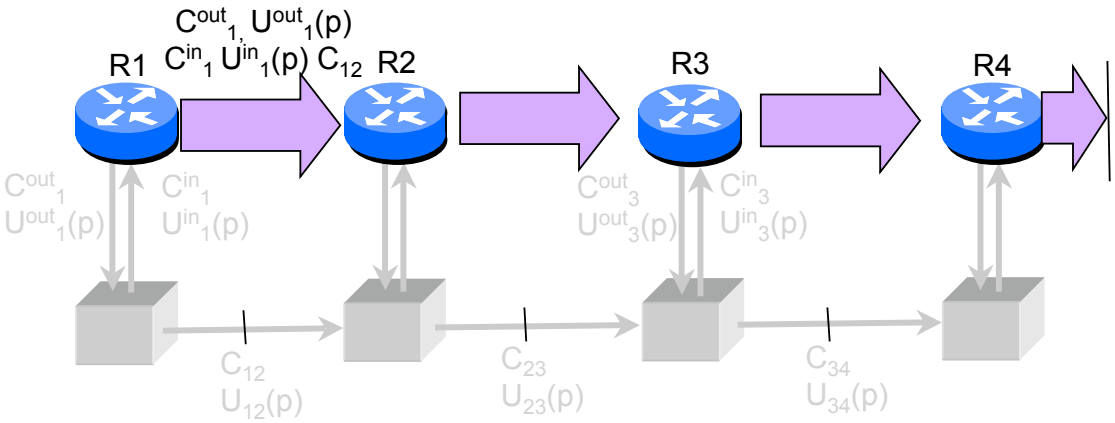


Figure 6.2: Linear representation of a bus-FA (with 4 nodes)

As shown in Figure 6.2, the proposed scheme uses N link state messages, one per node in the bus-FA. The link state generated by node n will summarize the different properties of the segment between node n and $n+1$, hiding the underlying lower layer. Of course, the link state of the last node N has a degenerate form, since there is no next hop in this case. We use the same semantic however, with appropriate parametric adaptations (e.g., some parameters will be null).

Using the previous notations, one can see that each LSA will need to define:

- C_n^{out},
- $U_n^{out}(p)$,

- C_n^{in},
- $U_n^{in}(p)$,
- $C_{n,n+1}$,
- and $U_{n,n+1}(p)$.

Using the recursion formula (6.1.1) introduced above, we do not need to specify $U_{n,n+1}(p)$. The first message is specific with $C_1^{in} = 0$ and $U_1^{in}(p) = 0, \forall p$. The last message is also specific, as there is no next hop. It is used to specify C_N^{in} and U_N^{in} - by convention, we set all other parameters to null.

One of the interesting properties of this approach is that there is no coordination requirements between nodes of the bus-FA. The different link states can be generated easily and locally by each node as the parameters are locally available (e.g. U_n^{in}, U_n^{out} for node n).

Deducing $U_{n,n+1}$ from the recursion formula (6.1.1), allows to reduce the number of messages in case the bus-FA capacities change. Imagine that some TE-LSP is set-up between node i and node j and that this LSP is nested in the bus-FA. The unreserved bandwidth parameters will thus change for $U_i^{out}(p)$, $U_{i,i+1}(p)$, ..., $U_j^{in}(p)$ (for the priority levels below the TE-LSP priority). With this representation however, only two messages are needed, one at each end-point of the client TE-LSP. The triggered link state at node i will allow one to get the new set of parameters $U_i^{out}(p)$ and the one at node j will announce the new values $U_j^{in}(p)$. All intermediate values $U_{i,i+1}$ can be inferred easily by any node after reception of these two messages.

We will briefly discuss here how OSPF-TE could be used to announce the representation described above. We do not intend to cover all technical details and give a full specification, but rather highlight the technical issues and how they might be solved.

First, a new TE link type should be defined: for instance, link-type="bus-FA". Then new TLVs will be required to encode all parameters: C_n^{out}, $U_n^{out}(p)$, C_n^{in}, $U_n^{in}(p)$, and $C_{n,n+1}$.

The most difficult aspect of the proposed scheme is that all link states composing the bus-LSP are independently generated by each node. This implies that without any proper extension, there is no mean for a node to associate the different segments of the bus-LSP with each other. The Link-ID is not sufficient: for example, the Link-ID could be set to the last hop router ID, but several bus-LSPs could be terminated at this very node. We propose a new TLV, the "Association ID". Note that this can be the same value as in the RSVP-TE attribute with the same name proposed in 6.2.4.

Since this representation requires new link types to be defined, network nodes unaware of these new types will be unable to use bus-FAs at all. This does not disrupt the routing protocol however, but rather limits the use of bus-FAs to network nodes which are able to understand the corresponding extensions to the routing protocol. However, in a multilayer

network (overlay model) where the routing is determined by a PCE (Path Computation Element) for instance, non-bus-FA-compatible nodes may use the bus-FA transparently to nest the LSPs for which they are ingress. Indeed, only the PCE needs to know how to handle bus-FAs, so in such an architecture, compatibility is in fact full, and all nodes may benefit from the establishment of bus-FAs in the network.

6.1.3.2 Virtual Node Representation

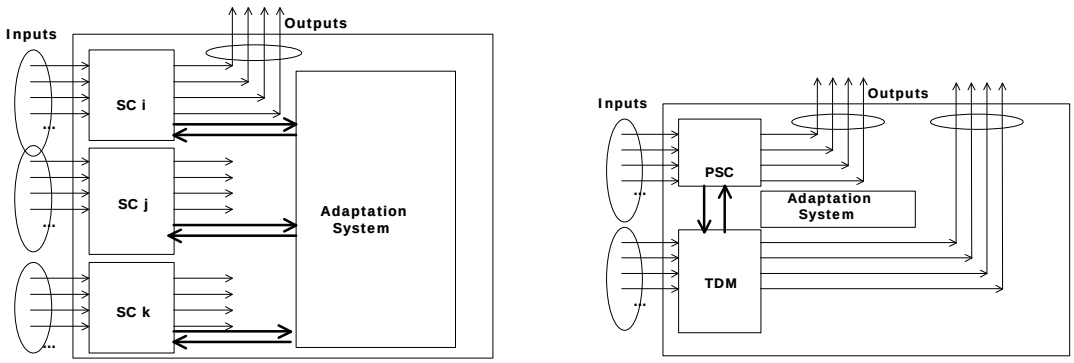
This approach is motivated by the increasing number of GMPLS hybrid nodes such as the MLXC introduced in Section 3.1.1. Hybrid nodes are GMPLS-enabled nodes which announce a diversity of switching capabilities, with the possibility of terminating and/or adapting the switching capabilities as defined in the GMPLS hierarchy [RFC4206]. This approach defines a general means of representing such hybrid nodes, the MLXC being a specific case thereof. This representation is based on the graph equivalent of bus-LSPs introduced in Section 3.3.

The proposed approach was also designed for full backward compatibility with legacy OSPF-TE. Nodes that are not bus-FA-aware will see the bus-FA as a set of nodes and TE links in several regions (switching capabilities): this will allow the setup of LSPs on these TE links. The main drawbacks are that this solution induces more overhead and raises some technical issues.

According to [SPR+06]: "A hybrid node can terminate links with different switching capabilities terminating on the same interface. So, it advertises at least one TE Link containing more than one ISCDs with different ISC values. For example, a node comprising of PSC and TDM links, which are interconnected via internal links. The external interfaces connected to the node have both PSC and TDM capability.". Figure 6.3(a) illustrates a representation of the data plane part of such a hybrid node : the node can be considered as a set of switching elements, one per supported switching capability (region), plus a subsystem that fulfills the adaptation capabilities required at the node, here, PSC and TDM adaptation. This adaptation subsystem comprises:

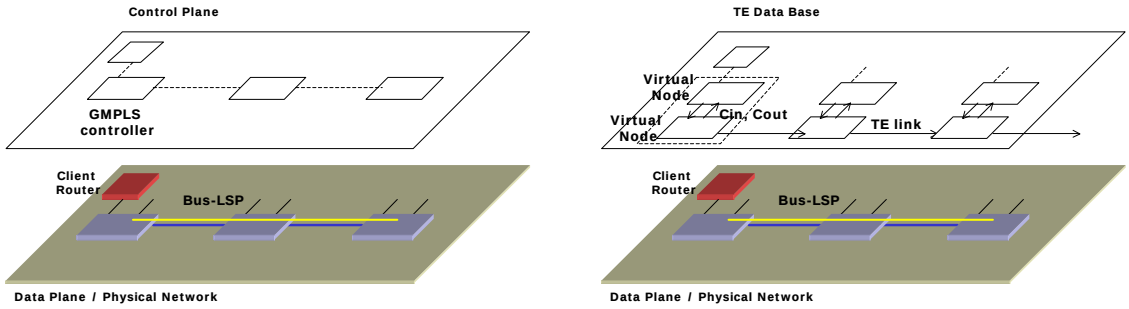
- Switching capability termination.
- Data plane adaptation and encapsulation.
- In general, any inter-working functions required for the adaptation of lower and higher levels of the GMPLS hierarchy.

The bus-FA node is a particular case of a hybrid node, with native upper-layer (here, PSC) and lower-layer (here, TDM) interfaces, and an adaptation system that allows adding or extracting packets from/to the lower-layer connection, as shown in Figure 6.3(b). The



(a) Generic Hybrid Node (b) Bus-FA node as a Hybrid Node

Figure 6.3: Bus-FA node as a Hybrid Node



(a) Data and control planes (b) TE database

Figure 6.4: TE nodes

adaptation system is simplified in order to represent only the capacities C^{in} and C^{out} which need to be managed.

The virtual node approach is represented in Figure 6.4. In order to represent in the TE database the different switching elements, the adaptation subsystems and their associated TE properties (e.g. bandwidth parameters), each switching element (e.g. packet switching matrix, OCX matrix, etc.) is represented as a node and the adaptation capabilities, modeled as internal links, are represented as TE links with specific TE properties. The TE database is thus extended in order to have a detailed description of the internal architecture of hybrid nodes in the simple form of nodes and links.

This approach breaks the established GMPLS routing paradigm where only TE links are announced and these are attribute extensions of physical links in the data plane, with the exception of Forwarding Adjacencies and bundled links [RFC3473]. In our case, we have to announce ”virtual” nodes (one per switching element) and ”virtual” TE links (internal

adaptation capabilities).

As represented in Figure 6.4(b), the bus-FA is represented by a set of independent links:

- TE links between the two virtual nodes : 2 TE links, one for each direction (upstream/downstream) with [PSC,TDM] switching capability. These TE links will announce parameters $C_i^{out}, \{U_i^{out}(p)\}$ and $C_i^{in}, \{U_i^{in}(p)\}$, respectively.
- TE links between consecutive hops in the bus-LSP (between the lower layer virtual nodes). They are unidirectional links with the switching capability of the lower layer. These links will announce parameters $C_{i,i+1}$ and $\{U_{i,i+1}(p)\}$.

With this virtual-node-based representation, the bus-FA is not represented as a specific (multiple access) TE-link but as a set of independent TE links at different regions. For bus-FA-enabled nodes, the distinct TE-links can be associated within a single compound object, in order to better represent the bus-FA. This can be achieved using a specific bus-FA Association-ID as introduced in Section 6.1.3.1 for instance. Note that the nodes that are not bus-FA-capable will not be able to understand this extension and will not see the bus-FA but independent TE links. This insures full backward compatibility as an LSP can still be set-up through these TE links and virtual nodes (normal LSP set-up in a multi-region context).

A bus-LSP connecting N nodes requires $2N$ virtual nodes and $3N-1$ TE links (to be compared with N nodes and $N-1$ TE links in the linear representation). Furthermore, in case a new LSP is nested within the bus-FA, a larger set of LSAs is required to announce the changes of the TE properties, since the parameters $\{U_{n,n+1}\}$ are explicitly announced and not inferred as in the previous case.

6.1.3.3 Half Mesh Representation

In Figure 6.5, a last option is illustrated. This boils down to simply representing the bus-FA with a half-mesh between all nodes involved. In this figure, $C_{12} = \min\left(C_1^{out}, C_2^{in}, C_{12}\right)$, $C_{13} = \min\left(C_1^{out}, C_3^{in}, \min(C_{12}, C_{23})\right) = \min\left(C_1^{out}, C_3^{in}, C_{12}, C_{23}\right)$ and so forth : $C_{ij} = \min\left(C_{out}^{i}, C_{in}^{j}, C_{i,i+1}, C_{i,i+2}, \ldots C_{ij}\right)$.

The same equalities hold for the variables $\{U_.\}$:

$$U_{ij} = \min\left(U_i^{out}, U_j^{in}, U_{i,i+1}, U_{i,i+2}, \ldots U_{ij}\right).$$

The rationale for such a representation is to have full backward compatibility: nodes with no bus-FA extensions are able to "see" the bus-FA (as standard TE links) and thus to set-up LSP inside this bus-FA. Each TE link can be set to accurately represent the bus-FA. For instance it is possible to use the same TE metric between any nodes of the bus-LSP

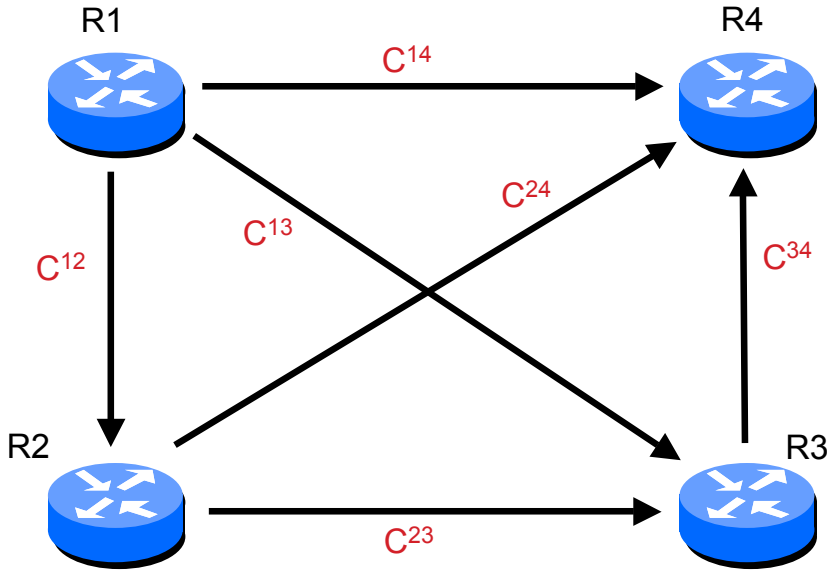


Figure 6.5: Half Mesh Representation

(the same TE metric for all the TE links of the half mesh) and not a cumulative metric at each individual segment. Such a metric policy is not possible with the virtual node representation in Section 6.1.3.2.

The main drawback of this approach is its very poor scalability due to the large number of TE LSAs that such a representation will generate, which can be approximated by $\frac{N(N-1)}{2}$ links for a bus-LSP with N nodes. Furthermore, each time there is a change in the bus-FA, such as a decrease of the available bandwidth due to a new nested LSP for instance, the whole mesh may potentially be impacted, therefore requiring a large number of TE LSA updates.

Finally, this representation poses some synchronization issues between nodes of the bus-LSP: in Figure 6.5 for instance, let us imagine than a significant change in available bandwidth occurs between nodes R3 and R4 (a nested LSP is set-up between R3 and R4 using the bus-FA). This will cause the generation of a TE LSA update by R3 for link R3-R4. R1 and R2 may also have to update their TE LSA due to the recursive formulas $U_{ij} = \min\left(U_i^{out}, U_j^{in}, U_{i,i+1}, U_{i,i+2}, \ldots, U_{ij}\right)$. How these nodes will be aware of this change is an issue for which we did not find an acceptable solution yet. If we do not assume any specific synchronization procedure between nodes of the bus-LSP, R1 and R2 will thus have to evaluate whether any received LSA update impacts their own set of TE links. This requires using, again, a specific field to associate the different TE links with each other, such as the "Association ID" of the previous solutions in sections 6.1.3.1 and 6.1.3.2. This association however induces a "cascade" effect (generated LSA update that in turn trigger other LSA updates) that would strongly penalize both convergence

and routing scalability.

6.1.3.4 Suggested representation

Different routing representations for the bus-FAs have been proposed and discussed previously. The implementation of these representations have been analyzed.

- The linear representation appears as the most efficient and scalable solution. It allows reaching the minimal overhead in term of generated LSAs; however it has minimal backward compatibility. It is however adapted in multi-area networks with PCEs.

- The virtual node representation offers very good backward compatibility. However, the overhead generated by this solution is slightly higher with respect to the previous case, yet it remains linear in the number of nodes.

- The mesh representation fully insures backward compatibility. However, this solution raises very strong scalability issues (routing messages polynomial in the number of nodes)and synchronization requirements between nodes.

The virtual node representation, similar to the graph representation of Section 3.3, appears therefore as the best compromise between backward compatibility and scalability.

6.1.4 Flooding node capabilities

While representing bus-FAs is very important to allow them to nest LSPs, a fully automated bus-LSP-based network needs a way to find out which nodes are actually Add/Drop-capable. This will allow the network to establish bus-LSPs connecting various Add/Drop capable nodes.

In [VRY+06], an extension to OSPF-TE is introduced which allows to flood TE node capabilities. For instance, it is suggested in the point-to-multipoint context to announce which nodes are able to be branch nodes (i.e., they replicate traffic on multiple branches). We propose to announce the Add/Drop capabilities in the same way. [VRY+06] defined 5 node capabilities, we suggest an additional sixth and seventh capabilities. This is coherent with [VRY+06]: "Note that new capability bits may be added in the future if required". We define an "A bit" which is used to signal the fact that a node is able to perform the Add function, and a "D bit" for the Drop function.

6.2 Signaling

6.2.1 Introduction

Bus-LSPs being new objects, some modifications are necessary in the control plane as well, that is, to the RSVP-TE [RFC3209, RFC3473] protocol which is the de-facto standard for the GMPLS control plane. RSVP-TE being made to manipulate point-to-point LSPs, some operations were never previously needed. Indeed, if a FA needed to be terminated, the associated LSP was closed; if a FA was needed, a corresponding FA was opened, and so on.

In contrast, bus-FAs offer multiple adjacencies within a single object, therefore it is very conceivable that part of these adjacencies need to be modified while preserving the others. A set of new operations was defined to tackle new issues raised by bus-FAs:

- Introduce the notion of add-drop node in the control messages
- Manage the nodes belonging to a bus-FA
  - Activate/Deactivate add-drop functionalities from a node on a bus-LSP
  - Add or remove a node at the head of the bus-LSP
  - Add or remove a node at the tail of the bus-LSP
- Sequentially merge two or more running bus-LSPs

In the following sections, we will therefore address these issues and explain how the various operations can be done.

6.2.2 Managing Add-Drop Points

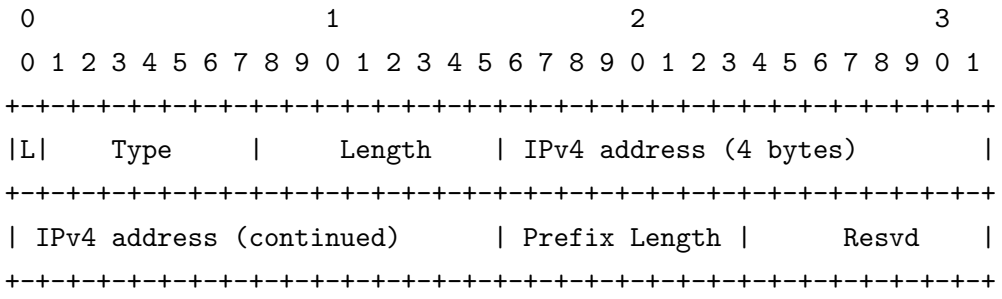
With no modifications to the existing protocols, it is impossible to even establish a bus-FA since nothing exists to signal the fact that a node on the path of the bus-LSP should be ready to execute add-drop operations. To this effect, it is important to include in the signaling the fact that a node will or not be an add-drop node of the bus-LSP.

6.2.2.1 The ERO object and subobjects

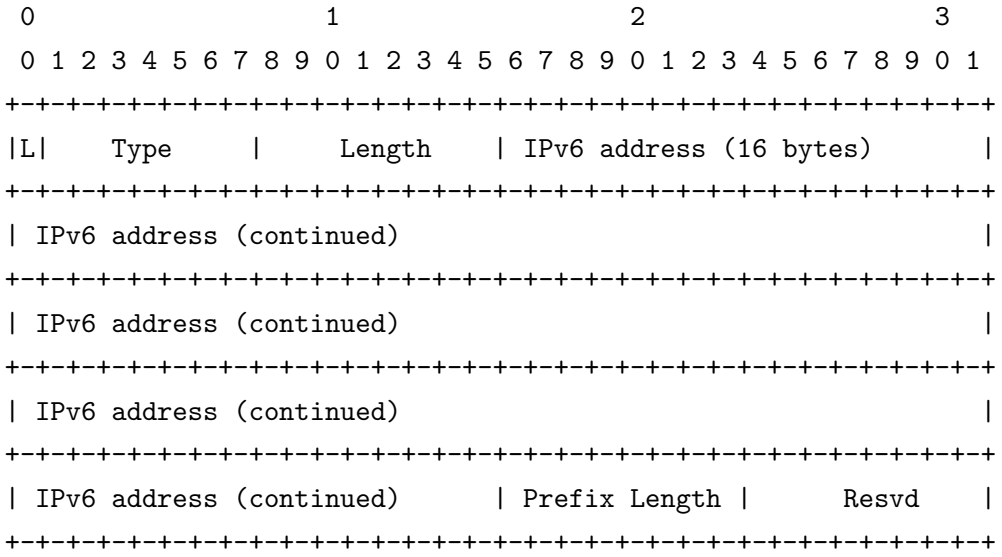
According to [RFC3209], the Explicit Route Object (ERO) can be used in a PATH message to indicate which nodes will constitute the path followed by the LSP. These nodes do not have to constitute the entirety of the path, that is only some intermediate nodes are mentioned, the others will be determined hop-by-hop by routing techniques as the PATH message will progress: this is called *loose routing*. These nodes can be addressed by an

IPv4 address, an IPv6 address, IPv4 or IPv6 prefixes or an AS number. For bus-LSPs, AS numbers and IP prefixes may not be considered as add-drop nodes, the objects used in those cases are therefore not modified. In the IP address cases, according to the RFC, the format of an ERO subobject is as follows:

For IPv4 addresses:



For IPv6 addresses:



The only difference here is the length of the address. The L bit is used to indicate whether the node must be reached by *strict* routing or by *loose* routing. If the node is marked as *strict* (L bit set to zero), then this node must be a direct neighbor of the preceding node in the ERO. Otherwise, a path will be evaluated to reach the node.

The Type field is fixed to 0x01 for IPv4 and 0x02 for IPv6 addresses.

The Length field indicates the total length of the subobject in bytes and must be set to 8 for IPv4 and to 20 for IPv6 addresses.

The Address field is a standard IPv4/IPv6 address.

The Prefix Length, if different from the length of an IPv4/IPv6 address, indicates that the address must be considered as an IP prefix, the length of which is specified by this field, in bits. If the node is a router, and the Address field contains the address of this router, then this Prefix Length must be set to 32 for IPv4 and to 256 for IPv6 addresses.

Finally the Resvd section will be disregarded by routers and is reserved for future use.

6.2.2.2 Encoding the Add-Drop Functionality

We propose to use the 2 last bits of the Resvd field to encode the fact that a node will be performing add-drop operations on the LSP (Add-Drop status, or AD status). We will denote these bits as the AD (add-drop) flags.

For non-compatible routers receiving the PATH message, this field will be disregarded (this means that these nodes cannot perform add-drop operations of course) and the message will be treated normally. This allows non-bus-LSP aware routers to be intermediate nodes on a bus-LSP, though they will not be an add-drop node.

For compatible routers, we propose to use the following values for the AD flags:

- 00 (0x00): no add drop operations
- 01(0x01): add only operations
- 10 (0x02): drop only operations
- 11 (0x03): add and drop operations.

Decrementing the counter of the packet (see 3.2.2) will be performed by the nodes performing the drop operations, since nodes which only add traffic do not have to read this counter. This must be taken into account when setting the counter by the ingress node and the nodes adding packets in the bus.

6.2.3 Bus-LSP Manipulation Methods Without Extensions

In this section, we present methods which allow to add and remove add and drop functionality to nodes which are part of the bus-LSP. These methods make use of the AD flags as defined in the above section 6.2.2.2.

Of course, one can change the AD status of various nodes, modify the endpoints, etc. by simply destroying the existing bus-LSP and opening a new one. This is called the *break-before-make* method. However it makes for a long traffic disruption, therefore it is sub-optimal. One can also do what is called *double booking* which consists in opening a second LSP while the first is running, then destroying the first LSP. However this leads to resource waste and cannot always be performed, in particular when the network is heavily

loaded and therefore cannot afford to double-book the resource reservation for a single LSP. The GMPLS framework provides tools to do the *make-before-break* technique. We will show in this section how these tools can be used for bus-LSPs.

6.2.3.1 RSVP-TE Identifiers

So as to present the ways bus-LSPs may be manipulated by the control plane, some of the inner workings of RSVP-TE must be explained, in particular the way an LSP is identified and signaled in the network.

The most important object of the RSVP-TE messages is called the SESSION object, which carries, among others, three fields:

- IPv4 (or IPv6) tunnel end point address (egress address)
- Tunnel ID: An identifier that remains constant over the life of the tunnel. This represents the connectivity between the endpoints, and it must stay constant even when the tunnel is rerouted.
- Extended Tunnel ID: An identifier used in the SESSION that remains constant over the life of the tunnel. Normally set to all zeros. Ingress nodes that wish to narrow the scope of a SESSION to the ingressegress pair MAY place their IPv4 (or IPv6) address here as a globally unique identifier.

Another very important object of RSVP-TE is the SENDER_TEMPLATE (for PATH messages) and the FILTER_SPEC (for RESV messages). This object carries, among others, these fields:

- IPv4 (or IPv6) tunnel sender address (ingress address)
- LSP ID: An identifier that can be changed to allow a sender to share resources with itself. It identifies the path and TE characteristics of the LSP. The LSP ID is used to differentiate LSPs that belong to the same LSP Tunnel (as identified by its Tunnel ID).

6.2.3.2 Adding and removing an intermediate AD point

The goal here is to activate an add drop point along the path which was previously forwarding the node without performing any add-drop functionality; or deactivate such a functionality.

When an ingress node with an existing path wants to modify the AD status of a node, two methods are available, each of which have their benefits.

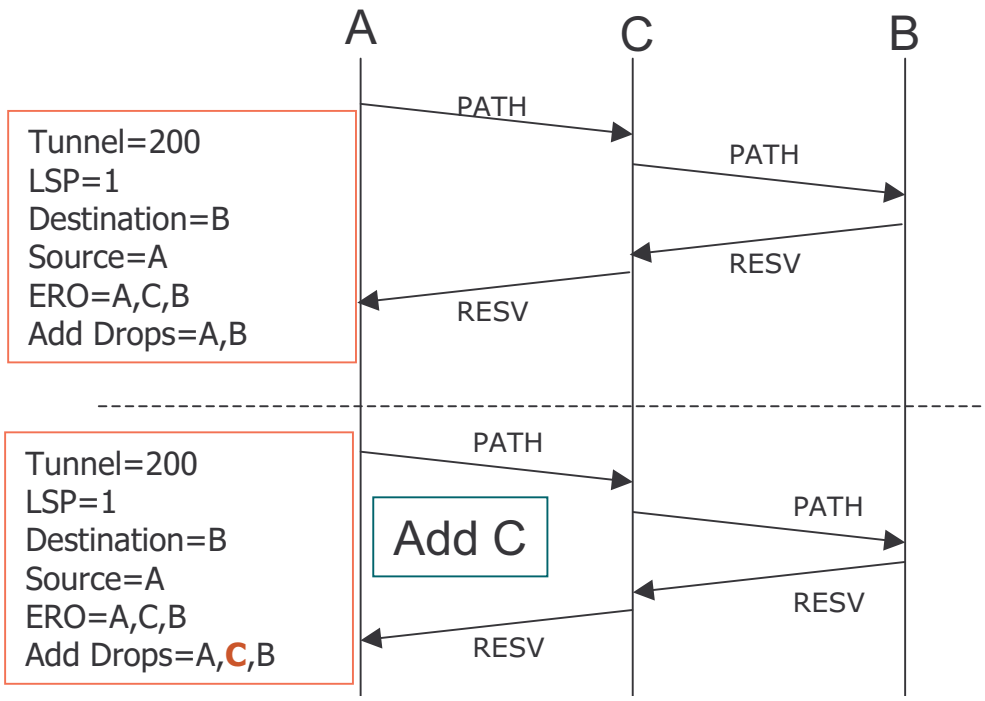


Figure 6.6: First method, adding an AD point

The first and simplest method, illustrated in Figure 6.6, is to simply refresh the Path messages along the bus-LSP and changing the AD statuses in the ERO. This must be initiated by the head-end. However, this will of course change the value that each node has to set on the counter indicating in the packet which node has to drop it. Therefore any node A seeing a modification of the AD statuses in the refreshed PATH messages should stop sending packets to a downstream node B which is such that another node C whose AD status is modified is in-between A and B (nodes are in this order: A ... C ... B). The packet emission will be able to resume as soon as the RRO indicates that node C (and all nodes in a similar situation) has also changed its AD status.

The second method, illustrated in Figure 6.7, may be used to guarantee that the traffic is not interrupted. It simply relies on applying a make-before-break procedure such as described in [RFC3209]. The head-end will form a new Path message as follows. The existing SESSION object is used unchanged. The ingress node picks a new LSP_ID to form a new SENDER_TEMPLATE. It creates an EXPLICIT_ROUTE object for the route, which contains in particular the updated AD statuses. The new PATH message is sent. The ingress node refreshes both the old and new path messages.

The egress node responds with a Resv message with an SE flow descriptor formatted as:

```
<FLOWSPEC><old_FILTER_SPEC><old_LABEL_OBJECT><new_FILTER_SPEC>
<new_LABEL_OBJECT>
```

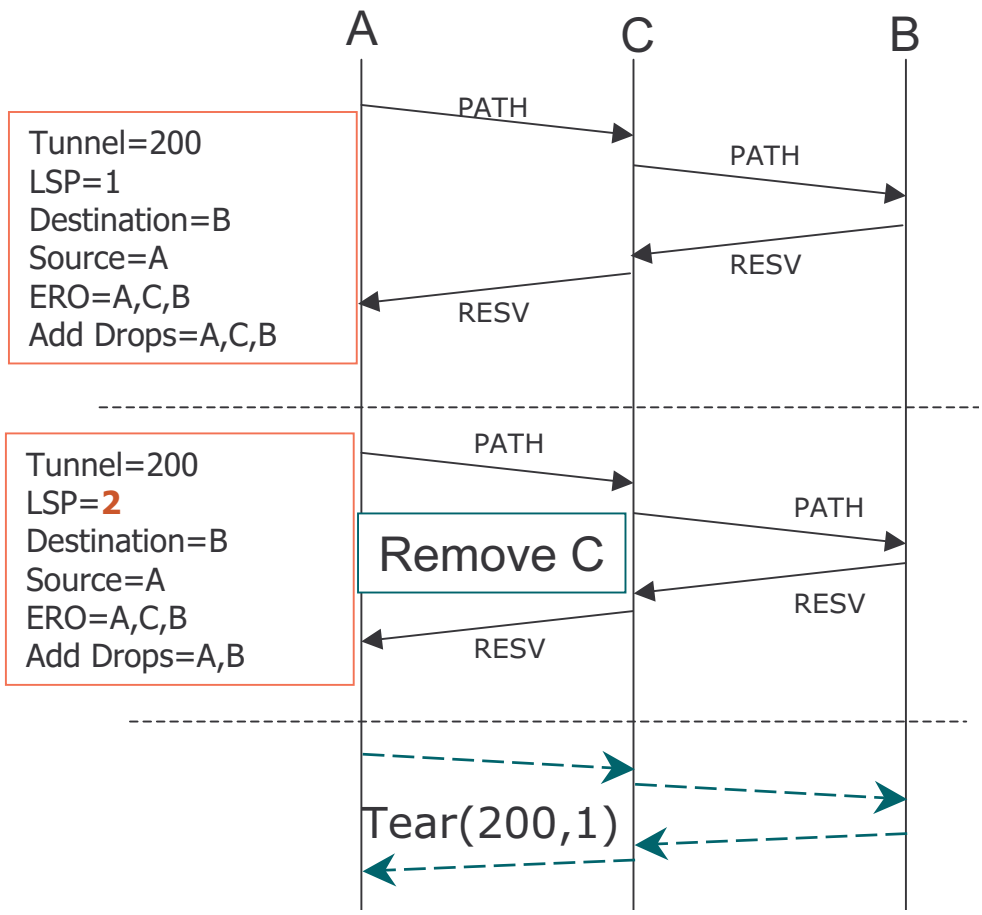


Figure 6.7: Method 2, removing an AD point

The FLOWSPEC and FILTER_SPEC are RSVP-TE objects which are used to specify the TE characteristics of the LSP. The LABEL object contains the used label for the LSP. Though two LSP IDs are being used, the RSVP-TE protocol is built to allow the intermediate nodes to do resource reservation sharing instead of double-booking, based on the fact that the Tunnel ID and the SESSION object are the same.

This message is sent upstream, hop-by-hop to the ingress node. When the ingress node receives the Resv Message(s), it may begin using the new route. It should send a PathTear message for the old route, which is a message which will erase the RSVP-TE bindings related to the old LSP ID in the nodes.

However, since this second method consists in signaling a second LSP, the labels will have to be changed from the first LSP (old AD statuses) to the new LSP (new AD statuses): this is a requirement of MPLS [RFC3209]. In some cases, this is not an option, for example if the label is the only available wavelength or SDH time slot. In that case, one must apply the first method. Otherwise the second method is a good way to modify the AD statuses

without disrupting the traffic at all, since the use of 2 LSPs guarantees two different sets of labels and therefore, the counters associated with each one will be easy to distinguish.

6.2.3.3 Adding and removing the ingress point

RSVP-TE is a destination oriented protocol, and can do resource reservation sharing in the following way: a node receiving a reservation request for an LSP with a Tunnel ID *and* a SESSION object for which it already has a reservation knows resource sharing must be done (as in the previous section 6.2.3.2). Therefore adding or removing an ingress node can be done by the new to-be ingress node which simply signals a new bus-LSP using the same SESSION object and the same Tunnel ID. Once the new bus-LSP is setup, the destruction of the previous one can be triggered by sending a Path Tear message. This is simply a standard make-before-break rerouting procedure as defined in [RFC3209] and an example of which is available in the previous section (6.2.3.2, second method). This procedure does not introduce any counter-related problems, as two distinct bus-LSPs are established, and though they share the resource reservations, they use two different labels and therefore counters are clearly identified and separated.

6.2.3.4 Extending the bus-LSP (adding a node beyond the egress)

Because RSVP-TE is a destination-oriented protocol due to its inheritance of RSVP, it is not made for changing the egress. It is however possible to achieve the same objective by applying a series of elementary operations and taking advantage of the nesting [RFC4206], [RFC3471] or stitching [AKV06] capabilities available in GMPLS. Since nesting is possible only where a hierarchy of labels is available, we will demonstrate the technique using stitching procedures; however, using nesting would be equivalent in terms of the number of opening and tearing down of LSPs. Nesting is simpler to put into application, and does not need the first step in which the LSP is prepared for stitching.

Let's assume a bus-LSP A-B-C-D is established and we wish to extend it to reach E. This is illustrated in Figure 6.8. The first step is to ask for stitching over the existing LSP. This is done by setting in the PATH message the "LSP Stitching Desired" bit, which is the 5th bit in the Attributes Flags TLV of the LSP_ATTRIBUTES object [AKV06]. Once the egress node receives this request, it informs the LSP's head-end that it is ready to perform stitching by setting in the RESV message the "LSP segment stitching ready", which is the 5th bit of the Flags field of the RRO Attributes sub-object. This finishes the first step of preparing the LSP for stitching.

The next procedure consists in using the [A,B,C,D] LSP as a S-LSP [AKV06]. This means it is used as a segment to support another LSP. In this case, the supported LSP is [A,D,E]. The new LSP headend, A will setup this LSP by emitting a PATH message, sent to the

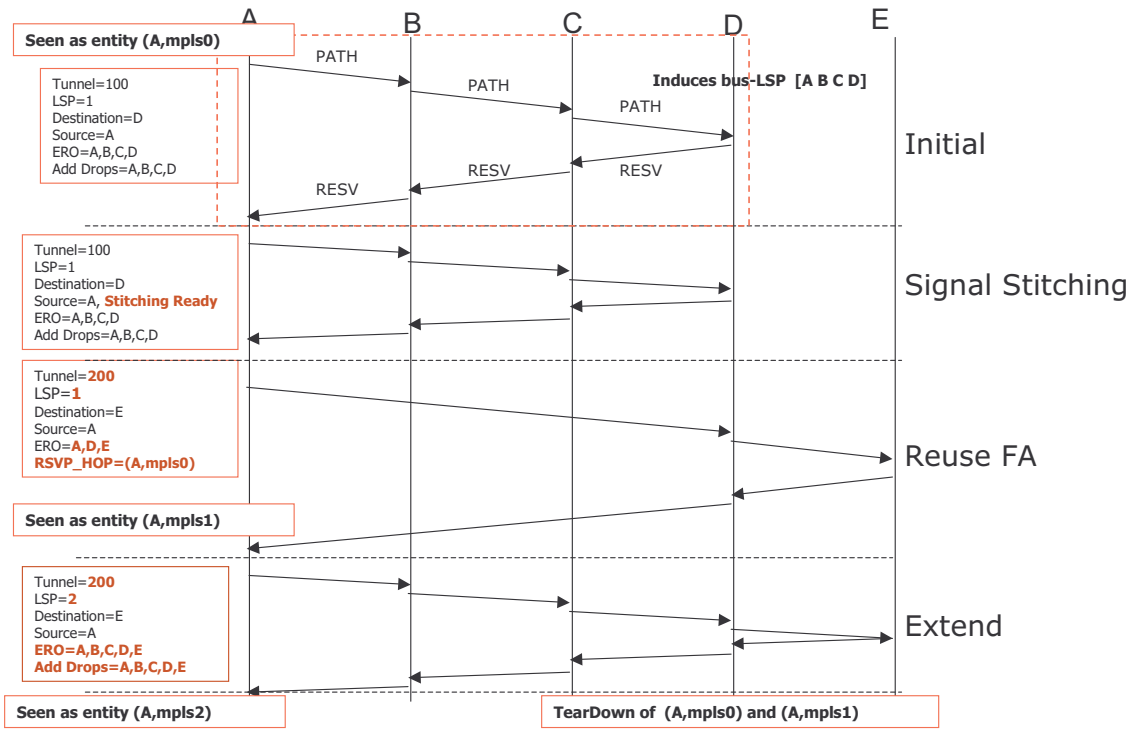


Figure 6.8: Extending a bus-LSP

extremity of the stitched LSP: D. This PATH message uses a new SESSION object (since the destination changes) and informs D that the LSP is stitched to [A,B,C,D] by the fact the RSVP_HOP points to the LSP [A,B,C,D] used as supporting tunnel. D will then itself forward the PATH message to E. The RESV messages will follow the reverse path. This way, [A,B,C,D] has been stitched with [D,E]. At this stage, when a packet is sent from A to E, using [A,D,E], it is sent to D via the [A,B,C,D] LSP, using the corresponding labels. Since the LSP was opened using stitching, D knows all packets arriving via the [A,B,C,D] LSP (this information is obtained by the correspondance between label and LSP), must be immediately forwarded onto the [D,E] segment. However, there exists no [A,B,C,D,E] LSP yet, counters are not applicable, and especially, nodes B and C have no indication that they are stitched to E. Therefore the procedure cannot be marked as complete yet.

head-end A must now reroute the [A,D,E] LSP to explicitly go through [A,B,C,D,E], then only will the bus-LSP [A,B,C,D,E] exist as such. Therefore it sends a new PATH message to open the [A,B,C,D,E] LSP, with the same identifiers (SESSION object and tunnel sender address) and a different LSP ID. At this point, traffic can be switched from [A,B,C,D] to [A,B,C,D,E]. Node A then performs standard make-before-break procedure as explained as in [RFC3209] and in previous sections 6.2.3.2. The head-end can then finally destroy (PathTear message) the old LSP [A,B,C,D] and the temporary [A,D,E] LSP.

In the case where nesting is possible, the above procedure is then followed exactly to the

only exception that the [A,B,C,D] LSP is not signaled for stitching. This does not change the result, the only precaution to be taken is to make sure that [A,B,C,D] is not used to nest any other LSP in which case the final teardown can only be done when all nested LSPs have been transferred to use different nesting LSPs (this includes the bus-LSP [A,B,C,D,E] of course).

Note that draft [BBPF06] introduces a way to change the egress of a bus-LSP, however, this is only in the case of a failure scenario, and it cannot be adapted because its main drawback is its lack to forbid definitive rerouting to the new egress. This extension only allows temporary egress change, and only when a link is actually unavailable. Also note that this extension is incompatible with ingress modification.

6.2.3.5 Removing the egress point

The process for removing the egress point of a bus-LSP without any extension to the signaling protocol is basically the reverse operation of the previous solution for extending the bus-LSP. However, it is more complex, because we need to establish LSPs without actually reserving resources for them.

Let's assume an [A,B,C,D,E] LSP is setup, and we wish to reduce it to [A,B,C,D], as shown in Figure 6.9. The first step consists in opening an [A,B,C,D] LSP but without committing the resource reservation. This is done by exploiting the PROTECTION object. We open [A,B,C,D] as a new LSP, different from [A,B,C,D,E] (new Tunnel ID, new LSP ID, new destination address). The PATH message needs to ask for a bandwidth reservation equal to the one of [A,B,C,D,E] (this is done in the SENDER_TEMPLATE). However, to prevent the resources to actually be reserved, the bits P (for *protecting*) and S (for *secondary*) are set to 1. This means the LSP is opened to be a protecting LSP (as defined in [LRP06]). The ASSOCIATION_ID of the ASSOCIATION object is not set. This means this LSP will be used to protect an LSP which is not setup yet. This new [A,B,C,D] LSP will therefore be ready to replace another LSP, but for now, the resources are not actually taken off the network. The [A,B,C,D] LSP is also signaled to use stitching (as in the previous section 6.2.3.4, using "LSP Stitching Desired" and "LSP segment stitching ready" mechanisms).

The next step consists in opening a protecting LSP [A,D,E] using the [A,B,C,D] S-LSP [AKV06], which should "protect" the [A,B,C,D,E] LSP. It will not be used for protection of course, but once again we are exploiting some of the available features of RSVP-TE to achieve our goal. Therefore, a PATH message is sent from A to D, then to E to signal a new LSP. This LSP uses the same Tunnel ID, destination address and sender address as [A,B,C,D,E]. The PATH message informs D that the LSP is stitched to [A,B,C,D] by the fact the RSVP_HOP points to the LSP [A,B,C,D] used as supporting tunnel.The ASSOCIATION object specifies in the ASSOCIATION_ID the LSP ID of LSP [A,B,C,D,E]: this

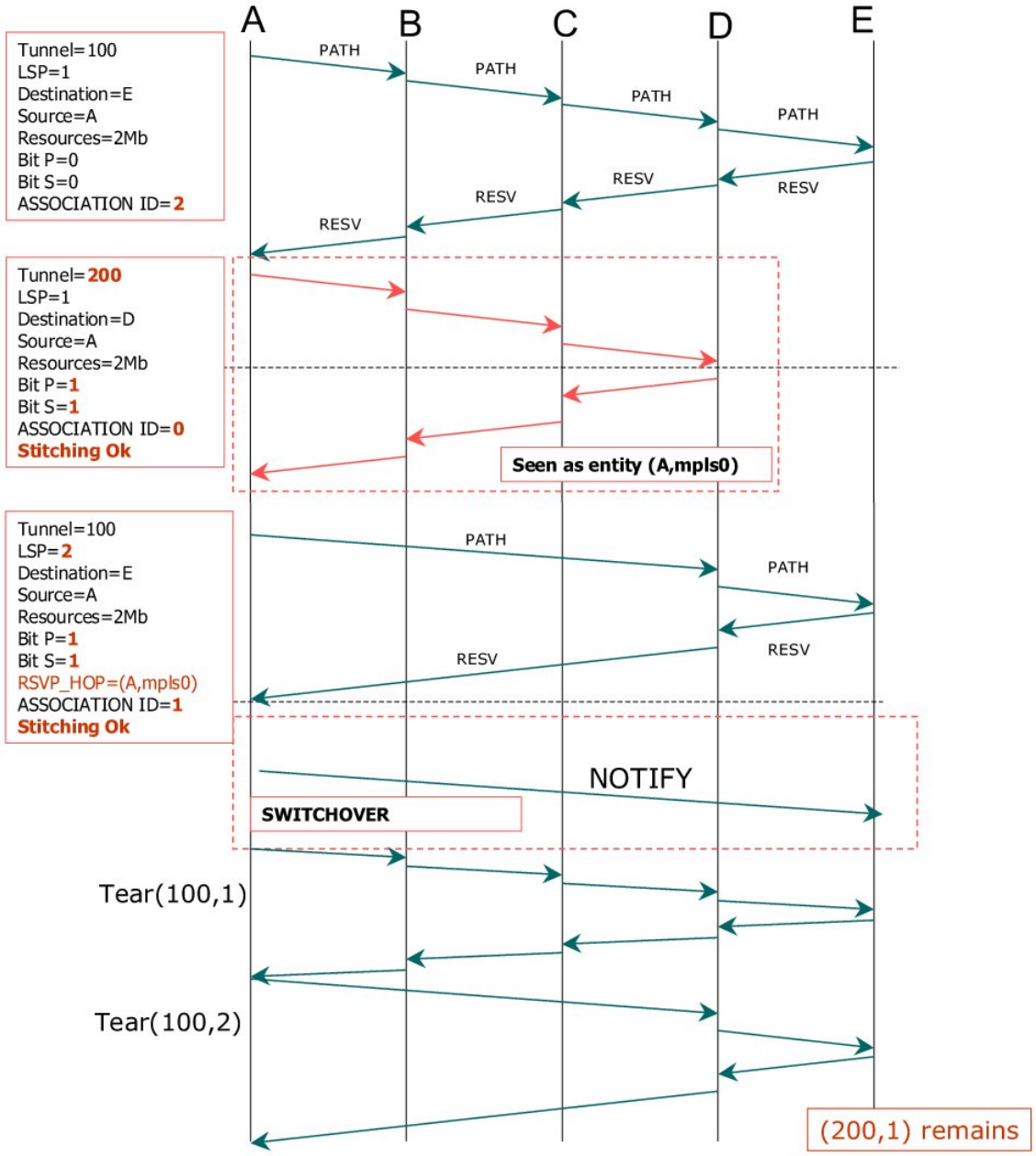


Figure 6.9: Removing the egress point

indicates that [A,D,E] protects [A,B,C,D,E]. However, [A,D,E] is signaled as a secondary and protecting LSP (bits P and S are set): resources are not yet committed since the protected LSP, [A,B,C,D,E], is still working.

At this point, node A sends a NOTIFY message to E to inform it that the protecting LSP must now be used, the detailed procedure for switching over from a woking LSP to a protecting LSP is detailed in [LRP06]. Automatically, resources are released from [A,B,C,D,E] and reallocated to [A,D,E] and [A,B,C,D] ([A,B,C,D] needs to be signalled

as working, for resources to be allocated). At this point both [A,B,C,D] and [A,D,E] are in use, [A,B,C,D,E] is deactivated. [A,B,C,D] can now be used as a regular bus-LSP, and traffic can be switched to [A,B,C,D]. A can now teardown [A,D,E] and [A,B,C,D,E] which are no longer in use.

This procedure depends however on the speed of resource reallocation in the case of protecting LSPs activation such as described in [LRP06], and on the fact that switching [A,B,C,D,E] to [A,D,E] will effectively allocate those resources to [A,B,C,D] as well, instead of preempting some other LSP's resources for [A,B,C,D]: this last question is implementation-dependent, since it is out of the scope of [LRP06] and related IETF documents.

6.2.3.6 Bus-LSP Merging

We will now consider the problem of merging bus-LSPs. Three different cases have to be considered:

- Adjacent merging: bus-LSPs [A,B] and [B,C,D,E] to be merged
- Disjoint merging: bus-LSPs [A,B] and [C,D,E] to be merged
- Overlapping merging: bus-LSPs [A,B,C,D] and [B,C,D,E] to be merged

Adjacent Merging

Suppose two bus-LSPs are to be merged, as described in Figure 6.10: [A,B] and [B,C,D,E]. This can be done by applying a procedure similar to that of the egress extension 6.2.3.4. The first step is to ask for stitching of LSP [A,B]. This is done as in previous section 6.2.3.4, using "LSP Stitching Desired" and "LSP segment stitching ready" mechanisms (again, note that this can also be done by nesting techniques, where available).

Once the LSP is ready for stitching, a new LSP is opened (new Tunnel ID, LSP ID): [A,B,E]. However it is specified that this new LSP is obtained by stitching [A,B] and [B,C,D,E], as defined in [AKV06]. The new [A,B,E] tunnel is then rerouted using the make-before-break procedure to become [A,B,C,D,E]. Though the two LSPs go through the same routers, it is again important that [A,B,C,D,E] be signaled as a single LSP, so that all nodes A, B, C, D, E are aware of the existence of the bus-LSP, as well as the other nodes in the routing area to which the bus-FA will be announced. Finally, both bus-LSPs [A,B] and [B,C,D,E] may be torn down as they are elegantly replaced by [A,B,C,D,E].

Disjoint Merging

To realize disjoint merging of two bus-LSPs [A,B] and [C,D,E] (see Figure 6.11), we will simply apply two operations:

- first extend the [C,D,E] to [B,C,D,E] by applying the procedure described in 6.2.3.3.

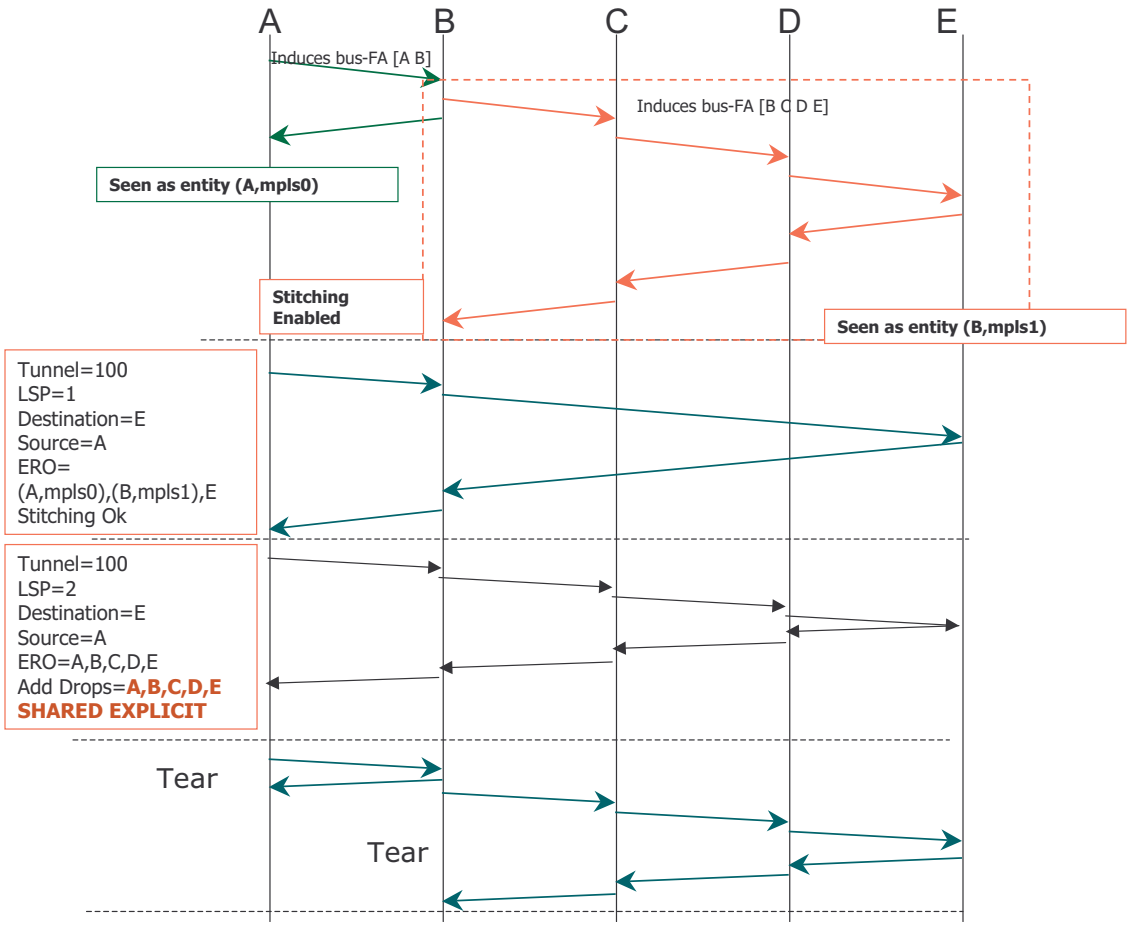


Figure 6.10: Adjacent Bus-LSP Merging

- second apply the adjacent merging as described previously

Of course, the first step could be replaced by extending [A,B] to [A,B,C], however the procedure for lengthening a bus-LSP is more complex when extending the downstream end.

Overlapping Merging

This is the most complex case of bus-LSP merging. For variation's sake, we will demonstrate how to perform this type of merging using nesting. We leave it to the reader as an exercise to apply stitching instead, knowing however that the stitching process happens the same as in previous scenarios.

Let's assume [A,B,C,D] and [B,C,D,E] are two established bus-LSPs, such as shown in Figure 6.12.

The first method described is illustrated in Figure 6.13. We open a nested LSP [A,D,E] which will have the following characteristics:

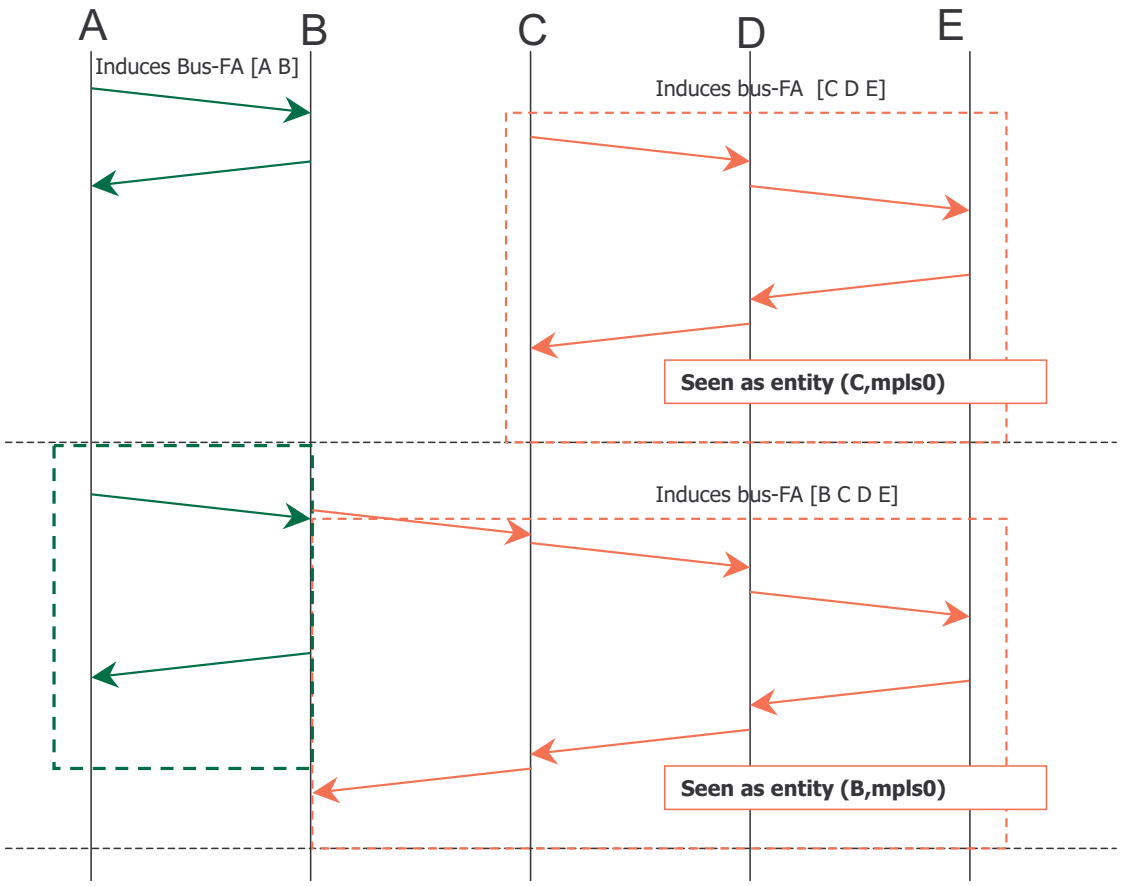


Figure 6.11: Disjoint Bus-LSP merging

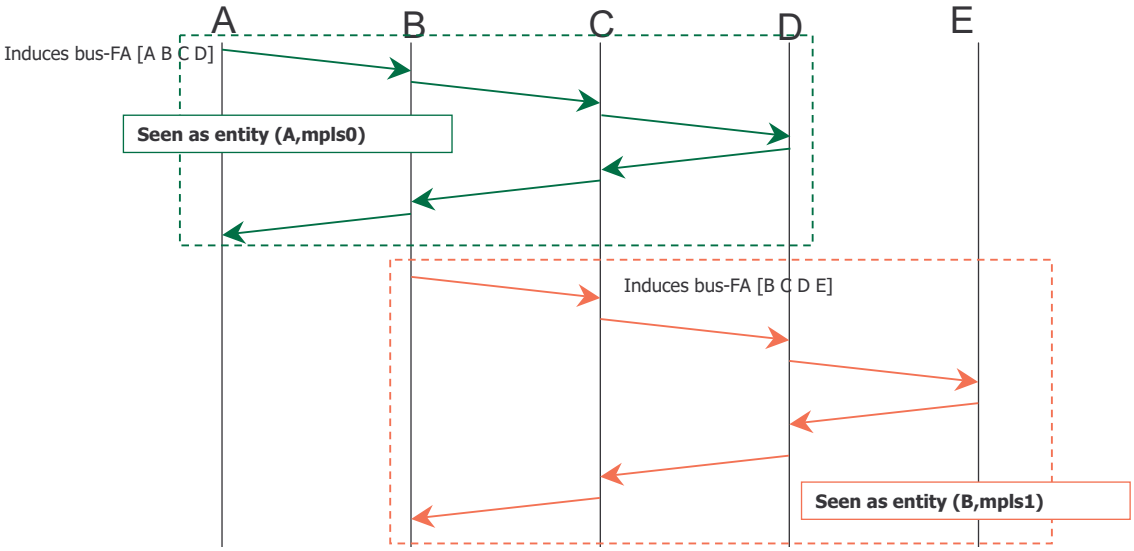


Figure 6.12: Overlapping bus-LSPs

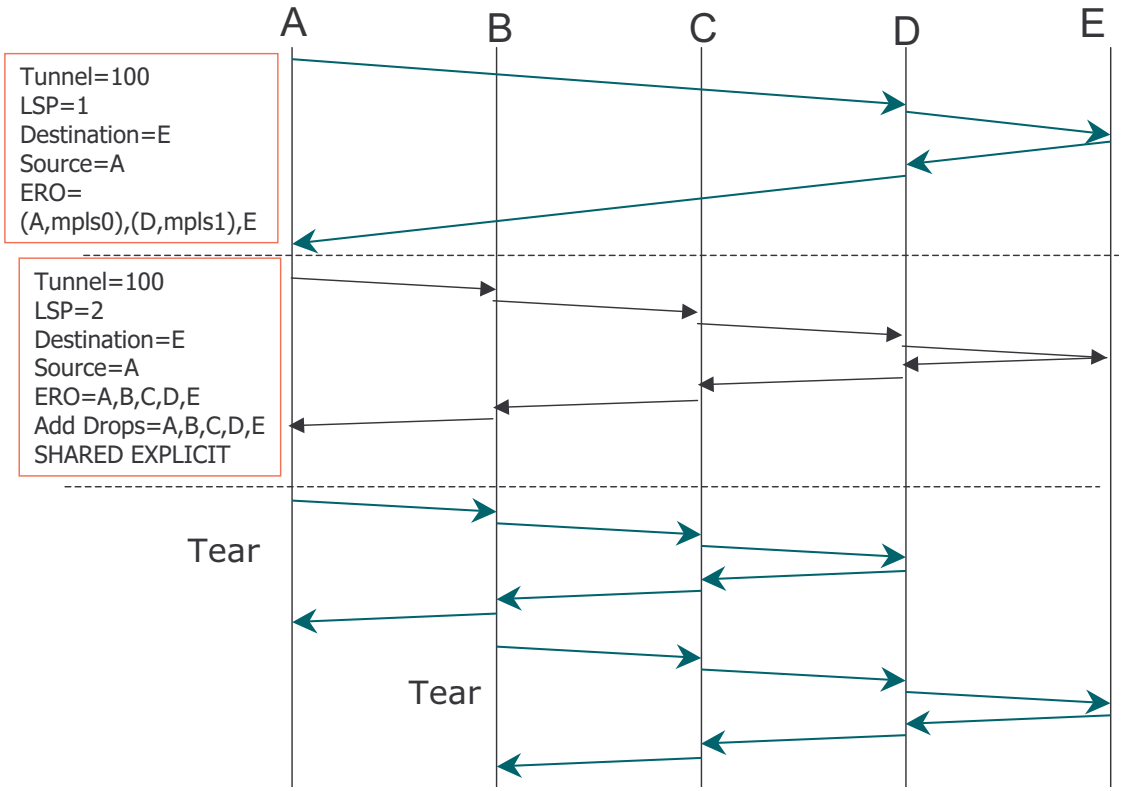


Figure 6.13: Merging Overlapping bus-LSPs, method 1

- a set of new identifiers: tunnel ID, LSP ID
- the destination (SESSION object) is E
- the source is A
- the ERO mentions the following hops: A, D, E. It is also mentioned that the D-E segment must use the existing bus-LSP segment [D,E] of [B,C,D,E].
- the RSVP_HOP object states that the LSP uses the [A,B,C,D] LSP to realize [A,D] connection.

The [A,D,E] nested bus-LSP is then rerouted according to [RFC3209] to [A,B,C,D,E]. However, note that at this point, nothing allows node B for instance to know that [A,D,E] and [A,B,C,D,E] need to share resources. Double booking is most likely going to happen here along the segments which are common to both initial LSPs ([B,C] and [C,D]). The initial bus-LSPs may now be torn down: [A,B,C,D] and [B,C,D,E].

To avoid the double-booking, an alternate scenario, which is however traffic-disrupting, is to shorten [B,C,D,E] to [D,E], as described in 6.2.3.3 and illustrated in Figure 6.2.3.6; then merge the two adjacent resultant bus-LSPs [A,B,C,D] and [D,E] as described in the

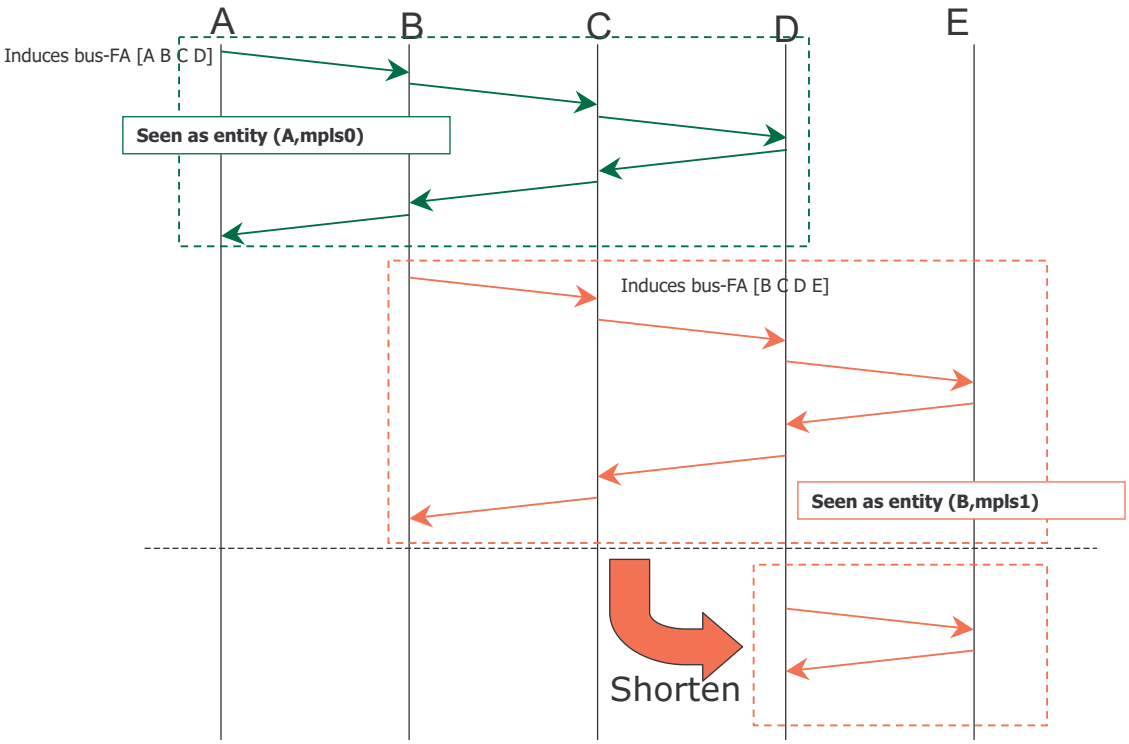


Figure 6.14: Merging Overlapping bus-LSPs, method 2

adjacent merging section. One could also equivalently shorten [A,B,C,D] to [A,B], though the process of manipulating egress is more complex. The choice of manipulating the ingress or the egress should be circumstance-based, noting that manipulating the ingress is much simpler, but may sometimes cause much traffic disruption. For instance, imagine merging [A,B,C] with [B,C,D,E,F,G,H,I]. In this case:

- manipulating the ingress consists in shortening [B,C,D,E,F,G,H,I] which might disrupt traffic from B to many destinations
- whereas manipulating the egress consists in shortening [A,B,C] to [A,B] which is maybe (depending on the traffic matrix) less disruptive, since the only adjacency destroyed is from A to C. (B to C is still available in the second bus-LSP).
- in this case, it might even be just as interesting to simply tear down [A,B,C] and extend [B,C,D,E,F,G,H,I] to become [A,B,C,D,E,F,G,H,I] as described in 6.2.3.3.

6.2.4 Extension Proposal

Because resource sharing is normally done based on the equality of the SESSION object (and in some cases, of the Sender address in the SENDER_TEMPLATE / FILTER_SPEC)

some of the bus-LSP manipulations are quite complex and can necessitate many operations, as the previous section 6.2.3 demonstrates: to simply extend the egress of a bus-LSP, 5 signaling operations are required (stitching preparation, opening an auxiliary LSP, rerouting the auxiliary LSP to the final bus-LSP, teardown of the initial LSP, teardown of the auxiliary LSP). In the merging case, some double booking or traffic-disruption cannot be avoided: this is a problem which should be addressed. Finally, complex operations, which involve multiple of the manipulations of 6.2.3 at once, are extremely complex and timely.

Therefore, to alleviate the signaling process, to allow more flexible operations and to enable new possibilities (real make-before break), an extension to RSVP-TE is proposed [BC07]. This extension is based on the PROTECTION and ASSOCIATION objects as defined in [LRP06]. However, the ASSOCIATION object is used in a slightly different way: it is, for now incompatible with [LRP06], but only little modifications of [LRP06] are needed for compatibility.

6.2.4.1 Recovery Attributes

The recovery attributes include all the parameters that determine the status of a LSP within the recovery scheme to which it is associated. These attributes are part of the PROTECTION object introduced in [LRP06]. They include the S (Secondary) bit, the P (Protecting) bit and the O (Operational) bit. The fields defined in [RFC3473] and in [LRP06] are unchanged by this document. The LSP recovery classification introduced in [LRP06] remains also unchanged, and the procedures here are designed for all eventualities:

- Full LSP Rerouting
- Preplanned LSP Rerouting without Extra-traffic
- LSP Protection with Extra-traffic
- Dedicated LSP Protection

Note: for the LSP protection scenarios (dedicated or with extra-traffic), the traffic may only be replicated downstream of the first node on both preempting and preempted LSPs.

6.2.4.2 LSP Association

The ASSOCIATION object, introduced in [LRP06], is used to associate the rerouted (initial, or preempted) and rerouting (new, or preempting) LSPs. When used for signaling the initial LSP, the Association ID (16 bit) of the ASSOCIATION object identifies the initial LSP Tunnel ID (whereas in [LRP06], it identifies the protecting LSP ID).

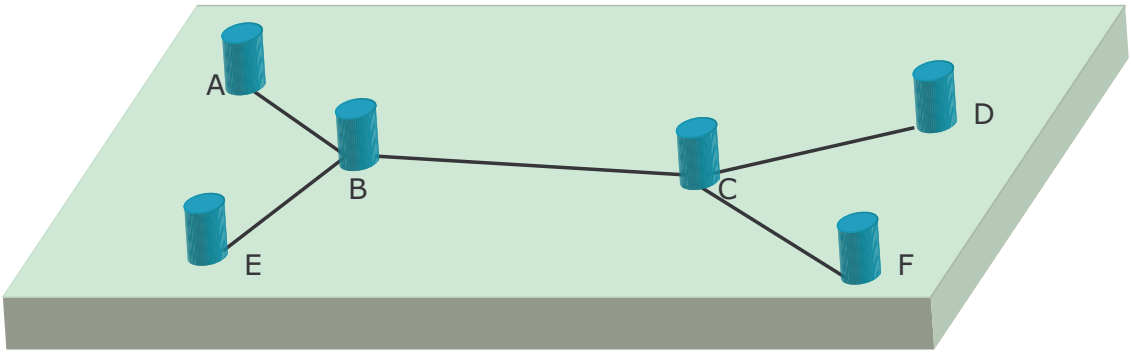


Figure 6.15: Network topology used for illustrating extension proposal

When signaling the new LSP, two ASSOCIATION Objects are used. The first one has its Association source set to the initial LSP source IP, and the Association ID set to the same Tunnel ID. In short: it carries the same ASSOCIATION Object as the initial LSP to be rerouted. The second ASSOCIATION object carries the new LSPs ingress as Association source and its Tunnel ID as Association ID.

Note: The ASSOCIATION Object in the initial LSP and the second ASSOCIATION in the preempting LSP are actually easy to infer from their SESSION Objects. They have been put here for ease of manipulation.

6.2.4.3 General procedures

Consider the network consisting of:

- 6 nodes A, B, C, D, E, and F,
- 5 bidirectional links (A,B), (E,B), (B,C), (C,F), (C,D).

This topology is illustrated in Figure 6.15. We will use this network as an illustration for our extension proposal.

The initial bus-LSP is [A,B,C,D] and the new bus-LSP is [E,B,C,F]. Whatever the protection scheme is, the procedures are similar. We call the preempted bus-LSP [A,B,C,D] LSP1 and preempting bus-LSP [E,B,C,F] LSP2.

The two LSPs carry different SESSION objects, LSP IDs and Source Addresses. So as to facilitate association therefore, LSP1 carries an ASSOCIATION Object. LSP2 carries the same ASSOCIATION Object. The identifiers for the illustrating example are detailed in Figure 6.16. How Node E can have knowledge of LSP1 Tunnel ID is beyond the scope of this extension to RSVP-TE, however, if E were a part of the LSP1 as well, then this information would be carried by signaling messages; else, some other mechanism must be designed, such as information flooding, in the routing information for instance. LSP2

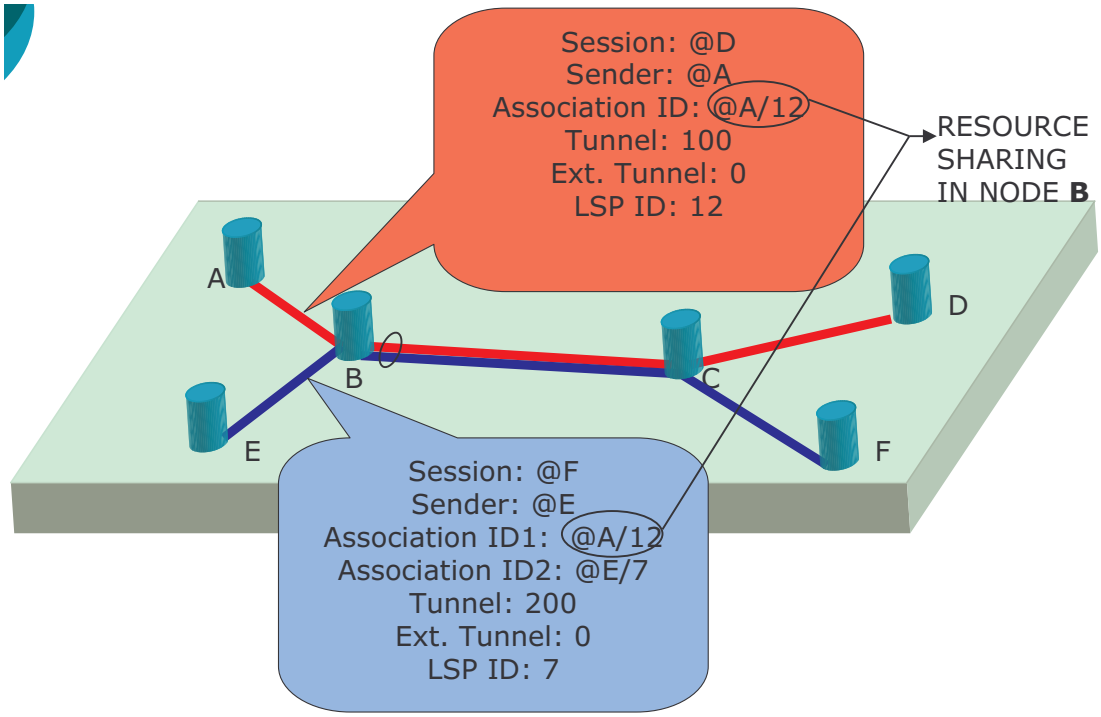


Figure 6.16: Identifiers for the illustration network

also carries a second ASSOCIATION Object with its own Tunnel ID and Source Address carried. Based simply on the ASSOCIATION Objects, nodes B and C are aware of the fact that LSP1 and LSP2 are associated.

To effectively switch to the preempting bus-LSP, a "Preemption asked" flag is set in the SESSION_ATTRIBUTES of LSP2. Upon reception of this flag, the first node common to both LSPs, here B, initiates the switchover. It knows it is the first common node to 2 LSPs based on the EROs / RROs. Before enabling LSP2, B sends a Notify message including LSP1's and LSP2's SESSION Object within the <upstream/downstream session list> (see [RFC3473]) and the new error code/subcode "Notify Preemption/LSP ready to preempt" (the exact value of this error code is to be defined by the community and IANA). This notification is sent to the head-end of LSP1, A, which decides, based on policy, whether to allow preemption or not. If it does not, nothing changes, and head-end B of LSP2 is responsible for further course of action. If it does, the A simply initiates a teardown of LSP1, according to procedures (see [RFC3473]). As B receives this teardown order, it will enforce the preemption, by enabling LSP2 (this depends on the protection policy, see [LRP06] for change of S,O,W bits and procedures order). B should maintain LSP1 alive (by refreshing PATH/RESV messages) until LSP2 has taken over, even though this means that a part of LSP1 will temporarily no longer exist and will yet be still seen downstream of B. Once LSP1 is tore down and LSP2 is the only remaining LSP, the new head-end E should remove the first ASSOCIATION Object, the other one becoming the

one used for further rerouting.

If an egress node should receive a ”Preemption asked” flag, an error should be sent to its ingress indicating that LSP2 and LSP1 do not have a shared node along their paths, when this is the case (based on ERO/RRO information). E may then teardown LSP2.

6.2.4.4 Scenario: appending a node to an bus-LSP

Let us consider the following situation. We have an bus-LSP established [A,B,C,D] and we wish to extend it for it to become [A,B,C,D,E] as in section 6.2.3.4. The initial bus-LSP, LSP1, has the following identifier set:

- SESSION: @D
- LSP ID: X
- Tunnel ID: Y
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object:
  - Association source: @A
  - Association ID: Y
- Protection Object Flags:
  - O: 1
  - S: 0
  - P: 0

While this first bus-LSP is being used, the second one, LSP2, can be setup using the following identifier set in the PATH message:

- SESSION: @E
- LSP ID: X’
- Tunnel ID: Y’
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object 1:

  - Association source: @A
  - Association ID: Y

- Association Object 2:
  - Association source: @A
  - Association ID: Y'

- Protection Object Flags:
  - O: 0
  - S: 1
  - P: 1

Concerning the ERO/RRO, if the head-end wishes to keep the same path for the [A,B,C,D] section of the bus-LSP, it is up to it to set the ERO accordingly.

For nodes such as B, C, and D, (i.e.: nodes which already are maintaining a state for LSP1) receiving the PATH message fo LSP2, the fact that the association object 1 mentions @A and ID Y, these nodes know that they have to share resources among LSP1 and future LSP2.

For nodes which receive the PATH message for LSP2 without prior reservation state for LSP1, such as E in our example, or intermediate loosely routed nodes (for instance, a B' node could have been inserted between B and C due to loose routing), these nodes process the PATH and RESV messages as usual.

When LSP2 PATH and RESV messages have gone all the way, LSP2 is provisioned and LSP1 can be switched over to LSP2. Given that both LSP1 and LSP2 share the same ingress, A, the preemption can take place immediately, without any Notification messages. In this case, A simply generates a TEARDOWN procedure for LSP1, letting LSP2 become operational (O bit set to 1, S to 0, P to 0). LSP2 should also stop carrying the Association Object referring to @A/Y.

We will not detail how pruning the egress node exactly happens, as it is the same procedure and necessitates no further explanations.

6.2.4.5 Scenario: pruning a head-end node from an bus-LSP

Let us consider the following situation. We have a bus-LSP established [A,B,C,D,E] and we wish to shorten it for it to become [B,C,D,E] as in section 6.2.3.3. The initial bus-LSP, LSP1, has the following identifier set:

- SESSION: @E

- LSP ID: X
- Tunnel ID: Y
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object:
  - Association source: @A
  - Association ID: Y
- Protection Object Flags:
  - O: 1
  - S: 0
  - P: 0

While this first bus-LSP is being used, the second one, LSP2, can be setup using the following identifier set in the PATH message:

- SESSION: @E
- LSP ID: X'
- Tunnel ID: Y'
- Ext. Tunnel ID: 0
- Tunnel Sender: @B
- Association Object 1:
  - Association source: @A
  - Association ID: Y
- Association Object 2:
  - Association source: @B
  - Association ID: Y'
- Protection Object Flags:
  - O: 0
  - S: 1

– P: 1

Concerning the ERO/RRO: if the head-end wishes to keep the same path for the [B,C,D,E] section of the bus-LSP, it is up to it to set the ERO accordingly.

For nodes such as C, D, and E, (i.e.: nodes which already are maintaining a state for LSP1) receiving the PATH message fo LSP2, the fact that the association object 1 mentions @A and ID Y, these nodes know that they have to share resources among LSP1 and future LSP2.

For nodes which receive the PATH message for LSP2 without prior reservation state for LSP1, such as intermediate loosely routed nodes (for instance, a B' node could have been inserted between B and C due to loose routing), these nodes process the PATH and RESV messages as usual.

When LSP2 PATH and RESV messages have gone all the way, LSP2 is provisioned and LSP1 can be switched over to LSP2. Given that LSP2's head-end is a part of LSP1, it can send a Notify message including LSP1's and LSP2's SESSION Object. This notification is sent to the head-end of LSP1, A, which decides, based on policy, whether to allow preemption or not. If it does, A can simply initiate a teardown of LSP1. Once LSP2 has become operational (O bit set to 1, S to 0, P to 0), LSP2 should also stop carrying the Association Object referring to @A/Y.

We will not detail how adding an ingress node exactly happens, as it is the same procedure and necessitates no further explanations.

6.2.4.6 Scenario: replacing an ingress and/or an egress

We will detail here the most complex of these three procedures, the two others can be obtained quite simply and are left as an exercise to the reader. Let us consider the following situation. We have a bus-LSP established [A,B,C,D] and we wish to extend it for it to become [E,B,C,F].

The initial bus-LSP, LSP1, has the following identifier set:

- SESSION: @D
- LSP ID: X
- Tunnel ID: Y
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object:

  - Association source: @A
  - Association ID: Y

- Protection Object Flags:
  - O: 1
  - S: 0
  - P: 0

While this first bus-LSP is being used, the second one, LSP2, can be setup using the following identifier set in the PATH message:

- SESSION: @F
- LSP ID: X'
- Tunnel ID: Y'
- Ext. Tunnel ID: 0
- Tunnel Sender: @E
- Association Object 1:
  - Association source: @A
  - Association ID: Y
- Association Object 2:
  - Association source: @F
  - Association ID: Y'
- Protection Object Flags:
  - O: 0
  - S: 1
  - P: 1

Concerning the ERO/RRO: if the head-end wishes to keep the same path for the [B,C] section of the LSP, it is up to it to set the ERO accordingly.

For nodes such as B and C, (i.e.: nodes which already are maintaining a state for LSP1) receiving the PATH message fo LSP2, the fact that the association object 1 mentions @A and ID Y, these nodes know that they have to share resources among LSP1 and future LSP2.

For nodes which receive the PATH message for LSP2 without prior reservation state for LSP1, such as F, or intermediate loosely routed nodes (for instance, a B' node could have been inserted between B and C due to loose routing), these nodes process the PATH and RESV messages as usual.

When LSP2 PATH and RESV messages have gone all the way, LSP2 is provisioned and LSP1 can be switched over to LSP2. To effectively switch to the preempting LSP2, a "Preemption asked" flag is set in the SESSION_ATTRIBUTES of LSP2. Upon reception of this flag, the first node common to both bus-LSPs, here B, initiates the switchover. To do so, B sends a Notify message including LSP1's and LSP2's SESSION Object. This notification is sent to the head-end of LSP1. Once A accepts preemption, it simply initiates a teardown of LSP1. As B receives this teardown order, it will enforce the preemption, by enabling LSP2. Once LSP2 has become operational, LSP2 should also stop carrying the Association Object referring to @A/Y.

6.2.4.7 Scenario: Using this extension to stitch / merge two bus-LSPs

Let us consider the following situation. We have these two bus-LSPs (LSP1 and LSP2) established: [A,B,C] and [D,E,F] and we wish to merge them to become LSP3 [A,B,C,D,E,F]. This example could also work just as well if C and D were the same nodes (adjacent merging case of section 6.2.3.6) or if [B,C] and [D,E] were equal (overlapping merging case of section 6.2.3.6).

We simply create LSP3 and associate it both with LSP1 and LSP2. It should therefore carry 3 association objects.

Here are the relevant identifier sets:

For LSP1:

- SESSION: @C
- LSP ID: X
- Tunnel ID: Y
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object 1:
  - Association source: @A
  - Association ID: Y

For LSP2:

- SESSION: @F
- LSP ID: X'
- Tunnel ID: Y'
- Ext. Tunnel ID: 0
- Tunnel Sender: @D
- Association Object 1:
  - Association source: @D
  - Association ID: Y'

For LSP3:

- SESSION: @F
- LSP ID: X"
- Tunnel ID: Y"
- Ext. Tunnel ID: 0
- Tunnel Sender: @A
- Association Object 1:
  - Association source: @A
  - Association ID: Y
- Association Object 2:
  - Association source: @D
  - Association ID: Y'
- Association Object 3:
  - Association source: @A
  - Association ID: Y"

For nodes such as B and C, (i.e.: nodes which already are maintaining a state for LSP1) receiving the PATH message for LSP3, the fact that the association object 1 mentions @A and ID Y, these nodes know that they have to share resources among LSP1 and future LSP3.

For nodes such as D, E, and F (i.e.: nodes which already are maintaining a state for LSP2) receiving the PATH message for LSP3, the fact that the association object 2 mentions @D and ID Y', these nodes know that they have to share resources among LSP2 and future LSP3.

For nodes which receive the PATH message for LSP2 without prior reservation state for LSP1, such as intermediate loosely routed nodes (for instance, a B' node could have been inserted between B and C due to loose routing), these nodes process the PATH and RESV messages as usual.

When LSP3 PATH and RESV messages have gone all the way, LSP3 is provisioned and LSP1 and LSP2 can be switched over to LSP3. To effectively switch to the preempting LSP2, a "Preemption asked" flag is set in the SESSION_ATTRIBUTES of LSP3. Upon reception of this flag, the first node common to two bus-LSPs, here D (A also, but given it is the head-end of two bus-LSPs, its task is simplified), initiates the switchover. If D accepts preemption, it simply initiates a teardown of LSP2 and enforces the preemption, by enabling LSP3. Likewise, A initiates a TearDown of LSP1. Once LSP3 has become operational, and LSP1 and LSP2 are destroyed, LSP3 should also stop carrying the Association Object referring to @A/Y and to @D/Y'.

There is no limitation on the number of merged bus-LSPs using this technique, and this allows to merge in just one pass multiple bus-LSPs.

6.3 Concluding Remarks

This chapter has presented the principles of extensions both to the GMPLS intra-domain routing protocol OSPF-TE and to the GMPLS signaling protocol RSVP-TE. Both these extensions enable the bus-LSP to be supported in a network, bringing the bus-LSP-related benefits, detailed in the two previous chapters 4 and 5, to GMPLS-controlled networks.

These extensions have not yet been submitted as drafts to the IETF CCAMP working group.

Chapter 7

Conclusion

We offer in this chapter the general conclusion of the different studies presented in this thesis. We summarize the main contributions of our work and suggest future research directions that stem from it. We also would like to encourage our readers to visit the web page at http://perso.enst.fr/~brehon/ to obtain source code of the tools developed and used for these studies.

7.1 Summary of Contribution

In the context of Metro access networks, our study applies to Ethernet metropolitan networking. Work at the Metro Ethernet Forum has standardized various flavors of Ethernet. However, no study had yet proposed an efficient way of providing the Ethernet VPN service while balancing the load over the network. Our work, presented in [BKC07], treats the problem as an optimization problem and provides a novel and original formulation of it. Due to tractability issues solving the problem formulated in an exact way, it also offers a heuristic algorithm for balancing the load of the clients over a meshed network. This approach provides better results in terms of load balancing than other existing algorithms which consider similar problems. We showed in the thesis that our heuristic outperforms all known concurrent approaches in terms of load balancing, by reducing the load on the maximally loaded link by 8% to 25%, at the cost of a slight increase of the average load on each link (3% to 9%). Our approach can be used in practical applications as the designed algorithms are simple and converge fast. The interest of this work is therefore theoretical and practical, and yields more balanced traffic-engineering of the network: this is of value to the network operator.

The rest of our thesis concentrates on Carrier Ethernet, by studying an innovative structure called the bus-LSP. This structure can be used in any GMPLS-controlled network and is not restricted to Ethernet, but could apply to IP over WDM, Packet over SONET, etc.

The first study, presented in [BK06] introduced the bus-LSP concept and qualitatively evaluated CAPEX and OPEX benefits that this structure may bring in single- and multi-layer networks. It also did a quantitative evaluation of the OPEX benefits in terms of layout complexity in single-layer networks. These results proved that bus-LSPs allow a layout complexity reduction of more than 80% while maintaining network bandwidth consumption near optimality (within a few percents).

In [BKPD07], we provided a new set of tools – FT-SPF and PIBRA algorithms – enabling the development of a heuristic approach for optimally designing the virtual layout in multi-layer networks. This set of tools was designed to solve the problem, the formulation of which was presented but impossible to use for realistic, large networks due to computational issues. This heuristic algorithm optimizes the network cost for the operator, both in terms of CAPEX and OPEX, by maximizing the total amount of traffic accepted by the network, minimizing the cost in upper layer routers, and minimizing the overall complexity of the layout measured in the number of connections setup. Our results show that the heuristic developed is able to perform in a short amount of time (in the order of minutes at most), and yields traffic-engineering rules which may be applied by a network operator. The numerical results of the heuristic proved to scale well with load, but not as well with network size (network scaling is 71% of optimal for VTHD, 36% for the Italian network). During the optimization procedure of the heuristic algorithm, the load of the maximally loaded link is reduced, while other parameters are also optimized, such as the amount of interfaces needed to interconnect both layers, the overall routing processor power needed, and the number of bus-LSPs needed. Numerical studies showed that our algorithm reduced the total traffic routed in the upper layer in the initial routing chosen by more than 45%, the number of overall bus-LSPs by more than 50%, and even reduced the average number of links (which directly translates to propagation delay). That is, our heuristic fills its purpose of minimizing both CAPEX and OPEX for the network operators implementing the bus-LSP technology.

Finally, the study of this new structure was completed by considering the GMPLS standard protocols extensions which are needed to operate bus-LSPs and announce the bus-FAs they generate. This resulted in the study of routing protocol (OSPF-TE) options and in the elaboration design guidelines; this also led to a signaling protocol (RSVP-TE) limitations analysis and a technical proposal for extending this protocol [BC07]. This study of the evolutions needed for the GMPLS control plane used to manipulate bus-LSPs and bus-FAs allows the integration of bus-LSPs in an automated, highly manageable network in which costs are minimized thanks to network engineering.

7.2 Future Work

The studies presented in this thesis open the way for future work. These additional considerations are in the domains of both Metro and Carrier Ethernet. Indeed, there is a strong driver by network operators and providers to use Ethernet in these domains. Ethernet is being extended to support many metro- and carrier-grade features such as hard quality of service and protection guarantees, high manageability and cross-domain and cross-layer interconnections, among many others. We list here some of the issues which stem from our research and which are related to this trend:

- In the Metro domain:
  - There is some concern about so-called *leakage*. Leakage happens during the learning (of MAC addresses) phase: the broadcast traffic of a customer can end up at some PEs where there is no presence of this customer. This is due to the fact that multiple customers are placed on the same tree while not being all present at the same PEs. The leakage phenomenon must be understood and filters developed to prevent this from happening.
  - There is today a large demand for multicast functionalities by service providers, for mass content distribution for instance. The mapping of multicast traffic over trees is an issue which is very close to the one considered in [BKC07], so it might be very interesting to build on and apply some of the principles explored to a different context, in particular for dimensioning the network using the load-balancing properties of our algorithm.
- In the Core domain (concerning bus-LSPs and bus-FAs):
  - The bus-LSP concept allows the interconnection of multiple nodes between which bus-FAs are created. This structure might therefore be used for offering a VPN service, using for instance just two unidirectional bus-LSPs running through all VPN sites (see Section 3.2.4). The bus-LSP structure would avoid setting up full mesh connections, however the trade-off in terms of resource use must be evaluated: since the bus-LSP is linear, a traveling salesman problem has to be solved to interconnect all sites in an optimal way; such a solution should be compared to resource consumption in non-bus-LSP enabled networks.
  - Bus-LSPs are limited in use, for now, to opaque optical networks (or non-optical networks), since an electronic device is needed to update the counter associated with each data packet must be decremented at each hop. However, designing some MAC protocol which would allow to do this in transparent optical networks would allow the use of the bus-LSP structure. Note that the *add* functionality has already been designed in [BPD+04].

- The heuristic approach developed in [BKPD07] provides CAPEX- and OPEX-wise interesting results, however there are several drawbacks to this approach. Some are technical issues which are raised and discussed in Section 5.5, but operators could definitely benefit from an algorithmic approach allowing for non-disruptive dynamic virtual topology design. This implies working on the rerouting procedures on our approach so as to evaluate and minimize traffic disruption during traffic matrix changes.
- The bus-LSP structure is one step away from offering some degraded multipoint-to-multipoint functionality, by allowing packets to be both dropped and forwarded in the same node. This would be useful for multicast VPNs or content distribution; however it needs additional research, in particular in the dimensioning area and the standard protocol support (routing and signaling protocols extensions).
- Making a bus-LSP circular (egress at the ingress) could provide some interesting functionalities in VPN and multipoint to multipoint. This can still be emulated today by two bus-structured FAMAs running parallel and in opposite directions. This area is closely related to the question of ring-shaped virtual topologies and their efficiency, and needs, of course, more signaling and routing research.
- Finally, the question of bus-LSP protection must be investigated: in particular, there are issues raised by the failure of an intermediate node/link. Some of the FAs induced by a partly failed bus-LSP might still be maintained if the nodes connected are on the same side of the failure; the question is therefore that of maintaining partly functional bus-LSPs, refreshing the valid FAs, and so on. Also, protection of bus-LSPs could be only partial, protecting only a subset of the bus-FA for example, or protection might be differentiated along the bus; this needs further investigation and analysis.

Appendix A

AMPL models

A.1 AMPL model for problem of Section 2.4

We present here the files which could be used in Section 2.4 to obtain exact results. However, even for the small data file presented here, CPLEX is not able to give a solution in a reasonable amount of time. Notice there is a piecewise-linear approximation of the (2.4.15). The model is available at: http://perso.enst.fr/~brehon/.

A.1.1 Model

```
#CPLEX command:
#reset;model VPN_STP\vpn.mod;data VPN_STP\test_net.dat;
#option solver cplex;solve;




#------------------------#
#    GIVEN               #
#------------------------#


param Nb_Links >=0;
param Nb_Nodes >= 0;
param Nb_VPN >= 0;
param Nb_ST >= 0;
param maxflow = 100000; # max flow on a link
param maxclients = 100; # max number of clients / VPN
```

```
param nb_breaks >=0; # number of breaks in the approx of x(1-x)

set LINK := 1 .. Nb_Links; # graph links
set NODE := 1 .. Nb_Nodes; # network nodes
set VPN := 1 .. Nb_VPN; # VPNs
set ST := 1 .. Nb_ST; # Spanning trees

param in_VPN {NODE,VPN} >= 0; # if node NODE belongs to VPN
param access_bw {VPN} >= 0;
  # minimal guaranteed access bandwidth for nodes of VPN
param endpoint {LINK,NODE} >=0 ; # if LINK has an end at NODE
param capacity {LINK}>=0; # capacity of LINK
param b{1..nb_breaks}; # break points of the approx of x(1-x)
param s{0..nb_breaks}; #slopes of the approx of x(1-x)
param nb_Clients{i in VPN} = sum{v in NODE} in_VPN[v,i];
  #nb of clients (sites) of vpn i




#------------------------#
#      VARIABLES         #
#------------------------#


var Flow {e in LINK, i in VPN , j in ST} >= 0;
  # flow on e due to VPN i mapped to ST j
var Flow_st {e in LINK , j in ST} >= 0; # volume on e due to ST j
var Flow_link {e in LINK} >= 0; #total volume on e
var Lambda >= 0; # traffic matrix multiplier
var Map {i in VPN, j in ST} binary; #if VPN i is mapped to ST j
var If {v in NODE, e in LINK, i in VPN, j in ST} >=0, integer;
    # number of clients of i to which access is through
    # v by e, when i is mapped to j
var Node_Prop {v in NODE, e in LINK, i in VPN, j in ST} >=0;
    # proportion of clients of i to which access is through v by e,
    # when i is mapped to j

var Dir {e in LINK, j in ST, v in NODE} binary;
```

```
    # =1 if e is active in j, in direction v -> e -> root
var u{j in ST} binary; #if ST j is used




#------------------------#
#     OBJECTIVES         #
#------------------------#


#OBJECTIVE 1: minimize number of STs
minimize Total_Flow: sum{e in LINK} Flow_link[e];
#minimize ST_Number: sum{j in ST} u[j];
#minimize ST_Num_and_Flows:
#   sum{j in ST} u[j] + sum{e in LINK} Flow_link[e];

#minimize Traf_Mult: Lambda;




#------------------------#
#   CONSTRAINTS          #
#------------------------#


#  DOMAIN DEFINITIONS

subject to Dir_If_Link_Connected{j in ST,e in LINK, v in NODE}:
  Dir[e,j,v] <= endpoint[e,v];
#"dir" variables are meaningless if e and v aren't connected

subject to If_If_Link_Connected{i in VPN,j in ST,e in LINK, v in NODE}:
  If[v,e,i,j] <= endpoint[e,v] * maxclients;
#"if" variables are meaningless if e and v aren't connected

subject to Connexity {j in ST, i in VPN, v in NODE,
                      e in  LINK : endpoint[e,v]=1}:
```

```
   If[v,e,i,j] <= nb_Clients[i];
# each node must allow access to at most all VPN nodes (this
# prevents loops with any vpn member to form)




#  MAPPING CONSTRAINTS

subject to One_ST {i in VPN}:
  sum {j in ST} Map[i,j] = 1;
# Constraint: one ST /VPN

subject to ST_use{j in ST}:
  u[j] >= sum {i in VPN} Map[i,j] / Nb_VPN;
#defines if a ST is used




#  SPANNING TREE DEFINITION


subject to ST_Link_Number {j in ST}:
  sum {e in LINK, v in NODE} Dir[e,j,v] = Nb_Nodes - 1;
# Constraint: a connex graph of n nodes has n-1 links in each ST

subject to One_Uplink {j in ST, v in NODE}:
  sum {e in LINK} Dir[e,j,v] <= 1;
# Constraint: each node has at most one "uplink"

subject to One_Dir {j in ST, e in LINK}:
  sum{v in NODE: endpoint[e,v]=1 } Dir[e,j,v] <= 1;
#Constraint: a link is an uplink to only ONE node




#  INTERFACE CONSTRAINTS
```

```
subject to VPN_Node {j in ST, i in VPN, v in NODE,
e in  LINK : in_VPN[v,i] = 1 and endpoint[e,v]=1}:
  If[v,e,i,j] = Map[i,j];
# Constraint: for VPN nodes, graph leaves, set value of interface to 1

subject to Non_VPN_Node {j in ST, i in VPN, v in NODE,
e in LINK: in_VPN[v,i] = 0 and endpoint[e,v]=1}:
  If[v,e,i,j] >= (sum { ee in  LINK, vv in NODE : ee <> e and
                        endpoint[ee,v]=1 and vv <> v and
                        endpoint[ee,vv]=1} If[vv,ee,i,j])
                 - ( 1 - sum { vv in NODE } Dir[e,j,vv]) * maxclients;
# Constraint: for non VPN nodes, interface is worth the sum of value of
# interfaces its other links go to

subject to ST_If_Only { j in ST, i in VPN, v in NODE, e in LINK}:
  If[v,e,i,j] <= sum { vv in NODE : endpoint[e,vv] = 1 } Dir[e,j,vv]
                 * maxclients;
#constraint: never set an interface value if link is not part of ST

subject to If_Sum_To_Clients{j in ST, i in VPN, e in LINK}:
  sum { v in NODE : endpoint[e,v] =1} If[v,e,i,j] <= nb_Clients[i];
  # * Map[i,j] - ( 1 - sum { vv in NODE } Dir[e,j,vv]) * nb_Clients[i];
#To forbid If values which are too big on both sides of a link,
#which makes the flow=0...

subject to If_If_ST_Mapped{v in NODE, j in ST,i in VPN, e in LINK}:
  If[v,e,i,j] <= Map[i,j] * maxclients;
# To forbid If values if i is not mapped to j




#FLOW CONSTRAINTS


#subject to ST_Flows { i in VPN, j in ST, e in LINK}:
#  Flow [e,i,j] <= maxflow * Map[i,j];
```

```
# No flow if ST is not mapped to the VPN
#useless (implicit)

#subject to Dir_Needed {j in ST, i in VPN, e in LINK}:
#  Flow[e,i,j] <= sum { v in NODE} Dir [e,j,v] * maxflow;
# Constraint: a flow can only be on a link with a direction (active)
#useless (implicit)

subject to ST_Link {j in ST, e in LINK}:
  Flow_st[e,j] = sum {i in VPN} Flow[e,i,j];
#capacity used by a ST on a link

subject to Link_Flow {e in LINK}:
  Flow_link[e] = sum {j in ST} Flow_st[e,j];
#capacity used on a link

subject to Node_Prop_Def{j in ST, i in VPN, v in NODE,
                         e in LINK:endpoint[e,v]=1}:
  Node_Prop[v,e,i,j] = If[v,e,i,j] / nb_Clients[i];
# Node_Prop[v,e,i,j] = sum { vv in NODE : vv <> v and
#                            endpoint[e,vv] =1} If[vv,e,i,j]
#                      / nb_Clients[i];

subject to Flow_Alloc {j in ST, i in VPN, v in NODE,
                       e in LINK:endpoint[e,v]=1}:
# Flow[e,i,j] >= (access_bw[i] / ( -1 + nb_Clients[i])) *
#                (sum { vv in NODE : vv <> v and endpoint[e,vv] = 1}
#                                    If[vv,e,i,j]                    )*
#                (nb_Clients[i] -
#                 sum{vv in NODE:vv <> v and endpoint[e,vv]=1}
#                                     If[vv,e,i,j]              );
# non linear, replaced by:
  Flow[e,i,j] >= (access_bw[i] / ( -1 + nb_Clients[i])) *
                  nb_Clients[i] * nb_Clients[i] *
                 (<<{k in 1..nb_breaks}b[k];{k in 0..nb_breaks} s[k]>>
                   Node_Prop[v,e,i,j]                                 );
# Constraint: flow on a link = avg vpn flow * nb nodes accessed from
#             other side of a node
# nb_breaks: nb points where slope changes.
```

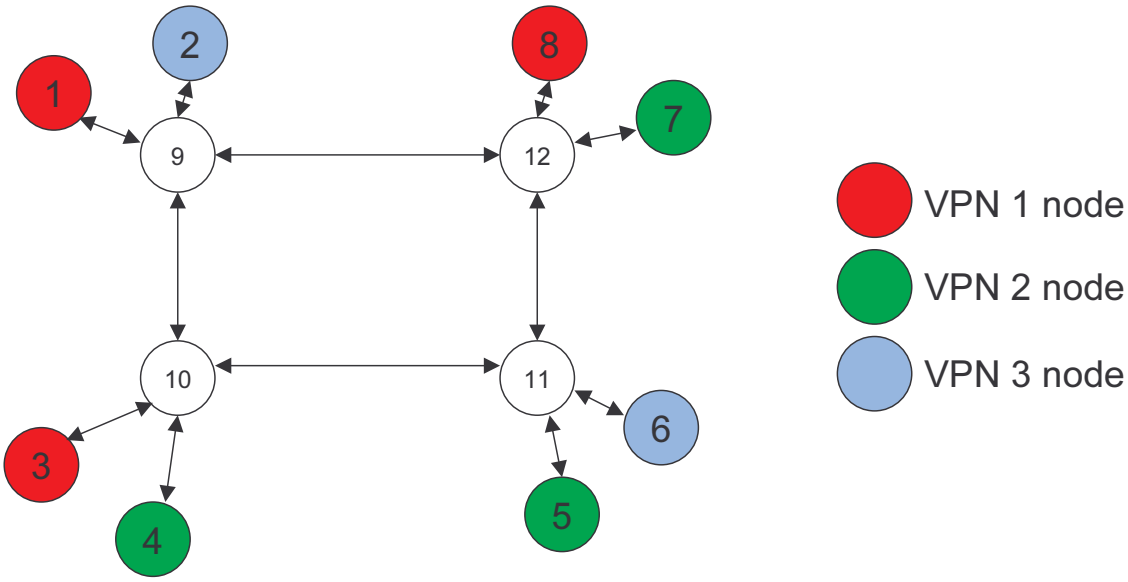


Figure A.1: Metro Access Network

```
#            b[k] abscissa of points, s[k]slope after point k.
#            to approx f(x)=x(1-x)


subject to Link_Capa {e in LINK}:
  Flow_link[e] <= capacity[e];# * Lambda;
# Constraint: link capacity constraint
```

A.1.2 Model data file

We represent the network shown in Figure A.1.

```
param Nb_Links  := 12;
param Nb_Nodes  := 12;
param Nb_VPN    :=  3;
param Nb_ST     :=  3;
param nb_breaks :=  4;
param b:=
1 0
2 0.25
3 0.75
4 1;
```

```
param s:=
0  0
1  1
2  0
3 -1
4  0;



#ligne: vpn
#colonne: nodes
#  v1 v2 v3
#n1
#n2

param in_VPN: 1 2 3 :=
1             1 0 0
2             0 0 1
3             1 0 0
4             0 1 0
5             0 1 0
6             0 0 1
7             0 1 0
8             1 0 0
9             0 0 0
10            0 0 0
11            0 0 0
12            0 0 0
;

#line: nodes
#column: links
#  n1 n2 n3
#l1
#l2

param endpoint: 1 2 3 4 5 6 7 8 9 10  11  12 :=
1               1 0 0 0 0 0 0 0 1  0   0   0
2               0 1 0 0 0 0 0 0 1  0   0   0
```

```
3               0 0 1 0 0 0 0 0 0  1   0   0
4               0 0 0 1 0 0 0 0 0  1   0   0
5               0 0 0 0 1 0 0 0 0  0   1   0
6               0 0 0 0 0 1 0 0 0  0   1   0
7               0 0 0 0 0 0 1 0 0  0   0   1
8               0 0 0 0 0 0 0 1 0  0   0   1
9               0 0 0 0 0 0 0 0 1  1   0   0
10              0 0 0 0 0 0 0 0 0  1   1   0
11              0 0 0 0 0 0 0 0 0  0   1   1
12              0 0 0 0 0 0 0 0 1  0   0   1
;


param access_bw:=
1 1
2 1
3 1;

param capacity:=
1  2
2  2
3  2
4  2
5  2
6  2
7  2
8  2
9  2
10 2
11 2
12 2;
```

A.2 AMPL model for problem of Section 4.2

A.2.1 Model

The model is available at: http://perso.enst.fr/~brehon/ as well as some programs to generate the data files based on simple textual network descriptions.

```
#CPLEX command:
#reset;model optim_fama3.mod;data optim_fama3.dat;
#option solver cplex;solve;




#------------------------#
#       GIVEN            #
#------------------------#


param Nb_Links >=0;
param Nb_Demands >= 0;
set LINK := 1 .. Nb_Links; # graph links
set PATH; # paths over the graph
set DEMAND := 1 .. Nb_Demands; # demands
set DEMAND_PATH within (DEMAND cross PATH);
  # possible combinations of demands to paths

param delta {DEMAND_PATH,LINK} >= 0;
  # if link e contributes to path p when demand d runs overs p.
param volume {DEMAND} >= 0; # volume of demand d
param capacity {LINK}; # capacity of link e




#------------------------#
#         VARIABLES      #
#------------------------#


var Flow {(d,p) in DEMAND_PATH} >= 0, <= volume[d];
  # volume of d routed over p
var Active {p in PATH} binary;
  #if path p is active active
#var Uses {p in PATH, e in LINK} binary;
  #if path p uses link e. Used for objective 3 only.
```

```
#------------------------#
#     OBJECTIVES         #
#------------------------#


minimize Nb_LSPs: sum {p in PATH} Active[p];
 Objective: total nb of active LSPs

#minimize Net_Cost:
#   sum {(d,p) in DEMAND_PATH, e in LINK} delta[d,p,e] * Flow[d,p];
# Objective: number of used links in network
# optionnally, multiply by cost/link

#minimize Max_Lambdas:
#   max{e in LINK} ( sum{p in PATH} (Active[p] * Uses[p,e]) );
#subject to Use{p in PATH, e in LINK}:
#   Uses[p,e] = max{(d,p) in DEMAND_PATH} delta[d,p,e];
#Objective: minimize max number of wavelengths used on a link




#------------------------#
#     CONSTRAINTS        #
#------------------------#


#subject to Net_Cost:
# sum {(d,p) in DEMAND_PATH, e in LINK} delta[d,p,e] * Flow[d,p] <= 5;
#constraint: not exceed optimal net cost (found by minimizing obj 2.)
#optionnally, multiply by cost/link

subject to Link_Capa {e in LINK}:
  sum {(d,p) in DEMAND_PATH} delta[d,p,e] * Flow[d,p] <= capacity[e];
# Constraint: link capacity constraint
```

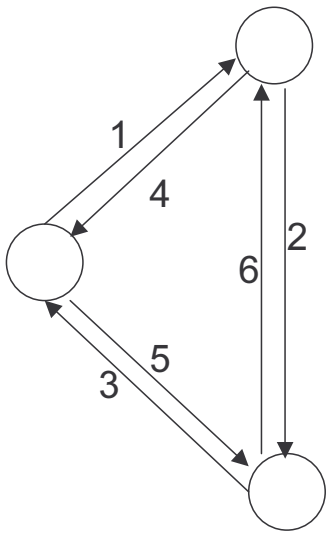


Figure A.2: Single Layer Network using bus-LSPs

```
subject to Active_Path {(d,p) in DEMAND_PATH}:
  Flow[d,p] <= volume[d] * Active[p];
# Constraint: a flow may only use an active path

subject to Demand_Satisfaction {d in DEMAND}:
  sum {(d,p) in DEMAND_PATH} Flow[d,p] = volume[d];
# Constraint: demand satisfaction
```

A.2.2 Model data file

We represent the network shown in Figure A.2. There are demands of volume 1 between each pair of nodes. Bus-LSPs of length 1 and 2 may be setup to satisfy demands. For example, demand 1 (connecting nodes at both ends of link 1) may be satisfied by a bus-LSP going through:

- link 1
- links 1 and 2
- or links 5 and 6

```
param Nb_Links := 6;
param Nb_Demands := 6;
set PATH := 1 12 5 56 4 45 2 23 6 64 3 31;
```

```
set DEMAND_PATH := (1,1) (1,12) (2,12) (5,12) (5,5) (5,56) (6,56) (1,56)
                   (4,4) (4,45) (5,45) (2,45) (2,2) (2,23) (3,23) (4,23)
                   (6,6) (6,64) (4,64) (2,64) (3,3) (3,31) (1,31) (6,31);

param delta:  1 2 3 4 5 6 :=
1 1           1 0 0 0 0 0
1 12          1 0 0 0 0 0
2 12          0 1 0 0 0 0
5 12          1 1 0 0 0 0
5 5           0 0 0 0 1 0
5 56          0 0 0 0 1 0
6 56          0 0 0 0 0 1
1 56          0 0 0 0 1 1
4 4           0 0 0 1 0 0
4 45          0 0 0 1 0 0
5 45          0 0 0 0 1 0
2 45          0 0 0 1 1 0
2 2           0 1 0 0 0 0
2 23          0 1 0 0 0 0
3 23          0 0 1 0 0 0
4 23          0 1 1 0 0 0
6 6           0 0 0 0 0 1
6 64          0 0 0 0 0 1
4 64          0 0 0 1 0 0
2 64          0 0 0 1 0 1
3 3           0 0 1 0 0 0
3 31          0 0 1 0 0 0
1 31          1 0 0 0 0 0
6 31          1 0 1 0 0 0
;


param volume:=
1 1
2 1
3 1
4 1
5 1
6 1;
```

```
param capacity:=
1 2
2 2
3 2
4 2
5 2
6 2;
```

A.3 AMPL model for problem of Section 5.2

A.3.1 Model

The model is available at: http://perso.enst.fr/~brehon/, as well as some programs to generate the data files based on simple textual network descriptions.

```
#CPLEX command:
#reset;model FAMA_2_LAYER\fama.mod;data FAMA_2_LAYER\carre_diag_aleat.dat;
#option solver cplex;solve;




#------------------------#
#         DONNEES        #
#------------------------#


param Nb_Links >=0;
param Nb_VLinks >= 0; # nb of possible virtual links
param Nb_Demands >= 0;
param Nb_Nodes >= 0;
set LINK := 1 .. Nb_Links; # links
set VLINK := 1 .. Nb_VLinks; # virtual links
set PATH; # paths on the physical graph
set VPATH; # virtual paths on the virtual topology
set DEMAND := 1 .. Nb_Demands; # network demands
set DEMAND_VPATH within (DEMAND cross VPATH);
  # possible combinations of demand assignment to virtual paths.
  # (d,p) exists means demand d can be satisfied by p
```

```
set VLINK_PATH within (VLINK cross PATH);
  # possible combinations of virtual links created by paths.
  # (e,q) exists means v-link e can be created by q
set NODE := 1 .. Nb_Nodes; # network nodes

param delta {DEMAND_VPATH,VLINK} >= 0;
  # if VLINK contributes to to VPATH when DEMAND uses VPATH.
param gamma {VLINK_PATH,LINK} >= 0;
  # if LINK contributes to PATH when VLINK uses PATH.
param ingress {VLINK,NODE} >=0 ; # if VLINK starts at NODE
param egress {VLINK,NODE} >=0 ; # if VLINK ends at NODE
param volume {DEMAND} >= 0; # volume of DEMAND
param capacity {LINK}; # capacity of LINK
param r_in {NODE}; # incoming capacity of IP router NODE
param r_out {NODE}; # outgoing capacity of IP router NODE
param M >= 0; # value of capacity of one bus-LSP




#------------------------#
#       VARIABLES        #
#------------------------#


var Up_Flow {(d,p) in DEMAND_VPATH} >= 0, <= volume[d];
  # volume of d routed over p
var Low_Flow {(e,q) in VLINK_PATH} >= 0;
  # volume of e routed over q
var Active {q in PATH} binary;
  # if q is active
var U {(e,q) in VLINK_PATH} binary;
  # if q is active and creates e
var Vcapacity {e in VLINK} >= 0;
  # capacity of vlink e
var Lambda >= 0; # traffic matrix multiplier
var Nb_Modules {(q,g) in PATH cross LINK} >=0, integer;
  #ceiling of sum(e)gamma_geq.z_eq / M
var Routing_Usage {v in NODE} >= 0;
```

```
var In{v in NODE} >= 0;
var Out{v in NODE} >= 0;




#------------------------#
#       OBJECTIVES       #
#------------------------#


#OBJECTIVE 1: maximize possible traffic
maximize Traf_Mult: Lambda;

#OBJECTIVE 2: minimize cost in routing equipment
#minimize Routing_Power: sum{v in NODE} Routing_Usage[v];
#subject to Traf_Mult: Lambda >=  0.25; #value obtained by obj.1

#OBJECTIVE 3: minimize number of LSPs
#minimize Nb_LSPs: sum {p in PATH} Active[p];
#subject to Traf_Mult: Lambda >= 0.25; #value obtained by obj.1
#subject to Routing_Power: sum{v in NODE} Routing_Usage[v] <= 0.9;
#value obtained by obj.2




#------------------------#
#     CONSTRAINTS        #
#------------------------#


subject to Demand_Satisfaction {d in DEMAND}:
  sum {(d,p) in DEMAND_VPATH} Up_Flow[d,p] >= Lambda * volume[d];
# Constraint: demand satisfaction

subject to VCapa_Def {e in VLINK}:
  Vcapacity[e] <= sum {(e,q) in VLINK_PATH} Low_Flow[e,q];
# Constraint: capacity def. of virtual links
```

```
subject to VLink_Capa {e in VLINK}:
  sum {(d,p) in DEMAND_VPATH} delta[d,p,e] * Up_Flow[d,p] <= Vcapacity[e];
# Constraint: v-link capacity constraint

subject to Link_Capa {g in LINK}:
  sum {(e,q) in VLINK_PATH} gamma[e,q,g] * Low_Flow[e,q] <= capacity[g];
# Constraint: link capacity constraint

subject to Active_Path {(e,q) in VLINK_PATH}:
  Low_Flow[e,q] <= 100000 * U[e,q];
# Constraint: flows only use active paths

subject to In_Traffic {v in NODE}:
  In[v] <= r_in[v];
# Constraint: incoming routed traffic limited

subject to Out_Traffic {v in NODE}:
  Out[v] <= r_out[v];
# Constraint: outgoing routed traffic limited

subject to Routing_def{v in NODE}:
  Routing_Usage[v] = In[v] + Out[v];
# Defining total routing power needed in network

subject to In_Traffic_Def {v in NODE}:
  #In[v] = sum{(e,q) in VLINK_PATH} egress[e,v] * Low_Flow[e,q];
  In[v] = sum{e in VLINK} egress[e,v] * Vcapacity[e];
# Def: incoming routed traffic

subject to Out_Traffic_Def {v in NODE}:
  #Out[v] = sum{(e,q) in VLINK_PATH} ingress[e,v] * Low_Flow[e,q];
  Out[v] = sum{e in VLINK} ingress[e,v] * Vcapacity[e];
# Def: outgoing routed traffic

#subject to No_Low_Split{e in VLINK}:
# sum{(e,q) in VLINK_PATH} U[e,q] <= 1 ;
# Constraint: no lower layer flow splitting
```

```
subject to Activation{(e,q) in VLINK_PATH}:
  U[e,q] <= Active[q];
# Definition: a bus-LSP is active if a VLINK uses it.

subject to Nb_Module_Def1{(q,g) in PATH cross LINK}:
Nb_Modules[q,g] >= sum{(e,q) in VLINK_PATH} gamma[e,q,g]
                   * Low_Flow[e,q] / M;
#subject to Nb_Module_Def2{(q,g) in PATH cross LINK}:
#Nb_Modules[q,g] <= (sum{(e,q) in VLINK_PATH} gamma[e,q,g]
#                    * Low_Flow[e,q] / M )
#                   +1;
#definition of the ceiling function
#warning: <= should really be a <, but cplex refuses it
#however, this last constraint is implicit when minimization is running

subject to Modular{g in LINK}:
sum{q in PATH} Nb_Modules[q,g] * M  <= capacity[g];
#modular allocation of bus-LSPs
```

A.3.2 Model data file

In this example data file, 3 nodes are linked by 2 unidirectional links of capacity 1 and module 1, such as in Figure 3.4. There are 3 demands, using a volume 0.4 each. Demands may be routed in the intermediate node in the upper layer or switched in the lower layer, depending on which bus-LSPs are setup.

```
param Nb_Links   := 2;
param Nb_Demands := 3;
param Nb_VLinks  := 3;
param Nb_Nodes   := 3;
param M          := 1;
set PATH         := 1_2 2_3 1_2_3 ;
set VPATH        := 1_2 1_3 2_3 1_2_3 ;
set DEMAND_VPATH := (1,1_2) (2,1_3) (3,2_3) (2,1_2_3) ;

set VLINK_PATH   := (1,1_2) (2,2_3) (1,1_2_3) (3,1_2_3) (2,1_2_3) ;

# Association of link number to nodes:
# 1 to node 2 : 1
```

```
# 2 to node 3 : 2

# Association of v-link number to nodes:
# 1 to node 2 : 1
# 2 to node 3 : 2
# 1 to node 3 : 3

# Association of demand number to nodes:
# 1 to node 2 : 1
# 1 to node 3 : 2
# 2 to node 3 : 3


#   e1 e2
# [d1,*,*]
# p1

param delta:=

[1,*,*] : 1 2 3 :=
1_2       1 0 0

[2,*,*] : 1 2 3 :=
1_3       0 0 1
1_2_3     1 1 0

[3,*,*] : 1 2 3 :=
2_3       0 1 0
;


#   g1 g2
# [e1,*,*]
# q1

param gamma:=

[1,*,*] : 1 2 :=
1_2       1 0
```

```
1_2_3     1 0

[2,*,*] : 1 2 :=
2_3       0 1
1_2_3     0 1

[3,*,*] : 1 2 :=
1_2_3     1 1
;


#ligne: noeuds
#colonne: vlinks
param ingress : 1 2 3 :=
1               1 0 0
2              0 1 0
3               1 0 0
;

#ligne: noeuds
#colonne: vlinks
param egress : 1 2 3 :=
1              0 1 0
2              0 0 1
3              0 0 1
;
param r_in :=
1 2
2 2
3 2
;
param r_out :=
1 2
2 2
3 2
;

param volume:=
1 0.4
```

```
2 0.4
3 0.4
;

param capacity:=
1 1
2 1
;
```

A.4 AMPL model for problem of Section 5.4

In this section, we present the files used to generate the *theoretical limit* values of Section 5.4. The model is available at: http://perso.enst.fr/~brehon/ as well as some programs to generate the data files based on simple textual network descriptions.

A.4.1 Model

```
#CPLEX command:
#reset;model optim_fama3.mod;data carre_aleat.dat;
#option solver cplex;solve;




#------------------------#
#     GIVEN              #
#------------------------#


param Nb_Links >=0;
param Nb_Demands >= 0;
set LINK := 1 .. Nb_Links; # links
set PATH; # paths in graph
set DEMAND := 1 .. Nb_Demands; # network demands
set DEMAND_PATH within (DEMAND cross PATH);
  # possible combinations of demand assignment to paths

param delta {DEMAND_PATH,LINK} >= 0;
  # if LINK contributes to PATH when DEMAND uses PATH.
```

```
param volume {DEMAND} >= 0; # volume of DEMAND
param capacity {LINK}; # capacity of LINK




#------------------------#
#         VARIABLES      #
#------------------------#


var Flow {(d,p) in DEMAND_PATH} >= 0; # volume of d running over p
var lambda >=0; # traffic matrix multiplier




#------------------------#
#      OBJECTIVE         #
#------------------------#


maximize Lambda_var: lambda;
#maximize traffic multiplier




#------------------------#
#       CONSTRAINTS      #
#------------------------#


subject to Link_Capa {e in LINK}:
  sum {(d,p) in DEMAND_PATH} delta[d,p,e] * Flow[d,p] <= capacity[e];
# Constraint: link capacity constraint

subject to Demand_Satisfaction {d in DEMAND}:
  sum {(d,p) in DEMAND_PATH} Flow[d,p] = lambda * volume[d];
```

```
# Constraint: demand satisfaction
```

A.4.2 Model data file

This should be the same type of data file as used for the model presented in Appendix A.2.

Bibliography

[ACG05] M. Ali, G. Chiruvolu, and A. Ge, *Traffic Engineering in Metro Ethernet*, IEEE Network (2005), pp. 10–17. 2.3

[AKV06] A. Ayyangar, K. Kompella, and J.P. Vasseur, *Label Switched Path Stitching with Generalized MPLS Traffic Engineering*, draft-ietf-ccamp-lsp-stitching-04.txt (2006). 6.2.3.4, 6.2.3.4, 6.2.3.5, 6.2.3.6

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993. 1.4

[BBPF06] Lou Berger, Igor Bryskin, Dimitri Papadimitriou, and Adrian Farrel, *Gmpls based segment recovery*, draft-ietf-ccamp-gmpls-segment-recovery-03.txt (2006). 6.2.3.4

[BC07] Yannick Brehon and Ramon Casellas, *Extensions to RSVP-TE for bus-LSP support*, IETF draft to be submitted, 2007. 6.2.4, 7.1

[BHS05] S. Balasubramanian, Wensheng He, and A.K. Somani, *Light-Trail Networks: Design and Survivability*, Local Computer Networks, 30th IEEE International Conference on, November 2005, pp. 174–181. 3.2.3.1

[BK06] Y. Brehon and D. Kofman, *Bus-Label Switched Paths, an approach to reduce the cost of multilayer networks*, Communications, 2006 IEEE International Conference on (ICC 2006), **5** (2006), pp. 2401–2406. 7.1

[BKC07] Y. Brehon, D. Kofman, and A. Casaca, *Virtual Private Network To Spanning Tree Mapping*, IFIP Networking Conference, May 2007. 7.1, 7.2

[BKPD07] Y. Brehon, D. Kofman, M. Pióro, and M. Diallo, *Optimal Virtual Topology Design using Bus-Label Switched Paths*, Selected Areas in Communications, Special issue on Traffic Engineering for Multi-Layer Networks, IEEE Journal on (JSAC), **25** (2007), pp. 1001–1010. 7.1, 7.2

[BPD+04] N. Bouabdallah, G. Pujolle, E. Dotaro, N. Le Sauze, and L. Ciavaglia, *Distributed Aggregation in All-Optical Wavelength Routed Networks*, Communications, 2004 IEEE International Conference on (ICC2004), vol. 3, June 2004, pp. 1806–1810. 3.2.3.2, 2, 4, 7.2

[BSK05] S. Balasubramanian, A.K. Somani, and A.E. Kamal, *Sparsely hubbed light-trail grooming networks*, Computer Communications and Networks, 14th International Conference on, October 2005, pp. 249–254. 3.2.3.1

[CMP+03] J. Comellas, R. Martinez, J. Prat, V. Sales, and G. Junyent, *Integrated IP/WDM routing in GMPLS-Based Optical Networks*, Network, IEEE **17** (2003), no. 2, pp. 22–27. 1.4

[CPLEX] ILOG, *CPLEX and AMPL optimization packages*, on-line, www.cplex.com. (document), 2.4.2, 5.4.1, 5.5.1

[Dij59] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik **1** (1959), pp. 83–89. (document), 2.5.1, 5.3.1, 6.1.1

[DR00] R. Dutta and G.N. Rouskas, *A survey of virtual topology design algorithms for wavelength routed optical networks*, Optical Networks Mag. **1** (2000), no. 1, pp. 73–89. 1.4

[FHS04] Jing Fang, Wensheng He, and A.K. Somani, *Optimal light trail design in WDM optical networks*, Communications, IEEE International Conference on, vol. 3, June 2004, pp. 1699–1703. 3.2.3.1

[FL98] Gang Feng and Zemin Liu, *Dynamic routing algorithms in ATM networks*, Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on, vol. 6, May 1998, pp. 468–501. 1.4

[GC03] A. Gumaste and I. Chlamtac, *Light-trails: a novel conceptual framework for conducting optical communications*, High Performance Switching and Routing, Workshop on, June 2003, pp. 251–256. 3.2.3.1

[GM02] A. Gencata and B. Mukherjee, *Virtual-topology adaptation for WDM mesh networks under dynamic traffic*, INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, vol. 1, June 2002, pp. 48–56. 1.3.2.1, 1.4.2

[GS95] O. Gerstel and A. Segall, *Dynamic maintenance of the virtual path layout*, INFOCOM '95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, April 1995, pp. 330–337. 1.3.2.1, 1.4

[GSM02] A. Gencata, L. Sahasrabuddhe, and B. Mukherjee, *Virtual-topology adaptation with minimal lightpath change for dynamic traffic in WDM mesh networks*, Optical Fiber Communication Conference and Exhibit, 2002, March 2002, pp. 783–784. 1.3.2.1, 1.4.2

[HZC06] X. He, M. Zhu, and Q. Chu, *Traffic Engineering for Metro Ethernet Based on Multiple Spanning Trees*, Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, International Conference on, ICN/ICONS/MCL 2006, 2006, p. 97. 2.3

[IEE98] IEEE, *Media Access Control Bridges*, IEEE 802.1d (1998). 2.1.2

[IEE01] IEEE, *Rapid Spanning Tree Configuration*, IEEE 802.1w (2001). 2.1.2

[IEE02] IEEE, *Multiple Spanning Trees*, IEEE 802.1s (2002). (document), 2.1.2

[IEE03] IEEE, *Virtual Bridged Local Area Networks*, IEEE 802.1q (2003). 2.1.1

[IEE06a] IEEE, *Provider Backbone Bridges, Draft 3.3*, IEEE 802.1ah (2006). 1.2, 2.1.3

[IEE06b] IEEE, *Provider Bridges*, IEEE 802.1ad (2006). 1.2, 2.1.3

[KKL00] K. Kar, M. Kodialam, and T. V. Lakshman, *Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications*, Selected Areas in Communications, IEEE Journal on **18(12)** (2000), pp. 2566 – 2579. (document), 5.3.2, 5.3.2, 5.3.2

[KL01] M. Kodialam and T.V. Lakshman, *Integrated dynamic IP and wavelength routing in IP over WDM networks*, INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, April 2001, pp. 22–26. 1.4

[KS01] R.M. Krishnaswamy and K.N. Sivarajan, *Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers*, Networking, IEEE/ACM Transactions on **9** (2001), no. 2, pp. 186–198. 1.4.1

[LA91] J-F.P. Labourdette and A.S. Acampora, *Logically rearrangeable multihop lightwave networks*, Communications, IEEE Transactions on **39** (1991), no. 8, pp. 1223–1230. 1.4.1

[LGCS02] S.K. Lee, D. Griffith, V. Coussot, and D. Su, *Explicit routing with QoS constraints in IP over WDM*, Communications, IEEE Proceedings **149** (2002), no. 2, pp. 83–91. 1.4

[LHA94] J-F.P. Labourdette, G.W. Hart, and A.S. Acampora, *Branch Exchange Sequences for Reconfiguration of Lightwave Networks*, Communications, IEEE Transactions on **42** (1994), no. 10, pp. 2822–2832. 1.4.1

[LRP06] J.P. Lang, Y. Rekhter, and D. Papadimitriou, *Rsvp-te extensions in support of end-to-end gmpls-based recovery*, draft-ietf-ccamp-gmpls-recovery-e2e-signaling-04.txt (2006). 6.2.3.5, 6.2.3.5, 6.2.4, 6.2.4.1, 6.2.4.2, 6.2.4.3

[MBRM96] B. Mukherjee, D. Banerjee, S. Ramamurthy, and A. Mukherjee, *Some principles for designing a wide-area WDM optical network*, Networking, IEEE/ACM Transactions on **4** (1996), no. 5, pp. 684–696. 1.4.1

[Mor06] T. Morin, *Requirements for Multicast in L3 Provider-Provisioned VPNs*, draft-ietf-l3vpn-ppvpn-mcast-reqts-10.txt (2006). 2.1.3

[MR07] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, Morgan Kaufmann, March 2007. 1.2

[NTLM02] A. Narula-Tam, P.J. Lin, and E. Modiano, *Efficient routing and wavelength assignment for reconfigurable WDM networks*, Selected Areas in Communications, IEEE Journal on **20** (2002), no. 1, pp. 75–88. 3.1.1

[OSK+03] E. Oki, K. Shiomoto, M. Katayama, W. Imajuku, and N. Yamanaka, *Performance of dynamic multi-layer routing schemes in IP+optical networks*, High Performance Switching and Routing, 2003, HPSR. Workshop on, June 2003, pp. 233–238. 1.3.2.1, 1.4.2

[Pap04] D. Papadimitriou, *Generalized MPLS Signaling for Layer-2 Label Switched Paths (LSP)*, draft-papadimitriou-ccamp-gmpls-l2sc-lsp-03.txt (2004). (document), 1.1, 2.1.2, 3.1.1

[PM04] M. P. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*, Morgan Kaufmann, July 2004. 1.4, 5.2.2, 5.5.1

[PNM+05] M. Padmaraj, S. Nair, M. Marchetti, G. Chiruvolu, M. Ali, and A. Ge, *Metro Ethernet Traffic Engineering Based on Optimal Spanning Trees*, Wireless and Optical Communications Networks, Second IFIP International Conference on, WOCN, 2005, pp. 568–572. 2.3, 2.6

[RFC1058] C. Hedrick, *Routing Information Protocol (RIP)*, Internet RFC 1058 (1988). 6.1.1

[RFC2328] J. Moy, *OSPF Version 2*, Internet RFC 2328 (1998). 3.1.1, 6.1.1

[RFC2370] R. Coltun, *The OSPF Opaque LSA Option*, Internet RFC 2370 (1998). 6.1.1

[RFC2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *Requirements for Traffic Engineering Over MPLS*, Internet RFC 2702 (1999). 3.1.1

[RFC3031] E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, Internet RFC 3031 (2001). 3.1.1, 3.2.1

[RFC3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, *RSVP-TE: Extensions to RSVP for LSP Tunnels*, Internet RFC 3209 (2001). 1.3.2.1, 6.2.1, 6.2.2.1, 6.2.3.2, 6.2.3.2, 6.2.3.3, 6.2.3.4, 6.2.3.6

[RFC3471] L. Berger, *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description*, Internet RFC 3471 (2003). 1.3.2.1, 2.1.2, 6.2.3.4

[RFC3472] P. Ashwood-Smith and L. Berger, *Generalized Multi-Protocol Label Switching (GMPLS) Signaling - Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions*, Internet RFC 3472 (2003). 1.3.2.2

[RFC3473] L. Berger, *Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions*, Internet RFC 3473 (2003). (document), 1.3.2.2, 6, 6.1.3.2, 6.2.1, 6.2.4.1, 6.2.4.3

[RFC3630] D. Katz, K. Kompella, and D. Yeung, *Traffic Engineering (TE) Extensions to OSPF Version 2*, Internet RFC 3630 (2003). (document), 1.3.2.2, 6

[RFC3945] E. Mannie, *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*, Internet RFC 3945 (2004). 1.3.2.1

[RFC3985] Bryant and Pate, *Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture*, Internet RFC 3985 (2005). 1.2

[RFC4026] L. Andersson and T. Madsen, *Provider Provisioned Virtual Private Network (VPN) Terminology*, Internet RFC 4026 (2005). 1.2

[RFC4202] Y.Rekhter K.Kompella, *Routing Extensions in Support of Generalized Multi-Protocol Label Switching* , Internet RFC 4202 (2005). 6

[RFC4206] K. Kompella and Y. Rekhter, *LSP Hierarchy with Generalized MPLS TE*, Internet RFC 4206 (2005). 3.2.1, 6.1.3.2, 6.2.3.4

[RFC4447] L. Martini, E. Rosen, N. El-Aawar, T. Smith, and G. Heron, *Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, Internet RFC 4447 (2006). 1.2

[RFC4655] A. Farrel, J.-P. Vasseur, and J. Ash, *A Path Computation Element (PCE)-Based Architecture*, Internet RFC 4655 (2006). 1.3.2.2, 1.4.2

[RFC4664] L. Andersson and E. Rosen, *Framework for Layer 2 Virtual Private Networks (L2VPNs)*, Internet RFC 4664 (2006). 1.2

[RR00] B. Ramamurthy and A. Ramakrishnan, *Virtual topology reconfiguration of wavelength-routed optical WDM networks*, Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE, vol. 2, November 2000, pp. 1269–1275. 1.4

[SH80] J. F. Shoch and J. A. Hupp, *Measured Performance of an Ethernet Local Network*, Communications of the ACM **23(12)** (1980), pp. 711 – 721. 1.1

[SIL+98] R. Sabella, E. Iannone, M. Listanti, M. Berdusco, and S. Binetti, *Impact of transmission performance on path routing in all-optical transport networks*, Journal on Selected Areas of Communications, JSAC IEEE **6** (1998), no. 2, pp. 1617–1622. 2.6, 5.4.2.3

[SL03] M.E.M. Saad and Zhi-Quan Luo, *Reconfiguration with no service disruption in multifiber WDM networks based on Lagrangean decomposition*, Communications, 2003. ICC '03. IEEE International Conference on, vol. 2, May 2003, pp. 1509–1513. 1.3.2.1, 1.4.1

[SOI+03] K. Shiomoto, E. Oki, W. Imajuku, S. Okamoto, and N. Yamanaka, *Distributed virtual network topology control mechanism in GMPLS-based multiregion networks*, Selected Areas in Communications, IEEE Journal on **21** (2003), no. 8, pp. 1254–1262. 1.4.2

[SPM01] N. Sreenath, G.R. Panesar, and C.S.R Murthy, *A two-phase approach for virtual topology reconfiguration of wavelength-routed WDM optical networks*, Networks, 2001. Proceedings. Ninth IEEE International Conference on, October 2001, pp. 371–376. 1.4

[SPR+06] K. Shiomoto, D. Papadimitriou, J.L. Le Roux, M. Vigoureux, and D. Brungard, *Requirements for GMPLS-based multi-region and multi-layer networks (MRN/MLN)*, draft-ietf-ccamp-gmpls-mln-reqs-02.txt (2006). 6.1.3.2

[SS03] J. Stosic and B. Spasenovski, *Practical models for design and reconfiguration of virtual topology in optical transport networks*, Telecommunications in Modern Satellite, Cable and Broadcasting Service, 2003. TELSIKS 2003. 6th International Conference on, vol. 1, October 2003, pp. 67–70. 1.4

[TZJT02] H. Takagi, Y. Zhang, X. Jia, and H. Takagi, *Virtual Topology Reconfiguration for Wide-Area WDM Networks*, Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on, vol. 1, June 2002, pp. 835–839. 1.4, 1.4.1

[VRY+06] J.P. Vasseur, J.L. Le Roux, S. Yasukawa, S. Previdi, P. Psenak, and P. Mabey, *IGP Routing Protocol Extensions for Discovery of Traffic Engineering Node Capabilities* , draft-ietf-ccamp-te-node-cap-04.txt (2006). 6.1.4

[VTHD] Réseau National de Recherche en Télécommunications, *Vraiment Très Haut Débit, http://www.vthd.org*, Ministère de l'Économie, des Finances et de l'Industrie de France. 2.6, 4.3, 5.4.2.2

[XST03] S. Xu, K. Sezaki, and Y. Tanaka, *A Heuristic Method of Logical Topology Reconfiguration in IP/WDM Optical Networks*, 10th International Conference on Telecommunications (ICTŠ2003), February 2003. 1.3.2.1, 1.4, 1.4.2

[YKH95] A. Yamashita, R. Kawamura, and H. Hadama, *Dynamic VP rearrangement in an ATM network*, Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE, vol. 2, November 1995, pp. 1379–1383. 1.3.2.1, 1.4

[YKS+02] N. Yamanaka, M. Katayama, K. Shiomoto, E. Oki, and N. Matsuura, *Multi-layer traffic engineering in photonic-GMPLS-router networks*, Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, vol. 3, November 2002, pp. 2731–2735. 1.4.1

[YR02] X. Yang and B. Ramamurthy, *An Analytical Model for Virtual Topology Reconfiguration in Optical Networks and A Case Study*, Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on, October 2002, pp. 302–308. 1.4

[Zim80] H. Zimmermann, *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*, Communications, IEEE Transactions on, **28(4)** (1980), pp. 425–432. (document), 1.1, 3.1.1

[ZLYHG02] L. Zhang, K. Lee, C-H. Youn, and H-G.Yeo, *Adaptative Virtual Topology Reconfiguration Policy Employing Multi-stage Traffic Prediction in Optical Internet*, High Performance Switching and Routing (2002). 1.3.2.1, 1.4, 1.4.1