Universitat de Girona

LABEL SPACE REDUCTION IN GMPLS AND ALL-OPTICAL LABEL SWAPPING NETWORKS

Fernando SOLANO DONADO

ISBN: 978-84-691-1271-7
Dipòsit legal: GI-3-2008

Label Space Reduction in GMPLS and All-Optical Label Swapping Networks

Fernando Solano Donado

Advisors: PhD. Ramón Fabregat and PhD. José Marzo

Doctorate Program in *Tecnologías de la Información*

Departament d'Arquitectura i Tecnología de Computadors

Broadband Communications and Distributed Systems group

Universitat de Girona

Girona, Catalonia, Spain

September 10, 2007

Contents

* **Chapters, sections and subsections containing contributions of the thesis are outlined by adding a superscript asterisk (*) at the end of their titles.**

List of Figures

List of Tables

Acknowledgments

I would like to thanks first at all my thesis directors, PhD. Fabregat and PhD. Marzo, for their complementary work in all aspects regarding my research: from the grant and all economical support I have received so far, to the tiniest missed dot in any article

Thanks to the Department of Universities, Research and Information Society (DURSI) of the Government of Catalonia and the European Social Funds for the given grants: the 4-years third cycle grant *Formació de personal Investigador* (FI), and the two 6-months visit *Beques per a Estades per a la recerca fora de Catalunya* (BE). Moreover, thanks to the COST project action number 293: *Graphs & Algorithms in Communication Networks*, for their financial and scientific support.

Thanks to all the professors who have collaborated with me along my third cycle studies. Specially those named below who I own a big knowledge-debt.

Thanks to PhD. Yezid Donoso at *Universidad del Norte* for all his time, knowledge and patience at the beginning of my studies. Collaborating with him was the "shortest path" to find my way.

Thanks to PhD. Thomas Stidsen at *Danish Technical University* for all the mathematical modeling background he taught me while my visit, knowledge that is certainly needed for most of my thesis contributions.

Thanks to Jean-Phillipe Vasseur in *Cisco Systems* and PhD. Jaudelice de Oliveira at *Drexel University* for all their feedback concerning my contributions in MPLS and RSVP-TE. Thanks to them my contributions on MPLS went far away from becoming an useless dream, and got closer to today's Internet reality.

Thanks to PhD. Jaudelice de Oliveira again and PhD. Timothy Kurzweg at *Drexel Univerity* for all their valuable comments in the G^+ architecture. They played an important role in the most important publications concerning this thesis.

Thanks to PhD. Didier Colle and Ruth Van Caenegem at *University of Ghent* for all the background and feedback given with regards to AOLS.

Special thanks to Jau, Lluis, Cayita, Chivis and Luisfer for their special psychological support in preventing me from getting either lazy or crazy through all these years.

Thanks to some friends abroad, who supported me abroad; namely: Beatriz Florian, Rebekka Rost, Anbu, Sukrit Dasgupta, Michele Conti, and my beloved Marysia.

Thanks to God and my family, who I own most of what I have become through the years... therefore, most of my achievements.

Thesis Outlines

The evolution of computer networks in the Internet has propelled Optical Transport Networks (OTN) in recent years. While the optical switching granularity has evolved from fibers to wavelengths to bursts to packets with very promising designs, fully-optical forwarding functions are still a newly born technology. In other words, although information optical codification and transmission has been successfully achieved, the bottleneck on OTNs has been foreseen to be in the matter how forwarding and processing are performed at each node in the network.

With the recent deployment of All-Optical Flip-Flop (AOFF) and All-Optical Logic XOR Gate (AOLXG), full Optical Packet Switching (OPS) (and Optical Burst Switching (OBS) as well) using All-Optical Label Swapping (AOLS) is closer to become a reality nowadays. With AOLS, OTN technologies will not only perform traffic switching completely optically at different granularity, but they will be capable of performing basic forwarding functions in the optical domain as well. Although AOLS speeds up dramatically optical switching, it is expensive. The cost of deploying AOLS grows linearly with the number of connections - *viz.* labels - that the network is able to support. Clearly, this raises a scalability problem.

This dissertation presents contributions of the author concerning the reduction of the cost for deploying AOLS. Even though AOLS cost is tied to its optical physical devices cost, the here presented contributions do not aim at proposing new optical physical devices with a reduced cost. Instead, several methods for using these devices efficiently are proposed.

This chapter gives an overview to the tackled problem and the whole thesis itself. The chapter is organized as follows. Initially, a recount of the evolution of OTN technologies is given, ending with the AOLS systems and its drawbacks. Once the problem has been stated, the objectives and contributions of this dissertation are listed. Finally, a section is devoted to describe the organization of the rest of the document.

[A] Motivation for Reducing Label Spaces in AOLS Networks

The increasing requirements on packet networking have motivated the development and deployment of complex network systems, which in turn required the development of sophisticated architectures. For instance, the drastic increase in bandwidth, quality of service, and multiple play service requirements motivated

the separation of the control and data planes.

Technologies concerning the data plane have evolved during the last years, and OTN technologies have been leading the field. OTNs use the Wavelength-Division-Multiplexing (WDM) switching architecture, making it the main trend for the next generation optical Internet. Unfortunately, WDM is capable of switching traffic only at a wavelength granularity. As a consequence, subwavelength switching must be deployed with the addition of traditional electronic switches (relatively slow) or with recently evolving optical technologies, such as OPS and OBS.

The control plane of OTNs technologies is foreseen to be driven by the Generic Multi-Protocol Label Switching (GMPLS) protocol, and many standards have been settled up to that point so far. The tendency of adopting GMPLS as the protocol leading the control plane had motivated many vendors to deployed most of GMPLS functionalities directly "burn-in" chip, as a way to speed up forwarding. In the case of OTN technologies this implies the deployment of hardware capable of managing labels in the optical domain; name it AOLS technologies.

As discussed at the end of this chapter, AOLS is expensive. Indeed, the capital expenditures of a network using AOLS (and its complexity) grow linearly with the number of labels needed to route the traffic in the network. Therefore, it is clear that reducing the number of labels used is completely desirable by any Internet Service Provider (ISP).

This chapter is devoted to introduce the reader to the technologies discussed in this dissertation. The chapter follows the timeline of the evolution of OTN technologies, ending it with the technology that concerns the most this document's contributions: AOLS.

Today's Mostly Deployed Solution: Lightpaths

Traditional Wavelength-Routing Switches (WRS) are still the most reliable technology used by ISPs nowadays. WRSs take advantage of the multiplexing capabilities of optical fibers to more efficiently route demands, a feature called WDM. With WDM, an optical fiber can be multiplexed into hundreds of wavelengths. Each wavelength capacity is equivalent to OC-192 (around 10 Gbps). By multiplexing, WRSs are capable of optically switch wavelengths from one fiber to another using a Photonic Cross-Connect (PXC), enabling the configuration of optical routes between any pair of WRSs in the network. These optical routes are named *lightpaths* [Dix03]. The interconnection of these optical elements can be seen in the upper part of Fig. 1. Fig. 1 also shows the forwarding of several lightpaths and demands, which will be described later.

Usually, a lightpath forwards more than one demand. This is motivated by: *a*) the number of wavelengths in a fiber being too small compared to the number of demands in the network, and *b*) the capacity of a wavelength being too large when routing a single customer demand. In this sense, customer demands are said to be subwavelength demands.

To improve resource utilization, a lightpath forwards many demands at the same time, and a demand is forwarded using consecutive lightpaths. In this context, it is said that subwavelengths demands are *groomed* into lightpaths [SSM06].

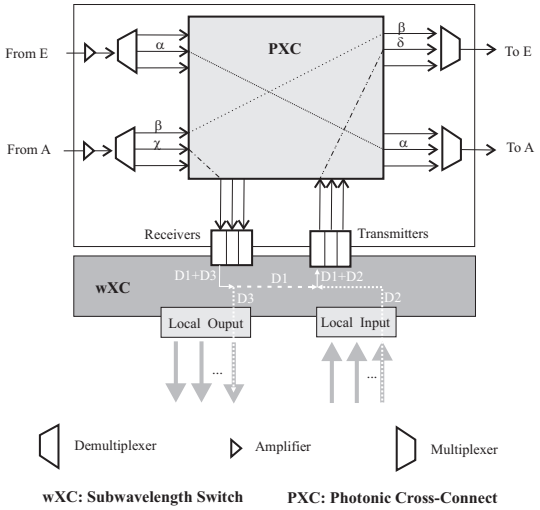


Figure 1: Classical WDM Architecture (node F)

When a groomed demand needs to be switched from one lightpath to another, all the lightpath demands need to be differentiated, processed and forwarded accordingly. Consider Fig. 2 as an example. Fig. 2(a) illustrates a mesh network of 6 WRSes connected physically by 8 bidirectional fiber links. These fiber links are used for setting up 6 lightpaths, shown in the same figure and described at the bottom. Fig. 2(b) mirrors the virtual topology resulting from the lightpaths in Fig. 2(a).

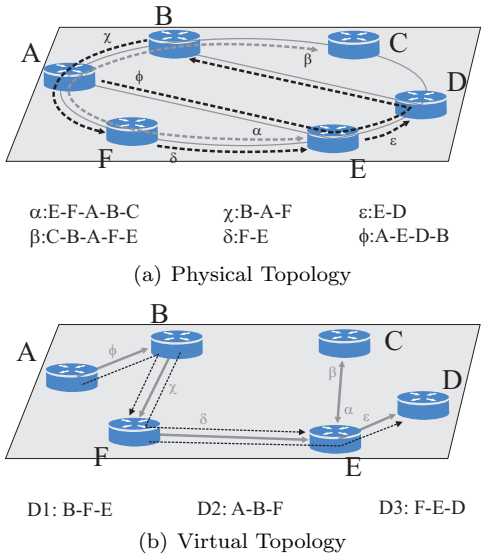
The subwavelength demands described in Table 1 are routed over the virtual topology: *D1* from B to E using lightpaths χ and δ, *D2* from F to D using lightpaths δ and ϵ, and *D3* from A to F using lightpaths ϕ and χ.

In this example, two lightpaths are used to forward each demand, and some lightpaths (e.g. δ) are used to forward more than one demand. In this case, WRS F needs to switch subwavelength traffic between lightpaths so demand *D1* and *D3* are groomed and forwarded to WRS E, and demand *D2* is dropped in the local network.

Henceforth, the term Subwavelength Cross-Connect or Subwavelength Switch (wXC) denotes a switching device capable of cross connecting and grooming subwavelength demands. A wXC can be implemented electronically or optically. When an Electronic Switch (EXC) is used , three expensive steps need to be performed in order to switch subwavelength demands. First, the optical signal needs to be converted in electronic packets (by means of a receiver). Second, a forwarding protocol in the EXC decides which is the next lightpath that the packets need to be forwarded to. Finally, the electronic packets are converted

Demand	From	To	Lightpath	
D1	B	E	$\chi = B \rightarrow A \rightarrow F$	$\delta = F \rightarrow E$
D2	F	D	$\delta = F \rightarrow E$	$\epsilon = E \rightarrow D$
D3	A	F	$\phi = A \rightarrow E \rightarrow D \rightarrow B$	$\chi = B \rightarrow A \rightarrow F$

Table 1: Subwavelength demand routing using Lightpaths



(a) Physical Topology

(b) Virtual Topology

Figure 2: Example of Lightpaths Configuration

back to optical signals (by means of a transmitter). This process is known as Opto-Electro-Optical conversion (OEO).

In Fig. 1, the wavelength and subwavelength switching of WRS at node F can be seen. WRS F handles 4 lightpaths (α, β, χ and δ) whose paths can be seen earlier in Fig. 2(a). Note that the wavelengths used by lightpaths α and β are optically switched between fibers using the PXC. Transmitters and receivers perform the OEO conversions between the PXC and the EXC. As mentioned before, the EXC of WRS F is in charge of switching subwavelength demands *D1*, *D2* and *D3* between lightpaths (χ and δ) and the low-speed local ports.

Optical Subwavelength Switching

The use of OEO in OTNs is undesirable because: *a*) they rely on electronic switches (which are, in comparison to wavelength switching, much slower), and *b*) they need transceivers (laser transmitters and optical receivers) to perform such conversions (increasing the capital expenditures of the network). These facts have encouraged researchers to perform subwavelength switching completely in the optical domain, as in OPS, OBS, Optical Burst Transport, and Packet Slot Routing [Dix03].

In this section, the two most known and promising technologies for optical subwavelength switching are explained.

Optical Packet Switching

OPS provides the finest routing granularity among all OTN technologies, at a packet-by-packet basis. Because of its fine routing granularity, optical packets must by differentiated by means of individual headers. Packet headers must be read by the optical switch in order to re-tune PXC ports, achieving optical packet forwarding. OPS is basically implemented with a fast PXC (fast enough

to re-tune PXC ports between packet arrivals) or an Arrayed Waveguide Grating (AWG).

Typically, OPS networks are slotted, i.e. all the packets have the same size. A fixed size time slot contains both the payload and the header. The time slot has a longer duration than the packet to provide guard bands.

Since PXC input/output ports can be set up incrementally (one by one) or jointly (a set of them together), it is possible to switch many incoming packets at the same time, or to switch each packet individually on the fly. In both cases, a bit-level synchronization and fast clock recovery are necessary for packet header recognition and packet delineation. Therefore, all the input packets arriving at the input ports need to be aligned in phase with one another before entering the PXC. The synchronization is performed by means of a set of Fiber Delay Lines (FDL).

After the synchronization has been made, a tap splits a small amount of power from the incoming packets for the header processing. The header processing circuits recognize a preamble at the beginning of the packet and then read the header information. It also passes the timing information of the incoming packet to the control plane to configure the synchronization stages and the PXC.

If the header is processed electronically, FDLs must be employed to delay the payload long enough to take the routing decision and configure the PXC. It should be emphasized that the greatest experienced delay while forwarding is caused because of the header processing, nowadays.

Optical Burst Switching

OBS is an approach that attempts to shift the computation and control complexity from the optical domain to the electrical domain, from the core to the edge of the network. OBS has the switching granularity between a circuit and a packet. A burst consists, then, of a set of packets adding up from tens of kilobytes to few megabytes long.

Before transmitting a data burst (carrying the payload), a control packet is initially sent. The control packet aims at configuring the switches (reserving resources and interconnecting input/output ports) as it propagates along. In this way, the data burst is never delayed nor processed.

Nevertheless, since the control packet processing and resources seizing may take some time at each hop, an offset time is considered just after the control packet is sent. The offset time should be at least long enough to let *all* the involved nodes process the control packet and configure themselves.

Tomorrow's Challenge: All-Optical Label Swapping

The Role of Header Processing

GMPLS is a protocol providing tag (or label) switching of information regardless the link and routing protocol involved [Man04]. Because of its capability for isolating the control plane from the data plane, GMPLS is the most promising option to drive the control plane of any of these OTNs technologies.

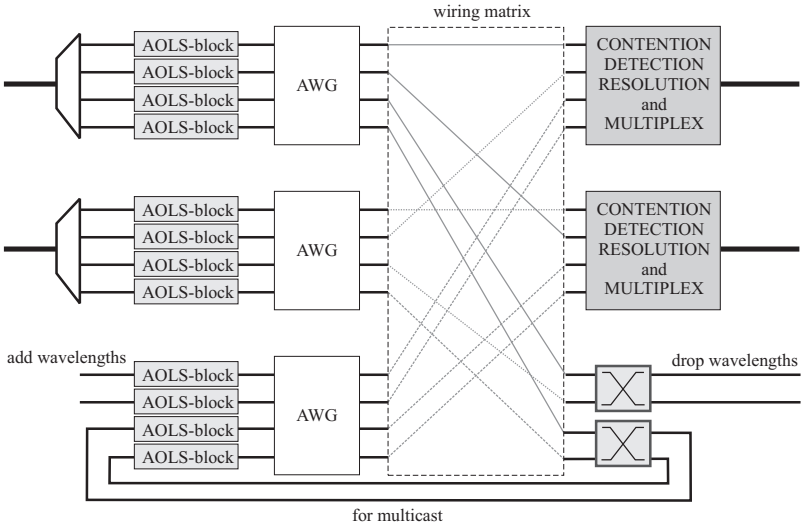


Figure 3: OPS architecture using AOLS

Considering a control plane driven by GMPLS, OPS packet headers (or OBS control packets) must encode a *label* in order to provide sufficient information about the packet (or burst) to the control plane. GMPLS labels have local meaning and would be used for *routing* and *forwarding* of optical packets and/or bursts.

Once a packet arrives to a node and its (incoming) label has been extracted, *routing* involves the decision (or computing) of the new (outgoing) label for the packet, and the decision of which fiber/wavelength is going to be used to forward the packet out of the switch. These decisions are taken using an internal lookup table and the current incoming label. Depending on its implementation, the incoming wavelength, and/or fiber port can be considered as well.

Complementary, *forwarding* involves rewriting the incoming label with the outgoing label, physically converting the labeled packet (or burst) to the new wavelength, and switching the packet from one optical fiber port to another. Other actions are taken as well in the forwarding process, such as buffering mechanisms and content resolution [BOR+00].

These two operations (routing and forwarding) should be taken quickly; the faster, the better the performance of the architecture is. In the former (OPS), the faster the packet header can be processed, the shorter guard bands can be; hence, enhancing the performance of the architecture. Moreover, less FDLs are needed. In the later (OPS), the faster the control packet is processed, the less offset time are required; hence, enhancing the performance of the architecture too.

All-Optical Label Swapping

While *forwarding* speed depends on the quality of the involved physical components (e.g. wavelength converters speed, PXC, etc.), *routing* requires information computing and decision taking making it dependable on how the header information (specially the label) is processed. It is clear that in order to *forward* a

packet, *routing* decisions must be taken first.

The processing of optical signals using pure-optical device is very limited yet. As a consequence, although labels were optically coded and transmitted, the first approaches taken were using fast electronic processing at every node [BOR+00, MCE+00]. This is, labels were converted to electronic information to get processed and then recoded again as optical signals, similarly as the OEO conversion process for subwavelength traffic[1].

Nowadays, to the best of our knowledge, there exist only one architecture capable of performing label processing completely optically, *viz.* AOLS. The architecture was developed under the All-Optical LAbel SwApping employing optical logic Gates in NEtwork nodes (LASAGNE) project [RKM+05, CMC+06, CCPD06] and can be seen in Fig. 4.

In LASAGNE, each demultiplexed wavelength is fed to an AOLS block that is in charge of extracting, processing and rewriting of the incoming optical label. Each AOLS block is comprised of a set of optical correlators based on AOLXGs [MRM+02], performing the comparison between the incoming label and a set of local addresses fixed in the node. These local addresses are generated using Optical Delay Lines (ODL), where each ODL generates a bit sequence out of one pulse.

Considering the limited functionality of a single AOLXG, comparing the incoming label to the local addresses implies that *for each possible incoming label a separate ODL and a AOLXG-based correlator have to be installed in the AOLS block* [RKM+05].

After the label has been identified, a single intensity pulse will appear at the output of the AOLXG correlator, the one matching the address. This pulse feeds both the new label generation block and the control block. The former contains a set of ODLs that generates the new label. In case of OPS, the new label is inserted in front of the payload, making the packet content ready to forward. The control block is made-up of AOFFs [DHL+03]. Depending on the matching address, the appropiate flip-flop will emit a continuous wave signal at a certain wavelength. The continuous wave feeds the tunable wavelength converter to change the packet to the correct wavelength.

Up to this point, the reader could have noticed that AOLS have notorious scalability problems. To perform AOLS, an OPS must have one AOLS block per wavelength, and each AOLS block must have per each input label an AOLXG correlator and a ODL. In addition, each outgoing label requires and additional ODL an input/output port in the PXC of the AOLS block and an AOFF. Therefore, AOLS capital expenditures are linearly dependable on the number of fibers (F), wavelengths (W) and labels (L) that need to be supported, i.e. $O(F \cdot W \cdot L)$.

[B] Objectives

Due to the expensive implementation of AOLS, the number of labels used in OTNs is foreseen to become an important issue soon.

The main objective of this thesis is to analyze different methods of reducing the number of labels used when an AOLS architecture is used. Clearly, under

[1] However, it must be remarked that the payload is not converted.

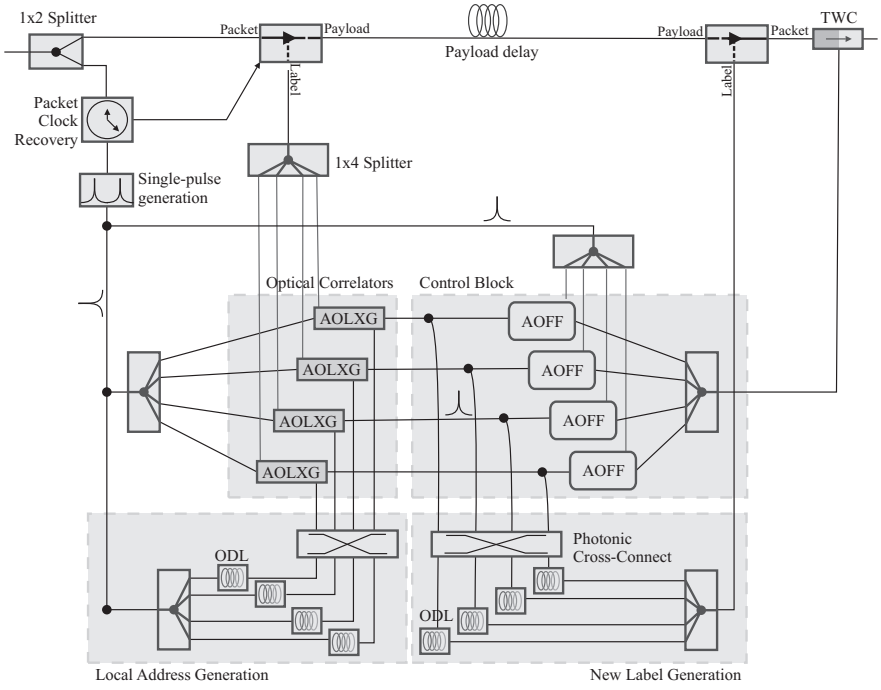


Figure 4: AOLS-block from the LASAGNE architecture

the LASAGNE implementation, as the number of labels is reduced, the deployment cost of AOLS are reduced as well. However, not only the focus is made on reducing labels, but in analyzing the different disadvantage of each of the proposed methods.

[C] Contributions

Although several studies have been made in the physical layer in order to make AOLS cheaper through better optical label codifications schemes and better optical devices, the contributions presented in this dissertation are focused on methods that purely reduce the number of labels used regardless how they are coded.

The contributions take into account the restrictions that the physical layer imposes in AOLS, OBS and OPS subsystems. But, by no means it proposes a new architecture to perform AOLS, OBS or OPS. Therefore, some details of the physical devices are not given, but referenced to documents with proper explanations instead.

As mentioned just before, the main objective concerns reducing labels in optical networks. However, as an initial part of the work, an easier problem is undertaken: reducing labels in pure GMPLS networks. Later, these solutions are extended in order to solve the problem in AOLS networks.

On Pure GMPLS Networks The Label Space Reduction problem in GMPLS has less severe motivations than in AOLS. However, it is foreseen than

other technologies besides AOLS, such as Ethernet or VPNs, would incur in label space problems when coupled with GMPLS.

The following list summarizes contributions made for GMPLS networks in general.

1. Analysis of existing label space reduction methods for GMPLS label merging and hierarchies
2. Developing of improved methods for label space reduction in GMPLS
   - The Asymmetric Tunnel (AT) concept and two algorithms: Longest Segment First algorithm (LSF) and Most Congested Space First algorithm (MCSF)
   - The Asymmetric Merged Tunnel (AMT) concept
   - Two Integer Linear Programs solving the label space reduction problem in GMPLS the Brute-Force integer lineal model (BF) and the Decompose & Match model (D&M)

On All-Optical Label Swapping Networks The application of the solutions mentioned above to AOLS networks is almost straightforward. Minor modifications have to be performed in the AOLS blocks architecture, which is the first contribution in this apart.

In addition, a new wavelength-routing switch architecture capable of reducing even more labels is initially proposed: the G^+ architecture. The G^+ architecture is employed as an improved method for *bypassing* optical processing.

Once the wavelength-switching architecture has been studied and the pertinent modifications to the AOLS-block has been discussed, the main problem is tackled. The Label Space Reduction problem in AOLS is analyzed considering distinct scenarios, leading to the main conclusions of the dissertation. The scenarios mainly analyze the three different approaches for reducing labels, namely: *label merging*, *label stacking* and *optical bypassing*.

[D] Contents

The rest of the document has been organized as follows.

Part I. The part is devoted to the analysis of the Label Space Reduction problem in GMPLS networks solely. The first chapter of this part explains background of GMPLS and related work on the label space reduction problem. The second chapter presents proposed methods by the author for label space reduction in GMPLS.

Chapter 1. The GMPLS forwarding mechanisms is firstly explained. Secondly, its main signaling protocol is described: Resource ReSerVation Protocol for Traffic Engineering (RSVP-TE). Finally, two ways of reducing labels are analyzed: MultiPoint-to-Point Label Switched Path (MP2P) (based on *label merging*) and Hierarchical Label Switched Paths (H-LSP)(based on *label stacking*).

Chapter 2. The chapter describes the contributions of the author in GMPLS networks. Most of them are already published in Conference Proceedings or Journals.

Considering both *label merging* and *label stacking*, two new methods of reducing labels are explained: the Asymmetric Tunnel (AT) and the Asymmetric Merged Tunnel (AMT). For each of them, algorithms (e.g. Longest Segment First algorithm (LSF) and Most Congested Space First algorithm (MCSF)) and optimization models (e.g. Brute-Force integer lineal model (BF) and Decompose & Match model (D&M)) are described.

Part II Once a broad analysis of the Label Space Reduction problem in GMPLS network is given, it is proceeded to a more complex scenario: AOLS networks, which the part is devoted to.

Chapter 3. As an introductory chapter, the LASAGNE architecture is presented in detail. Similarly as it was done in this chapter, the need of reducing labels in this architecture is highlighted as well.

Two variants of the main AOLS architecture, proposed already in the LASAGNE project are explained. While the first one enable an OPS to perform traditional *label swapping*, the second performs *label stripping*. Both variants are analyzed from the point of view of efficiency of the architecture.

In addition, a slight modification of the AOLS-blocks are proposed. The modification allows the architecture to stack labels, enabling the architecture to perform *label stacking*. With this, all the solutions presented in the previous part can be applied for AOLS as well.

Chapter 4. The number of labels used in the network depends on the length of its routes, *viz.* the number of hops. Reducing the number of hops involves a routing problem. WDM offers the capability of setting up a virtual network by means of lightpaths, as discussed previously. By setting up properly lightpaths in the network, a set of demands can be routed end-to-end with less subwavelength processing (hence, less labels). From the point of view of AOLS, this technique is named *optical bypassing* and this document is the first that discusses it.

In this chapter a new wavelength-switched architecture is proposed: the G^+ architecture. The architecture aims, initially, at reducing the amount of subwavelength switching need to route a traffic demand matrix. The reduction of labels is clearly foreseen, hence.

Chapter 5. The last chapter summarizes the most important results of this dissertation together with its limitations. Non-developed ideas in the dissertation are commented as well.

Part I

Label Space Reduction in GMPLS Networks

Chapter 1

MPLS Fundamentals

This chapter introduces the basic concepts of MultiProtocol Label Swithing (MPLS) forwarding. It summarizes the most important aspects of RSVP-TE, its main label distribution and resource reservation protocol. After this, two methods to reduce label spaces are commented: *MP2Ps* and *H-LSP*.

The most relevant contributions regarding MP2P are described. This includes: the algorithm proposed by Bhatnagar *et al.* [BGN05], the Zero-One programming model proposed by Saito *et al.* [SMY00] and the Applegate's upper bound [AT03]. The pros and cons of these contributions are discussed briefly. Moreover, some cons of using any MP2P heuristic for label space reductions are mentioned as well.

Later, the H-LSP method is explained. Its main drawbacks are analyzed.

1.1 Introduction to MPLS Forwarding

MPLS is a forwarding scheme that enables management mechanisms for the core network of a network provider, usually in an Internet environment. MPLS groups user's flows into aggregates and allows a certain capacity to be allocated to each aggregate. Its main characteristic is the separation of IP routers functions into two parts [Swa99]: the forwarding plane, responsible for how data packets are relayed between IP routers using a label swapping mechanism; and the control plane, consisting of layer routing protocols to distribute routing information between routers, and label binding procedures for converting this routing information into the forwarding tables needed for label switching. This separation of the fast forwarding and the routing mechanisms enables each component to be developed and modified independently.

Routers belonging to an MPLS domain are called Label Switched Routers (LSR). A connection established between two LSRs is called a Label Switched Path (LSP). The packets inside an MPLS domain go from one Label Edge Router (LER) to another one, i.e. an ingress LSR to an egress LSR using an LSP.

Traffic Engineering (TE) in MPLS networks is made possible mainly because a future of MPLS named *explicit routing*. With explicit routing, the ingress LSR of an LSP decides the whole route for the LSP. This is, the route is fixed at setup time and preserved over time, unless the ingress LSR explicitly changes

it or the network is unable to use it. In order to ease LSP routes computing, MPLS has been integrated to work with existing Internet routing protocols such as Open Shortest Path First (OSPF) [Moy98], Border Gateway Protocol (BGP) [RL95], or Intermediate System to Intermediate System (ISIS) [Ora90].

When a data packet comes into an MPLS domain, through an ingress LSR, the packet is classified into a specific Forwarding Equivalence Class (FEC) [RVC01]. A FEC groups packets with certain common properties (protocol, size, origin, destination, etc.). These packets are equally routed according to a combination of this information carried in the IP header of the packets and the local routing information maintained by the LSR. An MPLS header is then inserted for each packet in order to differentiate it from others at every hop.

The MPLS header contains a 20-bit Label, a 3-bit experimental (Exp) field, formally called Class of Service (CoS), a 1-bit Label Stack Indicator (LSI) and a 8-bit Time-to-Live (TTL) field [Ros01]. Fig. 1.1 shows the MPLS header. An LSR examines the Label and possibly the Exp field for forwarding the packet. Each LSR use the label as the index to look up the forwarding table. The incoming label is replaced by the outgoing Label and the packet is switched to the next LSR. Before a packet leaves the MPLS domain in a LER, its MPLS header is removed.

An interesting property of MPLS is that the LSI bit allows stacking of MPLS Labels [Ros01]. A Label stack is an ordered set of labels appended to a packet. This enables MPLS tunneling and hierarchies. The LSI bit is set to one for the last entry in the label stack (i.e. for the bottom of the stack) and zero for all other label stack entries. Note that only the top label of the label stack is processed in the LSR.

1.1.1 MPLS Forwarding Mechanism

As mentioned before, MPLS allows labels stacking. The standardization of label stacking is defined by this set of operations (op) in [RVC01]:

- SWAP: replace the label at the top by a new one,
- PUSH: replace the label at the top by a new one and then push one or more onto the stack, and
- POP: remove the label at top in the label stack

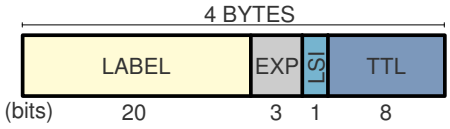


Figure 1.1: MPLS Header

On Notation - The string *[X/A/B]* over a link means that packets crossing that link have the set of labels X, A and B where X is on the top of the stack. On the other hand, the string *X: op Y** over a LSR means that the LSR performs an operation *op* (one of the previously explained) for packets marked with label X; the Y^* (could be zero, one or more than one label) is a parameter for the operation and its meaning depends on the stack operation itself.

An MPLS forwarding (or lookup) table is composed of a set of Next-Hop Label Forwarding Entrys (NHLFE) [RVC01]. Each one of them associates an incoming label with one of these operations (which are done in packet stacks with incoming label) and an outgoing forwarding port. In this way, LSRs can decide where to forward packets marked with a specific incoming label.

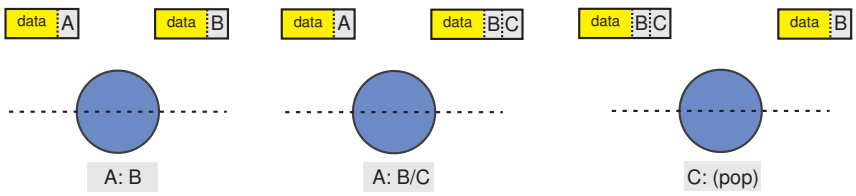
In the packet forwarding process, the Internet Engineering Task Force (IETF) standard imposed to regard only the label at top of the stack because of performance; i.e. the forwarding decision is only based on the *top* label. In this sense, the forwarding process behaves as follows:

1. the LSR extracts the label of the packet header first,
2. the LSR searches for a NHLFE referring to this label
3. with the information provided by the NHLFE, the LSR performs the operation in the stack that the NHLFE states, and
4. the LSR places the packet in the correct outgoing interface to reach the next hop in the LSP.

Note that *if* the LSR takes the forwarding decision based on the first two labels in the stack, the LSR has to have two NHLFEs since each NHLFE is allowed to read and pop only the first label: the label at top. The first NHLFE must refer to the top label and command a pop operation in the stack with outgoing port itself, therefore the second NHLFE must refer to the second label.

Penultimate Hop Popping

MPLS allows forwarding of packets without labels at the last hop of an LSP only. As a consequence, the egress LSR of the LSP (upon packet reception) would



(a) Swap (b) Push (c) Pop

Figure 1.2: MPLS Stack operations

perform routing based on the upper layer (most of the times IP). Therefore, the usage of Penultimate Hop Pop (PHP) reduces the label space.

However, All-Optical Switching (commented in detail in the next part) is purely based on labels, since All-Optical IP routing would be extremely costly. Therefore, we assume along all this dissertation that packets should carry *at least one* label in their headers at any hop.

1.1.2 Outlines on the Resource Reservation Protocol for Traffic Engineering

RSVP-TE [ABG+01] and Constraint-based Label Distribution Protocol (CR-LDP) [JACD02] are the two standardized protocols for label distribution in MPLS networks. Since the IETF had stopped progressing CR-LDP [AS03], in this subsection CR-LDP is not discussed.

RSVP-TE is an extension of the Resource ReSerVation Protocol (RSVP) [1] [BZB+97], proposed previously for IP networks. The initial purpose of RSVP was to provide a signaling protocol that is capable of reserving resources in order to provide Quality of Service (QoS) for IP flows. Due to the dynamism of IP routing, RSVP was thought as a soft-state protocol, i.e. each router stores RSVP information in a block of information named "state" and these states have to be refreshed every so often (otherwise, routers would consider them as old, having the right to discard them).

Reservation Styles

The way in which a flow is identified differs in RSVP and RSVP-TEDetails on how flows are identified are not given in this dissertation, but the reader can refer to its original docuemnts. Regardless of how RSVP or RSVP-TE identifies a flow, three reservation styles are provided. Each flow is signaled using only one of them.

- *Fixed-Filter Style.* Each flow has a unique reservation and the reservation is not shared with anybody else. Per each flow, there is a soft-state that keeps track of the reservation.

- *Wildcard-Filter Style.* The reservation is shared with all the flows with the same destination. This implies that, for flows having the same destination, there is only one state per destination. This reservation may be thought of as a shared "pipe", whose "size" is the largest of the resource requests from all receivers, independent of the number of senders using it.

- *Shared Explicit Style.* Similarly to the Wildcard-Filter reservation style, the Shared Explicit style creates one reservation for a set of flows with the same destination. However, the list of flows that can be *merged* is explicit (specified).

[1] R.S.V.P. is also a french acronym for 'Répondez S'il Vous Plaît' (please reply), which accords with the manner the protocol works before making a reservation.

Support of MP2P Connections in RSVP-TE*

The Wildcard-Filter and Shared Explicit reservation styles are appropriate for those applications that are unlikely to transmit data simultaneously to the same destination, since the reservation is shared among many sources. Examples of these applications include voice-conference applications. When these styles are used, it is said that a single MP2P connection has been established between many sources to the same destination. As a consequence, one soft-state is kept for the MP2P connection independently of how many sources (and flows) it comprises.

Extending RSVP in IP networks to RSVP-TE in MPLS networks was a easy task considering the Fixed-Filter reservation style. However, even though MPLS is able to 'merge' labels in order to create a sort of MP2P, the other two reservation styles are not completely supported so far in RSVP-TE (see §2.4.2 and §2.4.3 of [ABG+01]). The reason is simple. In RSVP every state must store the following information (at least): a list of sources (only one source if the reservation is not shared), the destination, information about the flow characteristic and the next hop of the flow. For a set of flows sharing one reservation (either by using the Wildcard-Filter or the Shared Explicit styles), there is only one state. Therefore, since a state keeps only one next hop entry, all flows sharing a reservation must be forwarded to the same next hop. This would contradict the philosophy of the explicit routing option in MPLS.

Although MP2Pis not supported yet by RSVP-TE henceforth, it is assumed that the IETF will find a solution to overcome this issue.

1.1.3 Label Space Reduction in MPLS

Reducing labels in pure MPLS networks has different motivations, not as strong as for AOLS.

- Once an LSP is established, all the LSRs involved should use a label in order to identify the LSP. In other words, every LSP packet must be marked with a label that identifies the LSP in the LSR (labels are local to the LSR). When an LSR receives a packet, the LSR looks for the packet label and then searches for a NHLFE in its memory that refers to this incoming label. A NHLFE provides information about which interface will be used to reach the next hop in the network [RVC01]. Clearly, the more LSPs a LSR supports, the more NHLFEs are needed.

- Furthermore, reservations are kept in states. As a consequence, the more LSPs a router supports, the more memory it needs to keep track of all the states.

- In addition, since RSVP-TE is a soft-state protocol, refresh messages has to be send every so often. In practice, every LSR in the network has to send (and receive as well) two refreshing messages every 30 seconds for every LSP. This has been foreseen as a scalability problem of both MPLS and RSVP-TE that the IETF is looking forward to solving.

An explosive increase on the label spaces is foreseen by mainly two factors: 1) the aims of MPLS deployment until the edge of the network, and 2) the use of multi-path traffic engineering methods [SFDM04].

As mentioned before, MPLS allows to handle a stack of labels in packets header. In order to manage this stack, the NHLFE contains a field indicating the operation that can be done in this stack. Taking advantage of the different possible operations a NHLFE may have, the number of labels used (or label space[2]) could be reduced more or less depending on how NHLFEs are configured, as explained in further chapters.

When one label stores forwarding information that can be used for more than one LSP-flow, it can be said that there is a label space reduction. Therefore, the general *LAbel SPAce REDuction (LASPARED)* problem can be formulated as:

LaSpaRed Formulation - how can the NHLFEs be set up for a set of LSRs in a network so that the total number of labels used in the network is minimized?

Note that as the number of incoming labels is reduced, the number of outgoing labels, NHLFEs and RSVP-TE soft-states are reduced as well.

Upper Bound for the Generic LaSpaRed problem - An upper bound to the generic problem is the sum of the hop count of all the LSPs, which implies a reduction of zero labels in the space.

Similarly,

Lower Bound for the Generic LaSpaRed problem - A lower bound for the generic problem is the number of links used by all the given paths, which implies using only one label per link regardless of how many LSPs are forwarded through it.

Under high network load conditions, the aforementioned lower bound is likely to become close to the number of links in the network, since all links would be used for flows routing.

The methods presented in this dissertation depend on the data plane capabilities. For instance, some ATM nodes are uncapable of merging flows, therefore uncapable of creating MP2P connections. However, in this part, it is assumed that the data plane technology is capable of performing such operations over flows. Furthermore, it is assumed that RSVP-TE will address the signaling of these methods in a close future.

1.2 MultiPoint-to-Point Trees

Although unsupported by its signaling protocols, MPLS is capable of *labels merging* as a way of reducing label spaces. This is performed by assigning to many LSPs the same outgoing label, if they share the path to an egress LSR.

This reduction scheme creates a set of connections where each one looks like an inverse tree rooted at the egress LSR with leaves at the ingress LSRs of the set of LSPs. For this reason, MPLS architecture calls this structure a

[2]We will use these two terms indistinctively: "number of used labels" and "label space" in this part.

MultiPoint-to-Point tree. In this sense, a remarkable property of each MP2P is that every LSR in the MP2P stores a single label for all forwarded LSPs, which also means that the number of used links of a MP2P is equal to the number of used labels indistinctly of the number of merged LSPs.

Let $\mathcal{N} = \{\alpha_0, \alpha_1, \ldots, \alpha_N\}$ be the set of all LSRs in an MPLS network. Let $\mathcal{P} = \{p_0, p_1, \ldots, p_P\}$ be a set of LSP indexes, where p_i is an index for an LSP route (the reader can relate one LSP index with one label). Let $f_{\alpha_j} : \mathcal{P} \rightarrow \mathcal{N}$ be the routing function for node α_j: given an LSP p_i, the function evaluates the next hop of the route.

The idea behind label space reduction is reducing the domain (and range as well) of every function f_{α_j}. In the case of *label merging*, the domain of the function is replaced by MP2P identifiers. Let $\mathcal{T} = \{t_0, t_1, \ldots, t_T\}$ be a set of MP2P indexes (the reader can relate one MP2P index with one label as well) and $f'_{\alpha_j} : \mathcal{T} \rightarrow \mathcal{N}$ be the routing function for node α_j.

Therefore, the problem is translated to find the best function that maps one element in $\mathcal{P}$ to $\mathcal{T}$ such that forwarding is not altered. In other words, find the smallest set $\mathcal{T}$ and function $g : \mathcal{P} \rightarrow \mathcal{T}$, such that $f'_{\alpha_j}(g(p_i)) = f_{\alpha_j}(p_i)$ for all $p_i \in \mathcal{P}$ and all $\alpha_j \in \mathcal{N}$.

Some interesting properties of this conceptualization are listed below:

- Since the binary relationship g^{-1} creates a partition over the set $\mathcal{P}$, an LSP (identifier) can belong only to one MP2P connection.
- Because $\mathcal{P}$ is partitioned (previous item) and f' is a function, the set of links conforming the LSPs in $g^{-1}(t_k)$ (for any MP2P connection $t_k \in \mathcal{T}$) form an *inverted tree*.

It is clear that for a single egress LSR $e \in \mathcal{N}$ there may be one or more MP2P trees; let $T_e \subseteq \mathcal{T}$ be the set of indexes for the found MP2P trees for an egress LSR e. To simplify notation, let N_{p_i} and N_{t_k} be the set of nodes used by path p_i and tree t_k respectively.

With these MP2P tree structures the label assignation is performed as follows. If β and γ are two consecutive LSRs in a tree t_k - i.e. $f'_\beta(t_k) = \gamma$ - and $f'^{-1}_\beta(t_k) = \{\alpha_0, \alpha_1, \ldots, \alpha_k\}, k \geq 1$, is the set of LSRs whose next hop is β, then LSR γ may query to β a *unique* label L to forward packets for the LSPs going through $\alpha_i, 0 \leq i \leq k$. In this way, LSR β will map many incoming packets marked with different labels, coming from α_i, to the same outgoing label L and, hence, LSR γ will receive all of them using the same label. Therefore, the number of labels used - and NHLFEs as well - in γ is reduced from k to one for the LSP routes in $g^{-1}(t_k)$.

1.2.1 The Merging Problem

As mentioned before, the label space reduction problem considering *label merging* or (MP2P connections) boils down to find the proper mapping function g ($\mathcal{T}$ is induced from it). In other words, the main setting up' problem is to group LSP routes such that in each group: a) there exists *at most* one common path between any pair of LSP routes and, b) the common path - if exists - *ends* in the egress LSR.

The configuration in Fig. 1.3 is considered as an example for showing the Merging Problem. Without merging, 34 labels are needed.

For this configuration, there can be created at least two MP2P for the five LSPs shown since link $N11 \rightarrow N10$ is crossed by many diverging LSPs: B, C, D and E - i.e. this forbids us to create a single tree with those five LSPs at once.

Depending on how LSPs are grouped for forming MP2P connections, the label space may be reduced more or less. For the example in figure Fig. 1.3, by merging LSPs A with both LSPs D and E - leaving LSP B and C on a new tree (Fig. 1.4(a)) - 18 labels are needed (16 labels less), whether by merging LSP A with both LSPs B and C - leaving LSP D and E on a new tree (Fig. 1.4(a)) - 17 labels that are needed (17 labels less).

Under these assumptions, the MERGING PROBLEM can be stated as follows.

> MERGING PROBLEM - GIVEN. The set of paths in the network, name it $\mathcal{P}$, and a bound on the number of labels, name it integer B,
> QUESTION. Is there a set of trees $\mathcal{T}$ such that
>
> - $\forall p \in \mathcal{P}$, $\exists t \in \mathcal{T}$ such that p is a branch of t and,
> - the number of all links in $\mathcal{T}$ does not exceed B?

1.2.2 Bhatnagar's Merging Algorithm

Bhatnagar, Ganguly and Nath in [BGN02], [BGN03] and [BGN05] considered the LAbel SPAce REDuction (LASPARED) problem and they proposed an algorithm to solve it by creating MP2P trees. In their work, they considered as a goal to minimize the *number of MP2P trees* created for a given set of pre-computed LSPs routes. By minimizing the number of created trees (counting non-merged LSPs as one tree), the *maximum* number of labels used a LSR may have in a network is bounded. The work of Bhatnagar *et al.* found in [BGN05] is transcribed in this subsection. The notation used is the same as the original document.

Merging Index The merging index, m_{ij} of a pair of paths $p_i, p_j \in P_e$ is defined as the number of continuous nodes starting from egress e that form part of both p_i and p_j. The merging index is set to zero if p_i and p_j meet at any point other than those forming the common chain from e.

The algorithm 1, in page 11, shows this process. Initially, all the paths form the set of trees. Using the merging indices of these trees, the Bhatnagar's algorithm chooses a pair and merges them into a *denser* tree, reducing the size of the tree set, see algorithm 2 on page 11. In selecting the next pair of trees to merge, the algorithm chooses the pair i and j with the *maximum* value of m_{ij}, line 3. After choosing a pair to merge, the algorithm updates the merging indices of all the remaining trees to reflect their new merging index with the newly formed tree, line 4.

Once the pair of trees to be merged is decided, the indices of the remaining trees are adjusted, see algorithm 3 in page 12. If a tree has a merging index of zero with either of the constituents, its merging index with the new tree is set to zero because it will form a cycle with the new tree, line 2. Otherwise, the tree can still merge with the new tree at the point where it could have merged with the constituent trees, line 4.

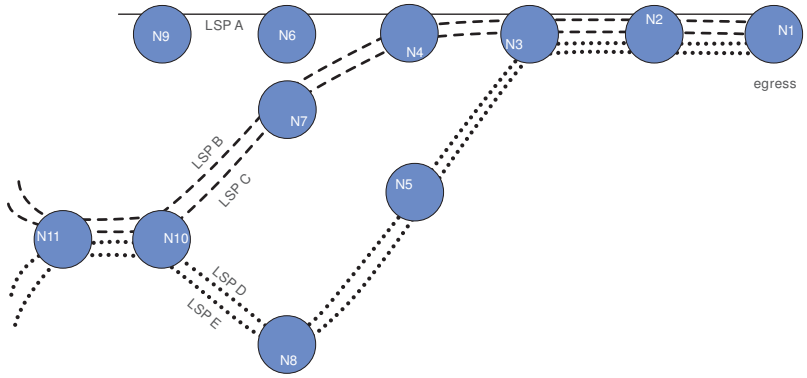
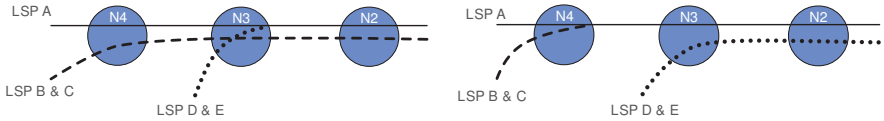


Figure 1.3: MP2P Scenario with five LSPs with the same egress LSR $N1$. There may exist more than one way to perform merging since $N11 \rightarrow N10$ is crossed by more than two diverging LSPs.



(a) Merging LSPs A and C (b) Merging LSPs A and B

Figure 1.4: MP2P solutions

Procedure `PathMergingIndices`

Input: P_e: the set of paths with egress e

begin
  foreach $p_i, p_j \in P_e$ **do**
    $Chain \leftarrow \{e\}$;
    while $NextNode_i == NextNode_j$ **do**
      $Chain \leftarrow Chain \cup NextNode_i$;
    Remove nodes in $Chain$ from p_i and p_j ;
    if *there are common nodes in* p_i *and* p_j **then**
      $m_{i,j} \leftarrow 0$;
    else
      $m_{i,j} \leftarrow \|Chain\|$;
end

Procedure `MergingAlgorithm`

Input: P_e: the set of paths going to egress e

begin
  Compute All Merging Indices;
  while $\exists m_{i,j} \neq 0$ **do**
    Choose i and j having $\max m_{i,j}$;
    Update Indices of all tress w.r.t. i, j;
    Merge i and j;
end

Procedure `TreeMergingIndices`

```
   Input: m_{k,i}: merging index of tree k and i
   Input: m_{k,j}: merging index of tree k and j
   Input: m_{i,j}: merging index of tree i and j
1  begin
2      if m_{k,i} == 0 or m_{k,j} == 0 then
3          m_{k,(ij)} ← 0 ;
4      else
5          m_{k,(ij)} ← m_{k,i} ;
6  end
```

1.2.3 Saito's Zero-One Programming Model

On the other hand, Saito, Miyao and Yoshida in [SMY00] proposed a Zero-One Integer Programming model aiming to minimize the number of created MP2P trees, similarly to the previously described algorithm of Bhatnagar *et al.*. Once more, the notation is kept the same as the original document.

Sets

$\{e\}$ Egress node.

T_e A set of MP2P trees of egress node e.

N A set of nodes.

$N_e^{Ingress}$ A set of ingress nodes for egress node e.

L A set of links, and each element is (l, m). The link goes from l to m.

L_e A subset of link set L, in which links from egress node and links to ingress nodes are removed. $L_e = L - \{(e, m) : (e, m) \in L\} - \{(l, m) : (l, m) \in L, \forall m \in N_e^{Ingress}\}$

$P_{(i,e)}$ The set of selected routes between ingress node i and egress node e.

$L^{p(i,e)}$ A set of links used in route $p_{(i,e)}$.

Variables

r^{t_e} Set to 1 if MP2P tree t_e includes a part of selected routes, otherwise set to 0.

$h_{(l,e)}^{t_e}$ Set to 1 if MP2P tree t_e uses link (l, m), otherwise set to 0.

$\delta_{p(i,e)}^{t_e}$ Set to 1 if MP2P tree t_e includes route $p_{(i,e)}$, otherwise set to 0.

A MP2P is represented by a set of variables $h_{(l,m)}^{t_e}$, which takes a value 1 if MP2P tree t_e uses link (l, m), otherwise takes a value 0. Let $\delta_{p(i,e)}^{t_e}$ be 1 if path $p_{(i,e)}$ is considered in MP2P tree t_e.

The Zero-One programming model can be, hence, formulated as:

$$\min \sum_{t_e \in T_e} r^{t_e} \tag{1.1}$$

subject to:

$$\sum_{m:(l,m)\in L_e} h^{t_e}_{(l,m)} \leq 1, \quad \forall l \in N \backslash \{e\},\ \forall t_e \in T_e \tag{1.2}$$

$$\sum_{(l,m)\in\{L^{p_{(i,e)}} \cap L_e\}} h^{t_e}_{(l,m)} \geq \|L^{p_{(i,e)}}\| \cdot \delta^{t_e}_{p_{(i,e)}}, \quad \forall p_{(i,e)} \in P_{(i,e)}, \forall i \in N_e^{Ingress}, \forall t_e \in T_e \tag{1.3}$$

$$\sum_{t_e \in T_e} \delta^{t_e}_{p_{(i,e)}} = 1, \quad \forall p_{(i,e)} \in P_{(i,e)},\ \forall i \in N_e^{Ingress} \tag{1.4}$$

$$\sum_{i \in N_e^{Ingress}} \sum_{p_{(i,e)} \in P_{(i,e)}} \delta^{t_e}_{p_{(i,e)}} \leq \sum_{i \in N_e^{Ingress}} \|P_{(i,e)}\| \cdot r^{t_e}, \forall t_e \in T_e \tag{1.5}$$

$$h^{t_e}_{(l,m)},\ \delta^{t_e}_{p_{(i,e)}},\ r^{t_e} = 0/1 \tag{1.6}$$

The objective function, (1.1), indicates minimizing the number of MP2P trees. The constraint in (1.2) establishes that MP2P trees must have only one outgoing link in each forwarding node. (1.3) guaranties that each MP2P tree branch follows the same path as their underlying LSPs. (1.5) gives a definition of r^{t_e}, as mentioned before. (1.6) defines the domain of the variables as binary.

1.2.4 Applegate's Bound

Applegate and Thorup proposed in [AT03] an algorithm that builds $N + M$ MP2P trees, where N and M refer to the number of LSRs and links respectively in a given network. They asserted that each LSR uses at most $N + M$ labels since they bound the number of build MP2P trees to $N + M$. However, this is not clear considering that the merging LSRs of a MP2P tree must count two different NHLFEs for two different incoming interfaces of a LSR even if the incoming labels are the same (the reader may go to §3.14 of [RVC01]).

Their bound is based on a procedure that *reroutes* the traffic (previously routed in the network) while allocating the same bandwidth (see 4).

It must be pointed out that other QoS parameters were left out in the rerouting heuristic. Therefore, it is possible to lack of previously ensured guarantees for delay, jitter, etc. Therefore, although labels are dramatically reduced, QoS is enormously degraded.

Since rerouting is considered by the authors, the computed bound will not be considered for MPLS LASPARED.

1.2.5 MP2P Drawbacks*

All previously presented related work have in common the same objective: to minimize the number of created MP2P trees. The example in Fig. 1.3 shows

Function `TreeFlow(`v`,`c`,`R`)`

Input: v: a node, like u
Input: c: maximal flow toward t
Input: R: a MP2P index
Data: $F^t_{(u,v)}$: flow on link (i, v) going to t
Data: $D_{(v,t)}$: 0 if v is not a source, otherwise the demand of source v to destination t
Data: D^R_v: demand of node v for tree R

```
1  begin
2      f ← D_(v,t) ;
3      if f ≥ c then
4          f ← c ;
5      D^R_v ← f ;
6      forall parent u of v in R do
7          g ← TreeFlow(u, min{c − f, F^t_(u,v)}, R) ;
8          F^R_(u,v) ← g ;
9          f ← f + g ;
10     return f
11 end
```

that such goal may be *weak* for some configurations since there may exist two solutions with the same number of MP2P trees for a given set of LSP routes, but differing in the number of used labels. Therefore, it is possible that while minimizing the number of created trees, the solutions of the problem can be suboptimal in matters of labels.

So far, although Bhatnagar's and Saito's work outlined the importance of reducing the label space *directly* (not the minimum number of MP2P trees, but the number of used labels instead) as future works, no solutions were found for the MERGING PROBLEM considering this objective by using MP2P trees.

In addition, we proceed to show two major drawbacks of the MP2P method in general, discussed by the author in [SSFM07]. The first drawback is a consequence of the MPLS forwarding mechanism: *the label space of an LSR cannot be reduced using LSPs with different egress nodes.* The second drawback is related to the treelike shape of MP2P connections:

Fact 1 (MP2P Biased Reduction) *The greatest percentage of reduced labels by MP2P in a network are located in LSRs near an egress LSRs.*

In order to corroborate this statement, we carried out the following experiment.

A reduced version of the Australian ISP topology discovered by the Rocketfuel engine, shown in Fig. 1.5, was used. The simplified version has 28 nodes, each one corresponding to a different location in Australia[3]. The nine nodes having the lowest connectivity degree were selected as edge routers[4]. A set of

[3] A set of nodes belonging to one location were mixed as a single node.

[4] We also performed experiments with randomly selected edge routers. The conclusion follows equally if the average hop distance to *all* the egress is considered instead. In the interest of space and non-redundancy, we do not include them in the article.

500 different LSPs were routed between the edge routers. The minimum number of labels using MP2P was computed. Throughout this work, we classify the relative label space reduction of a node in one of four ranges: [0%,25%); [25%,50%); [50%,75%) and [75%,100%). In the figure, the color of the nodes indicate its range of label space reduction using MP2P. At the same time, we ranked the nodes according to the minimum number of hops to the closest egress: zero for egress, one for egress' neighbors, and so on. As seen in the figure, the shape of the LSRs corresponds to their rank. Table 1.1 shows the average label space reduction percentage for each node rank in the example.

Finally, we computed the correlation coefficient between the nodes's rank and the percentage of the total number of labels reduced. The correlation is large negative (close to -0.8), confirming our previous statement. This means that the LSRs at the core of the network will not reduce their label space as much.

In the next chapter we will propose a way of easing this asymmetry by stacking one label, or tunneling. Moreover, it will be shown later that our solution reduces the number of labels used even more when compared with MP2P.

1.2.6 Is MP2P Computation Actually NP-Complete?*

As regarded previously, previous work of many authors affirm that the problem of minimizing the label space using MP2P - the MERGING PROBLEM - cannot be solved optimally with a polynomial algorithm (NP-Complete), since it involves a decision problem: deciding which LSPs are merged together (equally, deciding g recalling nomenclature used at the beginning of this section).

However, in this subsection, the tree consideration is analyzed and it is concluded that it is needless making the problem much easier to solve. In fact, label merging can be optimally computed in polinomial time. The algorithm for finding such optimal solution is proposed here: *Full Label Merging* algorithm. Therefore, the MERGING PROBLEM is reclassified as P, instead of NP-Complete. This contribution can be regarded in [SFM07].

Where is the so-called MERGING PROBLEM ?

Some authors contributions - e.g. [AT03], [SMY00], [NP04], [BGN05] - considered that a MP2P connection looks like an *inverse tree*, allowing them to map the well-known problem of MINIMUM CLIQUE PARTITION [GJ02] to the MERGING PROBLEM discussed here. This restriction is the main point of discussion of this subsection.

The most remarkable problem of this tree consideration is that every pair

Table 1.1: Average of the Label Space Reduction Percentage for every Rank.

Rank	Num. of Nodes	Avg. LaSpaRed
0	9	88.84%
1	8	44.39%
2	5	33.29%
3	5	26.70%
4	*1*	38.88%

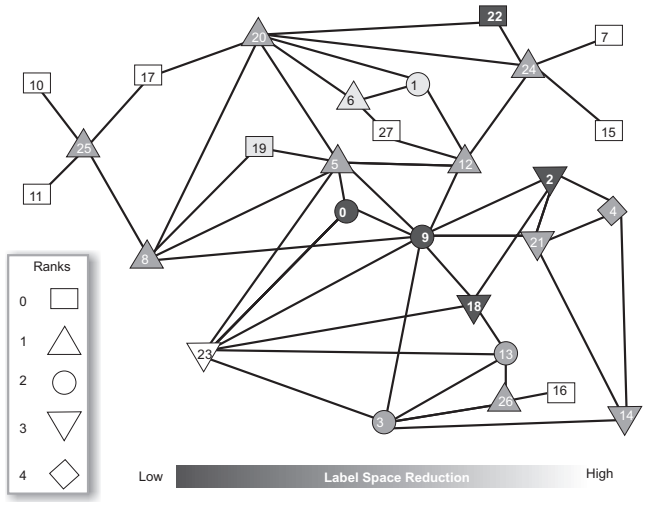


Figure 1.5: Simplified version of the Australian Rocketfuel ISP topology. Nodes ranked according to closest egress proximity and colored according to level of label reduction using MP2P.

of selected LSPs for a tree must not be intersected in more that one segment ended at the egress LSR name it the *ending segment*.

The most important reason to take this consideration is that the created MP2P trees store exactly one outgoing label at each LSR for all forwarded LSPs. Although the consideration makes LSPs management easier, it makes the problem NP-Complete since taking an LSP route would avoid considering other LSP routes for the same tree that interfere with the already taken LSP. The appropriate selection of a set of routes as branches for a MP2P tree will make the label spaces increase depending on how many LSRs are shared among these routes. This Merging Problem was initially analyzed by Saito *et al.* in [SMY00] and formally claimed NP-Complete by Bhatnagar *et al.* in §III of [BGN03].

It should be pointed out that the longer the branches on a MP2P tree are, the less the number of non-interfering LSPs as branches of such tree would be; and which is more important:

Fact 2 *The number of feasible reduced labels is proportional to the number of merged LSPs in a tree.*

Labels Merging without Tree Structures

The main point of discussion of this chapter is *why selecting routes not interfering but in the ending segment?* For further reference, this asking is denominated as the *non-interfering question*.

Suppose the tree consideration is overridden in what label merging regards, i.e. labels may be merged for LSPs that are intersected in many places. This makes the set of merged LSPs to look dislike inverse trees, although all of them end in the same egress LSR. Without the tree consideration, labels binding must be performed according to the following rule. If two LSPs are intersected

elsewhere but in their ending segment, they *must* use a different label; otherwise, they use the same label inside the intersected segment. This could make LSRs to use more than one outgoing label, since merged LSPs could be intersected in not ending segments of the MP2P connection. This type of label merging - without the tree consideration - are named *full labels merging* in order to distinguish it from the traditional MP2P trees.

From this point of view, the tree consideration may decrease the possibilities of reducing the label space respect to this proposal, full labels merging. For example, consider the LSP configuration shown in Fig. 1.3. The number of used labels are 15 labels considering that a) LSR N10 and N11 use two labels - one for LSPs B and C, and another for D and E - and, b) each link in the path $N4 \rightarrow N2$ uses one label for all LSPs. With the tree consideration the number of reduced labels could be 16 or 17 at most, but never 19 as it was just claimed. Refer to Fig. 1.6 for visualization.

Fact 3 *With full labels merging, all the LSPs going to an egress LSR can be placed in only one MP2P connection, hence achieving a better reduction taking care of labels binding.*

For this reason, the problem of deciding which LSP routes must be picked up to create an MP2P connection is overcome since all of them can be picked up at a time for a single MP2P connection. Therefore, up to this point, the MERGING PROBLEM could not be NP-Complete because there is no decision problem to deal with.

To our best knowledge, an accurate motivation to hold the non-interfering question was not found .

A Polynomial-Time Optimal Solution

Based on the ideas presented in previous section, the proposed labels binding rules for full labels merging are formalized with an easy online polynomial algorithm using only label swapping operations on the NHLFEs.

The following algorithm assigns the label for a new LSP L. The algorithm is recursive in terms of the network LSRs. In each recursive call, an LSR N is queried for a label. The first requested LSR is the ingress LSR of the LSP.

As it can be seen in the algorithm, labels binding takes place regardless of the LSP route itself, but mainly on the incoming and outgoing interfaces on the queried LSR N that the new LSP L traverse. Note that since two LSPs share an ending segment, they share the same outgoing and incoming labels as well. However, if they diverge, their labels become different. If they converge together again later, their labels will be still different because they will have different previously assigned outgoing labels. Therefore, no incorrect label swapping or forwarding may take place here.

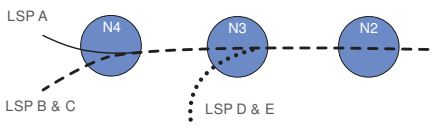


Figure 1.6: Full Label Merging Solution

Function `FullLabelMerging(`N,L`)`

Input: LSR N
Input: LSP L
Output: Requested Label

```
1  begin
2      i ← incomingInterface(N, L) ;
3      o ← outgoingInterface(N, L) ;
4      if N == egress(L) then
5          e ← NHLFE[i, o, −] ;
6      else
7          x ← FullLabelMerging(N + 1, L) ;
8          e ← NHLFE[i, o, x] ;
9      if e = ∅ then
10         l ← newLabel(N) ;
11         e ← newEntry(i, o, l) ;
12         addEntry(N, e);
13     return incomingLabel(e);
14 end
```

As there is no decision problem regarding labels binding (fact 3), the order in which LSP routes are given to the algorithm is not relevant.

To map this algorithm to a protocol could be simple since: a) it is recursive on the LSRs, and b) the output label of each iteration is computed only with the local information to the LSR. Moreover, the algorithm behaves similarly to RSVP-TE in terms of labels assignation.

Fact 4 *The solutions found by the algorithm are optimal since a) there is no decision problem (fact 3), and b) the reduction on labels merging schemes depends on the selected branches and the full label merging solution takes them all at a time (fact 2); no possible improved solution could be computed. Since the algorithm finds the optimal solution to the problem in polynomial time, the* Merging Problem *is not NP-Complete.*

On the other hand, it is clear that to propose either optimization models to compute the optimal set of MP2P trees in terms of reduced labels, heuristics, or online algorithms for near-optimal solutions might be inappropriate contributions.

Numerical Improvements

Although the main point of discussion of this chapter was to show a simple way (Full Label Merging algorithm) to compute the optimal labels binding using label merging, using simulations the number of reduced labels that the full label merging solution can improve - because of the tree consideration absence - respect to MP2P trees was analyzed too.

In all simulations, the network topologies used are generated using *Power Laws* - described in [SFFF03] - with 20 nodes and rank exponent -0.7. In each simulation, four of the 20 LSRs are selected as LERs, i.e. assuming both ingress and egress functions.

LSPs routes are computed using a multi-objective evolutionary algorithm in which several QoS metrics are considered [DFM04]. It should be remarked that, since the routing solution is multi-objective, a set of possible routes could be computed for a single set of demands. In this case, the average of the reductions is calculated. The number of generated demands (LSP routes) is varied between the pairs of LERs from 12 to 100. Then, the number of used labels using both Bhatnagar *et al.* algorithm and the one presented here is computed; their relative difference is taken as the improvement factor.

Table 1.2 shows that the 60% of the performed tests achieved a better reduction using full label merging, while the remaining 40% keeps the same reduction.

1.3 Hierarchical LSPs

1.3.1 Working Principle

Kompella and Rekhter in [KR05] propose a mechanism to make MPLS scalable through the creation of a hierarchy of LSPs in a network. The hierarchy is created as follows.

1. An LSR creates an LSP with large bandwidth. The LSP will act as a parent LSP in the hierarchy for subsequent LSPs.
2. The head-end LSR advertises a new link (a forwarding adjacency, or FA) connecting itself with the tail-end LSR of the parent LSP.The advertisement is made into the same instance of ISIS / OSPF.
3. When a traditional LSP (or child LSP in the hierarchy) is needed to be setup, other LSRs may use the new Forwarding Adjacency (FA) for the path computation of the child LSP, and
4. The new child LSP is nested (by using the label stack construct) into the parent LSP announced as a FA.

In brief, parent LSPs are created first and then considered as virtual links (FA-links) in the network by other LSRs. According to [KR05], the maximum reservable bandwidth of a FA-link is set to the bandwidth of associated parent LSP (see §3.1.6 and §3.1.7), unless a different value is configured manually. Once all parent LSPs have been setup, child LSPs are routed considering both original physical links and induced FA-links.

The main motivations for creating a hierarchy of LSPs are: a) network management simplification, and b) soft-state reduction. On the other hand, the most important disadvantages are explained below.

Table 1.2: Improvement of full label merging respect to MP2P trees.

Percentage of Tests		Improvement Ratio
40%		not improved
60%	59%	$\geq 1\%$
	55%	$\geq 2\%$
	25%	$\geq 3\%$
	15%	$\geq 4\%$

1.3.2 H-LSPs Drawbacks*

Path Optimality

The placement of child LSPs over parent LSPs could resolve into far from optimal route solutions. For instance, in the case of large traffic demands, it could be highly unlikely to find a relative-short path with enough available bandwidth in the parent LSP mesh network for routing the demand. Sometimes, this could lead to either connection denial or parent LSP mesh network resizing.

Resource Affinity

Affinity refers to the general characteristics from which network resources can be distinguished, or classified [AMA+99]. It is mainly applied for links differentiation into several classes (or colors). In this way, an LSP can be setup so it is either forbidden or forced to consider links with certain color(s). This is helpful at the time of computing a path aiming at satisfying certain QoS, e.g. an LSP could take the highly-reliable "green" links, but not the slow "blue" ones.

The affinity of a FA-link is left opened in [KR05] (see §3.1.8). However, for the correct routing of child LSPs, the affinities of FA-links should correspond with all (the union of) the affinities of the physical links that the parent LSPs traverse. Therefore, it may be required to provision a set of parent LSP meshes for matching several child LSPs affinities. Since the child LSPs affinities are not known *apriori* (when parent LSPs are created), the network provider must supply an exponential number (in terms of the number of the colors) of parent LSP meshes. This goes against the main purpose of H-LSP.

Bandwidth Allocation

Bandwidth allocation in H-LSP points out the following problems.

Child Allocation It is not always straightforward to determine the required bandwidth for an LSP, and several techniques based on measurement and/or prediction can be used to that end. The introduction of an additional level of hierarchy undoubtedly increases the complexity, since the parent LSPs must be sized knowing the paths followed by the child LSP. This imposes to pre-compute all the LSPs path (which is a real limitation for network operators nowadays). In the case of child LSP computed using a distributed Constraint-based Shortest Path First (CSPF) approach, the sizing of the parent LSP is a serious challenge.

Resizing Signaling In the event of having a necessity for increasing the bandwidth of a child LSP, the signaling process could be somehow complicated and slow. In the case that the FA links do not have enough bandwidth to satisfy the new bandwidth requirement, either the child LSP could be rerouted through new FA links or the associated parent LSPs could be resized. The later involves another resizing step, delaying the signaling.

Protection

A number of Service Providers rely on Fast ReRoute (FRR) (local protection scheme defined in [PSA05]) to reroute a set of affected LSP onto a backup

tunnel upon a network element failure. Indeed, it has been one of the main reasons of many providers to switch to MPLS as the control plane technology handling their core networks. Although MPLS protection is claimed to be performed within hundreds of milliseconds, it is well known that protection schemes increases the number of states in the networks and its complexity. Aiming at reducing states using H-LSP introduces the following new inconveniences on FRR *per se*.

The protection of the child LSP against the failure of the head-end of their parent LSP requires the provisioning of a backup tunnel from the point of local repair (node immediately upstream to the parent LSP head-end) to the parent LSP tail-end, see Fig. 1.7. Considering physical links, this situation leads to longer backup paths. Long backup paths not only trend to misuse network resources, but also increase the rerouting time.

1.4 Chapter Remarks

In this chapter the fundamentals of MPLSis discussed. We dug into two previously proposed methods for reducing label spaces: MP2P and H-LSP.

Concerning MP2P its most relevant contributions were discussed. In addition, the MP2P NP-Completeness was discussed. From that point, it was claimed that the MERGING PROBLEM is solvable in polynomial time by means of a simple heuristic: *Full Label Merging* algorithm. The algorithm not only computes the optimal solution in polynomial time, but also achieves better reductions.

H-LSP was analyzed as well. Several drawback for Traffic Engineering were found, namely: Path Optimality, Resource Affinity, Bandwidth Allocation and Protection.

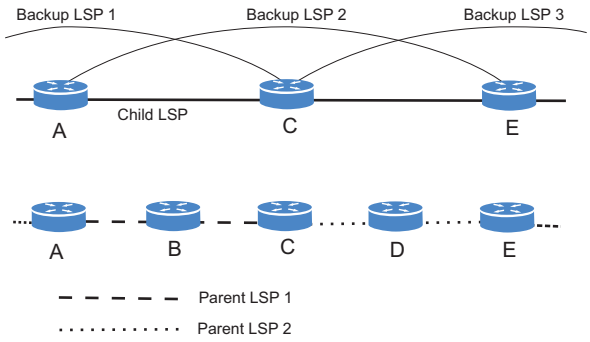


Figure 1.7: Protection and MPLS hierarchies

Chapter 2

Reducing Labels in GMPLS Networks*

In this chapter, different proposed methods are studied for label space reduction.

To overcome most of the drawbacks of H-LSP (mentioned in the previous chapter), the methods presented here aim at constructing *reversed* GMPLS hierarchies. This is, the child LSP routes are initially given (computed by an external routing TE algorithm, hence fixed from this perspective) and over them a hierarchy is computed with the objective of reducing labels.

Path optimality, affinities and bandwidth allocation are not any longer a problem due the fact that child LSPs were computed first taking into account solely the network physical resources. To simplify notation, in this chapter, child LSPs will be named simply LSPs and parent LSPs will be named *tunnels*. The purpose of this chapter is to present contributions concerning how these tunnels can be computed, such that the best reduction of labels in the network is achieved.

In this chapter, two new methods for reducing labels are analyzed: AT and AMT. Algorithms and mathematical models for each of the methods are discussed within each of the sections.

2.1 The Basis: Label Stacking

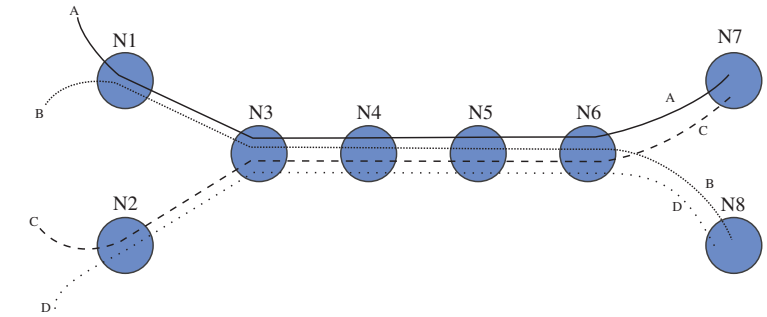
Label stacking methods refer to pushing the same label in a set of LSPs, so the LSRs can regard them as belonging to the same 'LSP'. Fig. 2.1 gives a brief idea of the label stacking basis used as mean to reduce the label space.

Let us introduce the concept of *segment*.

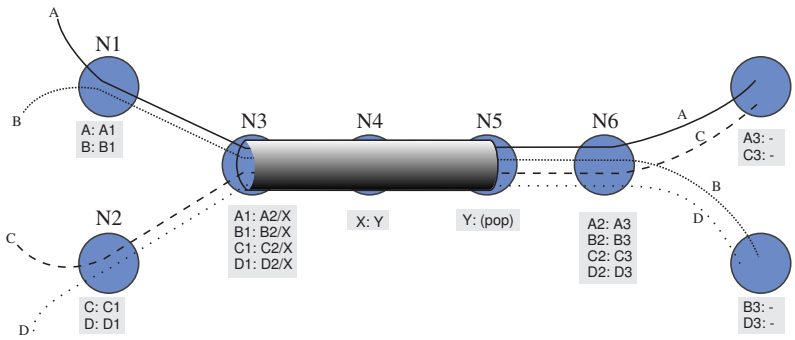
Definition 1 (Segment) *A* segment *is a sequence of two or more consecutive network links - denoting a path - in a network between two (not necessarily adjacent) nodes.*

In this sense, any LSP in the network follows a segment in the network, but not any segment of the network is necessarily used as an LSP.

Consider two LSPs, A and B, whose routes are intersected in segment (or path segments), somewhere in the network:



(a) Example of Topology



(b) Label Stacking

Figure 2.1: Tunneling in MPLS

- LSP A: $\alpha_{0...A} \rightarrow \lambda_{0...N} \rightarrow \alpha'_{0...A'}$, and
- LSP B: $\beta_{0...B} \rightarrow \lambda_{0...N} \rightarrow \beta'_{0...B'}$,

Note that path segment $\{\lambda_0, \lambda_1, \ldots, \lambda_N\}$ is taken by both LSPs. This scheme allows LSRs in the shared path segment ($\{\lambda_0, \lambda_1, \ldots, \lambda_N\}$) to use the same label by pushing at λ_0 the same, but new, label onto both LSPs label stacks; and then popping the stack at λ_N. Therefore, LSRs $\lambda_{1...N}$ will only employ one label for both LSPs. In this case, it is said that LSPs A and B are *stacked* or *tunneled* over $\lambda_{0...N}$. Complementary, the segment $\{\lambda_0, \lambda_1, \ldots, \lambda_N\}$ is considered as a *tunnel*.

2.1.1 Types of LSRs in a Tunnel

First of all, depending on the operation the LSR performs in the packet stacks (registered in the NHLFEs), the LSRs in a tunnel can be differentiated in the following types:

Definition 2 (Pushing LSR) *A pushing LSR is that which has at* least *one 'push' operation in a set of NHLFEs concerning a set of tunneled LSPs (it can also contain swap operations)*

Definition 3 (Swapping LSR) *In a swapping LSR, all NHLFEs are swap operations for the tunneled LSPs*

Definition 4 (Popping LSR) *A popping LSR is that which all its NHLFEs are pop operations for the tunneled LSPs.*

Definition 5 (Receiving LSR) *Since a* popping LSR *can only access to the top of the stack (even if it pops the stack later), all packets are forwarded to the same place. We will refer to the next forwarding hop of a* popping LSR *in a tunnel as the* receiving LSR *of the tunnel.*

2.1.2 Minimal Tunnel Length

Although *receiving LSRs* do not perform any special operations on packet stacks, they will be counted as part of tunnels throughout this work. The reason is simple, all LSPs in a tunnel *must* share the *same* receiving LSR. An example of these four types of LSRs can be regarded in the early Fig. 2.1 in the same order they were mentioned (*pushing LSR*, *swapping LSR*, *popping LSR*, *receiving LSR*).

Fact 5 (Minimal Tunnel Length) *A tunnel must have at least a pushing LSR, a popping LSR, and a receiving LSR.*

2.1.3 No packets replication

In common Point-to-MultiPoint (P2MP) LSP forwarding, each time that a branching LSR needs to forward a packet, the branching LSR replicates the incoming packet and then swaps the incoming label of each replicated packets for the next-hop (downstream) LSRs' incoming label.

Since the label behind the top is never swapped along a tunnel, it should be noticed that the label of a packet before being tunneled is the same label *given* to the receiving LSR (see Fig. 2.1). As a consequence, including branching LSRs in tunnels (i.e. P2MP tunneling) means that *all* the set of receiving LSRs gets packets with the *same* label: the swapped incoming label at the push LSR(preserved along all the tunnel behind the top label).

To illustrate this, consider the P2MP configuration in Fig. 2.2 with two P2MP LSPs. It shows an improper P2MP tunnel which stacks both P2MP LSPs. Without a tunnel, LSR N3 will replace an incoming label with two different outgoing labels in order to assure correct packet forwarding to egress nodes N6 and N7.

When LSR $N3$ receives a packet, it extracts the top label Y and seeks for a NHLFE referring to the extracted label. Based on the NHLFE information, the LSR duplicates the packet (one for being forwarded to $N4$ and the other to $N5$). Then, it swaps the top label on both outgoing packets according with the incoming label at the respective next hop ($Z1$ and $Z2$, respectively). Notice that the LSR is not able to swap the label behind the top, i.e. A or B. Therefore, at the end of the tunnel, the receiving LSRs ($N6$ and $N7$) get packets with the same label behind the top.

Such a forwarding is possible only if all the receiving LSRs of the tunnel agree in their incoming label. For instance, LSRs $N6$ and $N7$ must agree to receive packets of P2MP LSP A with the same label A and moreover, packets of P2MP LSP B with the same label B.

Making LSRs to agree on their incoming label for a particular multicast flow would require a complicated signaling. Therefore, it is foreseen that P2MP tunnels cannot be considered as solutions for the label space reduction problem using stacking.

Fact 6 (No Packet Replication) *P2MP connections cannot be established inside tunnels.*

Supporting Multicast Connections In order to reduce label spaces in P2MP connections, the connections have to be split in several Point-to-Point LSP (P2P) connections at the branch points. In the example, each P2MP connection has to be considered as three different P2P connections: $N1 \rightarrow N2 \rightarrow N3$, $N3 \rightarrow N4 \rightarrow N6$ and $N3 \rightarrow N5 \rightarrow N7$.

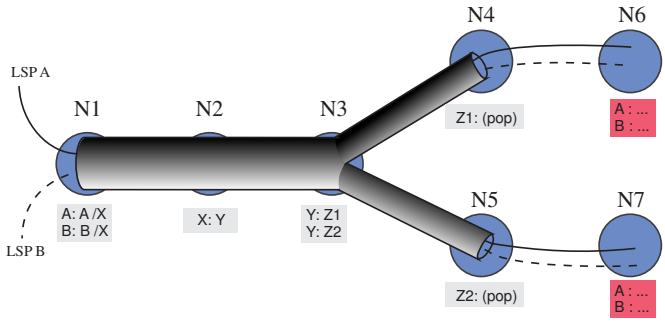


Figure 2.2: Unfeasible Point-to-MultiPoint Tunnel

In order to deliberate on both types of LSPs - P2P and P2MP, the following definition mentioned barely before is given:

In this sense, all the given P2P LSP paths are segments, but not the opposite. Despite this, the term 'segment' is employed for referring to a 'P2P LSP' sometimes.

It must be pointed out that a P2MP LSPs is mapped to a set of at most $2 \cdot d - 1$ segments, where d is in this case the number of egress LSRs of the P2MP connection.

2.1.4 All Popping at Once

Because MPLS LSRs can only consider the top of the packet stack, packets arriving with the same label at the top will be treated equally. Therefore, there is no way that an LSR can *untunnel* (i.e. pop the stack) some (not all) of the LSPs within a tunnel.

Fact 7 (All Popping at Once) *All LSPs in a tunnel must be popped by a single LSR (the popping LSR), but they can be pushed to a tunnel at any LSR (except the last one).*

2.2 Asymmetric Tunneling

In this section we talked about a broader types of tunnels, the *asymmetric tunnels*. A definition is firstly given and then a description of the Tunneling Problem is given, which applies to the previously presented types of tunnels and the asymmetric tunnels as well. The rest of the section is devoted to present heuristics aiming at reducing the label space through the usage of asymmetric tunnels.

The AT concept comes from the idea that *not all* the stacked LSPs are tunneled along all LSRs, this means that LSPs can be pushed at different points in the tunnel.

Formally, if another LSP, say $C : \gamma_{0...C} \rightarrow \lambda_{i...N} \rightarrow \gamma'_{0...C'}$, shares the final sequence of LSRs in a tunnel, $\lambda_{i...N}$, then LSR λ_i can perform a *push* into C label stack in order to forward C packets through the tunnel. In this case, it is said that C is *partially tunneled*.

For example, in Fig. 2.3, N4 stacks LSP D by pushing the same label that is swapped for the tunnel, i.e. X3. The *All Popping at Once* fact usually makes ATs "bigger" at the end than at the beginning. In the example, an AT is built for three LSPs (one of them partially tunneled). The label space has been dropped off from 20 to 17 (reduction factor of 15%).

It should be pointed out that: a) the proposal mentioned here fulfills all QoS guaranties since no route is changed and no new LSP is established, b) the stack depth is only increased by one (because only one label is pushed) in some segments and, c) no change is needed for the current MPLS architecture.

2.2.1 The Tunneling Problem

To compute an optimal AT solution, the Tunneling Problem presented in this subsection must be faced.

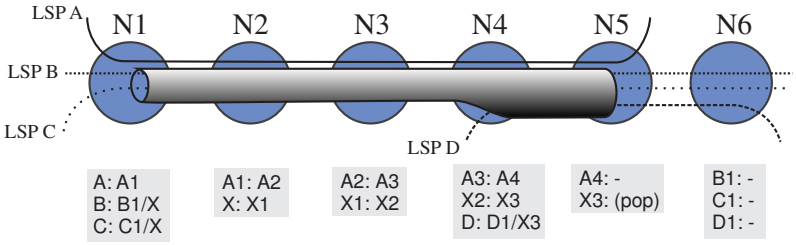


Figure 2.3: Asymmetric Tunnel Example.

Any segment of the network can be taken to create a tunnel, as long as there are at least two LSPs that can be stacked on it. If all the segments in the network comprising at least two LSPs are taken as tunnels, some tunnels would overlap. Since an LSP cannot be stacked in two tunnels over the same link, the associations between LSPs and tunnels is tricky.

For instance, consider the LSP configuration shown in Fig. 2.4(a). Two feasible solutions for the same problem may be regarded in Fig. 2.3 on page no. 28 and in Fig. 2.4(b). Note that both solutions cannot be considered together, since they would associate one LSP to more than tunnel.

In this example, the first solution (Fig. 2.3 on page no. 28) builds a tunnel that makes LSRs in the network use a total of 15 labels, while the second solution (Fig. 2.4(b)) makes them use 14 labels.

The Tunneling Problem can be stated as follows.

> Tunneling Problem - Given. A set of paths in the network, name it $\mathcal{P}$, and a bound on the number of labels, name it integer B,
> Question. Is there a set of tunnels, $\mathcal{T}$, such that
>
> - $\forall p \in \mathcal{P}$ can be associated to at most one tunnel per link and,
> - the number of labels used in the network does not exceed B?

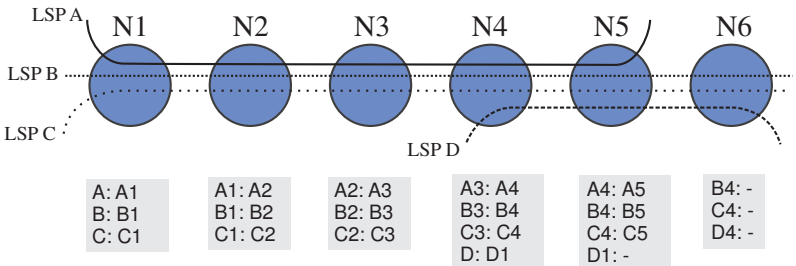
2.2.2 The Longest Segment First Algorithm

Since there are many ways to build asymmetric tunnels, this section proposes the first heuristic (chronologically) contemplated to build them [SFDM05a, SFDM05b]. The algorithm is based on the idea that *the more swapping LSRs a tunnel has, the more the label space is reduced*. Therefore, the LSF algorithm aims to find tunnels with the greatest number of swapping LSRs.

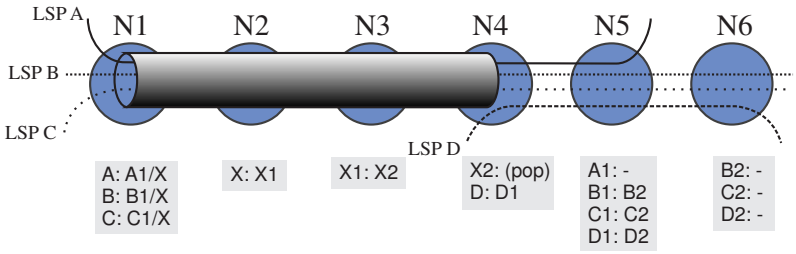
Henceforth, segments under notation $X : \{\lambda_i \rightarrow \lambda_{i+1} \rightarrow \cdots \rightarrow \lambda_j\}$ may be rewritten as $X : \{\lambda_i \ldots \lambda_j\}$ denoting the sequence of LSRs (a segment) starting at λ_i and ending at λ_j of LSP X. The route between the two nodes is implicit, otherwise the former notation is used in order to explicitly denote it.

In this subsection the LSF algorithm is explained through the example regarded in Fig. 2.5. It should be pointed out that a P2P segment may be implemented as a list (vector or sorted set) of the LSRs through it passes.

The LSF algorithm is an iterative algorithm over a set of non-stacked segments. Initially, let us consider the set of segments that has not been stacked



(a) Example of a Topology



(b) An Optimal Solution to the Problem

Figure 2.4: Solutions using AT for a given problem.

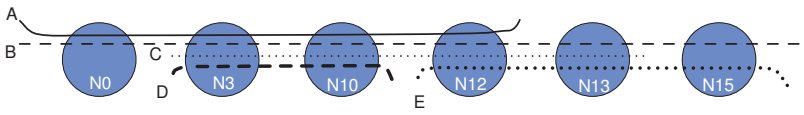


Figure 2.5: Scenario for describing the Longest Segment First algorithm.

before iteration k as $\mathcal{N}_k$. Clearly, in the first iteration, $\mathcal{N}_1$ gathers all the P2P connections and/or the decomposition of all the P2MP connections in the network. In Fig. 2.5, since the example has been made of P2P LSPs, $\mathcal{N}_0 = \{A : (N0 \dots N12), \quad B : (N0 \dots N15), \quad C : (N3 \dots N13), \quad D : (N3 \rightarrow N10), \quad E : (N12 \dots N15)\}$.

LSF builds an asymmetric tunnel in every iterations until it finds no more. Firstly, the algorithm finds two segments in $\mathcal{N}_k$, namely p_1 and p_2, that share the *longest* segment of LSRs. In other words, it finds out which pair of segments gives the *longest intersection* with length greater than two. Let us name the intersected segment t and, r the last LSR of segment t:

$$t = \arg \max_{p_i, p_j \in \mathcal{N}_k} \|p_i \cap p_j\|$$

In the example, LSF sets $p_1 = A : (N0 \dots N12)$ and $p_2 = B : (N0 \dots N15)$, which makes $t = (N0 \dots N12)$ (because those are the shared LSRs of both) with its last LSR $r = N12$, the *receiving LSR*. Note that, it is also possible to begin with $p_1 = B : (N0 \dots N15)$ and $p_2 = C : (N3 \dots N13)$.

If t cannot be found or if t comprises less than three LSRs (or two links), then the LSF algorithm ends.

So far, the first decision problem has been decided in the iteration: where to start and where to end the tunnel. For the second decision problem, LSF selects *all* the segments in $\mathcal{N}_k$ that can be tunneled, or *partially* tunneled, with t. All these segments are put in a new set named $\mathcal{L}$. In other words, it finds all segments in $\mathcal{N}_k$ that a) intersect t in more than three hops (two links), and b) include LSR r; more precisely:

$$\mathcal{L} = \{p \in \mathcal{N} \mid \|t \cap p\| \geq 3, \ r \in p\}$$

It is clear that $p_1 \in \mathcal{L}$ and $p_2 \in \mathcal{L}$. In the example, $\mathcal{L} = \{A : (N0 \dots N12), \quad B : (N0 \dots N15), \quad C : (N3 \dots N13)\}$ because they all intersect t in more than three consecutive LSRs including r $(N12)$.

Then, the tunnel is built across the LSRs in t with LSPs in $\mathcal{L}$. As a matter of fact, not all the path of every segment in can be covered, but only the portion covered by the links in t: $T = \{p \cap t \mid \forall p \in \mathcal{L}\}$.

In the example, $T = \{A : (N0 \dots N12), \quad B : (N0 \dots N12), \quad C : (N3 \dots N12)\}$. Note that $N13$ is not included in tunnel t in spite that $N13$ belongs to LSP B and C. The same happens for $N15$ and LSP B. NHLFEs of all LSRs contained in t should be set up later for segments in T using a signaling protocol.

The algorithm iterates by setting $\mathcal{N}_{k+1}$ to the segments not tunneled of $\mathcal{L}$,

$$\mathcal{N}_{k+1} = \{p - t \mid p \in \mathcal{L}\} \cup (\mathcal{N}_k - \mathcal{L})$$

In the example, for the following iteration, LSF sets $\mathcal{N}_1 = \{B : (N13 \rightarrow N15), \quad C : (N13), \quad D : (N3 \rightarrow N10), \quad E : (N12 \dots N15)\}$.

The algorithm re-iterates by selecting once more p_1 and p_2, if possible. For the example, the algorithm finalizes because the biggest intersection among segments in new $\mathcal{N}_1$ is too small to make a tunnel, i.e. $\|B : (N13 \rightarrow N15) \cap E : (N12 \rightarrow N15)\| \ngeq 3$.

Note that the average size of segments in $\mathcal{N}_k$ is smaller than those in $\mathcal{N}_{k+1}$. This assures LSF finalization. Because of its greedy-based heuristic, the complexity of LSF is $O(n^5)$ where n denotes the number of LSRs in the network.

Drawbacks of the LSF Algorithm Its main drawback is the computational complexity due the need of knowing the state of the whole network at a time. Since the tunneled segment to be built is selected as the longest in the network, the algorithm must know exactly the routes of all the LSPs. Furthermore, it has to intersect every possible pair of LSPs in order to find out the longest segment in the network, which increases its complexity.

Despite its complexity, it underestimates good solutions easily. As mentioned before, the LSF algorithm solves the first decision problem (the bounds of the tunnel), and based on its solution picks the better solution for the second decision problem (which LSPs are stacked). Since LSF selects the bounding LSRs of an asymmetric tunnel (i.e. first pushing LSR and popping LSR) as the two most distant LSRs for any 2 P2P segments in $\mathcal{N}_k$, the algorithm does not explore solutions bounded inside segments between those two most distant LSRs. As a matter of fact, sometimes the best reduction is achieved by building tunnels stacking the more LSPs as possible (the widest), instead of building tunnels stacking the more LSRs as possible (the longest).

For instance, Fig. 2.3 on page no. 28 and Fig. 2.4(b) on page no. 29 show two label stacking solutions for the same problem, as commented before. The solution in Fig. 2.3 corresponds to the given by the LSF algorithm, while the solution in Fig. 2.4(b) was computed manually. In this example, LSF algorithm builds a tunnel that makes LSRs in the network to use a total of 15 labels, while the second solution make them to use 14 labels.

2.2.3 The Most Congested Space First Algorithm

Although LSF finds good solutions, it does not explore solutions between the selected first and last nodes, as mentioned just before. The MCSF aims to solve these drawback with lower complexity [SFM05a]. This may improve not only response time, but also the number of reduced labels.

The algorithm comprises two procedures which must be used in sequence, one after the other. The first procedure guesses which the best *receiving* LSR to start the reduction with is: Receiving LSR Selection procedure (RLS). The second procedure aims at finding the best tunnel *to* the selected receiving LSR (chosen previously by the RLS procedure): Upstream Querying to Downstream Selection procedure (UQDS).

The following notation is going to be used to describe the procedures. Let $\mathcal{H}$ be the set of all NHLFEs in a MPLS network (previously configured for the given pre-computed LSP routes) in which each element, $h \in \mathcal{H}$, is represented as a tuple $(n, l, i, d) \in \mathcal{N} \times \mathcal{S} \times \mathcal{N} \times \mathcal{N}$ denoting that LSR n forwards LSP l from LSR i to LSR o. Given $h \in \mathcal{H}$ the functions $N(h) : \mathcal{H} \to \mathcal{N}$ and $I(h) : \mathcal{H} \to \mathcal{N}$ give the LSR that stores NHLFE h, named n, and its upstream LSR (previous hop), named i, respectively. In the same way $D(h)$ denotes the downstream LSR of NHLFE h (next hop), named d.

Let $\star \subseteq \mathcal{H} \times \mathcal{H}$ by the *equivalence relationship* defined as: $\star = \{(h_1, h_2) \mid I(h_1) = I(h_2), N(h_1) = N(h_2), h_1 \in (H), h_2 \in (H)\}$. In other words, two NHLFEs, h_1 and h_2, are *equivalent* (i.e. $h_1 \star h_2$) if, and only if, they are stored in the same LSR and the have in common the incoming interface (upstream LSR). The set of equivalent NHLFEs to a particular NHLFE h in a subset $\mathcal{H}'$ is denoted by $[h]_{\mathcal{H}'}$ (the subindex $\mathcal{H}'$ will be removed unless it is necessary). Like any equivalence relationship, $\star$ defines a *factor set* in $\mathcal{H}' \subseteq \mathcal{H}$ that will be denoted as $\mathcal{H}'/\star$.

For notation simplicity, assume that $I([h]_{\mathcal{H}'}) = I(h)$ and $N([h]_{\mathcal{H}'}) = N(h)$.

Receiving LSR Selection Procedure

The selection of the receiving LSR plays an important role in the reduction, since it delimits the tunnel at its end and all the subsequent decisions. The selection is based on an estimator. The estimator gives a value of how many labels can be saved by creating an AT ending in that LSR.

An estimation is done for all LSRs in the network. The reduction estimation of an LSR is computed from the set of NHLFE in that LSR. The estimator is set to the maximum number of entries in an LSR that have the same previous hop (i.e. incoming interface). If every incoming interface on an LSR uses a separate label space, the algorithm will start tunnels construction with the LSR with the most congested space; therefore its name.

Taking into account the notation previously described, the selected receiving LSR r is chosen as:

$$r = N\left(\arg \max_{[h]\in\mathcal{H}/\star} \|[h]_{\mathcal{H}}\|\right)$$

Once the most congested LSR is selected, the UQDS algorithm takes place with it and its congested LSPs. Since it depends on NHLFEs implementation, the complexity of this process is estimated to $O(n)$.

Upstream Querying to Downstream Selection Procedure

UQDS assumes that the receiving LSR of the new tunnel to be created has been selected previously by the RLSprocedure. In order to reduce the complexity and explore intermediate solutions, this algorithm is designed in a *hop-by-hop querying-response* basis. The algorithm is divided in two phases. The first phase queries all the upstream LSRs for the best reduction. The second phase selects the best path that the tunnel must follow in order to achieve the best reduction.

In the first phase (upstream querying phase), since there are many ways to setup a tunnel ending at the selected receiving LSR, a querying 'message' is distributed among all these possible tunnels. The query message is forwarded upstream and replicated at every hop. The route followed by all the replicated messages is an inverted tree path, where each leaf of the tree is a possible pushing LSR and the root is the receiving LSR itself. It should be pointed out that each branch of the tree is feasible tunnel.

In the second phase (downstream selection phase), each leaf forwards back to the root (downstream) the reduction computed in that branch. For this, each time the query message steps on a node, the query message is updated so it reflects the possible label space reduction (the reduction computed) made from the leaf (a feasible pushing LSR) until the implicated LSR. Because of the tree-shape, it is possible that an LSR receives more than one reduction computed from two different branches. In this case, the LSR selects the best branch by identifying the best computed reduction. This two mechanism can be explained with the following set of recursive formulaes.

First at all, let $P(\mathcal{B}, \mathcal{A}) = \{a \in \mathcal{A} \mid \exists b \in \mathcal{B},\ N(a) = I(b),\ D(a) = N(b)\}$; function P represents the subset of NHLFEs in $\mathcal{A}$ that are forwarding packets to

those in $\mathcal{B}$. Now, the number of reduced labels after k UQDS requests considering only a subset of NHLFEs in the network, $\mathcal{H}' \subseteq \mathcal{H}$, is:

$$R_k(\mathcal{H}') = \max_{[h']\in\mathcal{H}'/\star} V_k([h']_{\mathcal{H}'})$$

where

$$V_k(\mathcal{B}) = \begin{cases} 0, & \text{if } \|\mathcal{B}\| < 2 \\ R_{k+1}(P(\mathcal{B},\mathcal{H})) + \|\mathcal{B}\| - 1, & \text{otherwise} \end{cases}$$

and $\mathcal{H}$ is the set of *all* NHLFEs in the network. Note that trough iterations, the equivalence relationship $\star$ and the function P make $\mathcal{H}'$ smaller, which assures the termination of the procedure and the algorithm as well.

At the k-th iteration of the UQDS algorithm, the selected upstream LSR that takes part in the tunnel is computed by:

$$N_k(\mathcal{H}') = I\left(\arg \max_{[h']\in\mathcal{H}'/\star} V_k([h']_{\mathcal{H}'})\right)$$

To initiate the algorithm, the first parameter set is set to

$$\mathcal{H}_0^{'} = \{h \in \mathcal{H} \mid N(h) = r\} = \max_{[h]\in\mathcal{H}/\star} \|[h]_{\mathcal{H}}\|$$

As example, consider the configuration in Fig. 2.6 with five LSPs forwarded across six LSRs. Assume that, in a bigger network, the RLS procedure had chosen $N6$ as the most congested LSR in the network, and therefore, as the receiving LSR for UQDS.

Only the second iteration of the algorithm is explained in detail. At the second iteration of the algorithm ($k = 1$), the LSR to be analyzed is $N5$ which contains the following entries:

(id)	n	l	i	o
6	N5	A	N3	N6
7	N5	B	N3	N6
8	N5	C	N3	N6
9	N5	D	N4	N6
10	N5	E	N4	N6

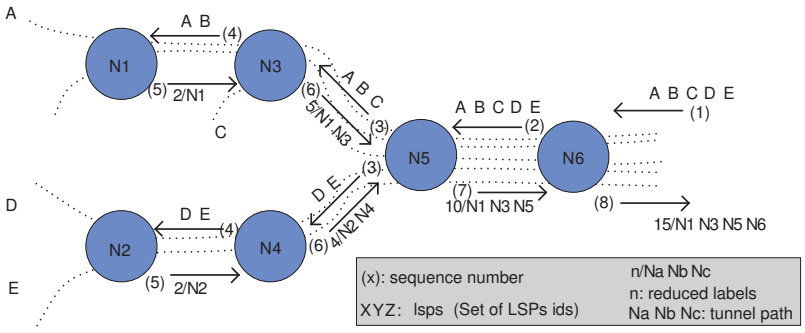


Figure 2.6: MCSF Recursive Invocations.

To simplify notation, the NHLFEs are denoted only by their index, hence $\mathcal{H}_1^{'} = \{6, 7, 8, 9, 10\}$. Since $6 \star 7$, $7 \star 8$ and $9 \star 10$, then $[6] = [7] = [8] = \{6, 7, 8\}$ and $[9] = [10] = \{9, 10\}$. Therefore $\mathcal{H}_1^{'}/\star = \{[6], [9]\} = \{\{6, 7, 8\}, \{9, 10\}\}$.

The reduction from $N5$ is then

$$R_1(\{6, 7, 8, 9, 10\}) = \max_{[e] \in \{[6],[9]\}} V([e])$$

with

$$\begin{aligned} V([6]) &= R_2(P(\{6, 7, 8\}, \mathcal{H})) + 2 \\ V([9]) &= R_2(P(\{9, 10\}, \mathcal{H})) + 1 \end{aligned}$$

Assuming that $\mathcal{H}$ contains:

(id)	n	l	i	o
11	N3	A	N1	N5
12	N3	B	N1	N5
13	N3	C	N3	N5
14	N4	D	N2	N5
15	N4	E	N2	N5

then $P(\{6, 7, 8\}) = \{11, 12, 13\}$, where $11 \star 12$, and $P(\{9, 10\}) = \{14, 15\}$, where $14 \star 15$. Recurrent call to R makes $V([6]) = 1 + 2$ and $V([9]) = 1 + 1$ hence $R_1(\{6, 7, 8, 9, 10\}) = V([6]) = 3$. From $N5$, the next step in the tunnel is then $N3$, since $I([6]) = N3$.

The complexity of UQDS is computed to $O(n^2)$ in the worst case. Taking UQDS complexity together with RLS complexity, the MCSF complexity is computed to $O(n^3)$.

UQDS Algorithm We proceed to show a pseudocode of the previously mathematically defined procedure.

The following list gives a brief description of the variables used in the pseudocode.

k LSR id.

N Segments id set. Feasible segments to be tunneled.

bestRed Integer. Best reduction found through iterations.

branch LSR id vector. Selected branch for being tunneled.

i LSR id. A neighbor LSR of LSR *k*.

allI Segments id set. Segments that are forwarded directly from LSR *i* to LSR *k*.

fromI Segments id set. Contains a subset of *N* that are forwarded from LSR *i*.

iRed Integer. The best reduction computed from a leaf until LSR *i*.

v LSR ids vector. Indicates the selected branch to be tunneled until LSR *i*.

Each UQDS call returns:

Function `McsfUqds`(j, N)

```
   Input: LSR j
   Input: Set N
   Output: Integer, Vector
1  begin
2      bestRed ← 0;
3      branch ← ∅;
4      foreach neighbor LSR i do
5          allI ← getSegmentsFromTo(i, j);
6          fromI ← intersectSets(allI, N);
7          if ‖fromI‖ > 1 then
8              {iRed, v} ← McsfUqds(i, fromI);
9              if iRed > bestRed then
10                 bestRed ← iRed;
11                 branch ← appendAtEnd(v, i);
12     bestRed ← bestRed + ‖N‖;
13     return {bestRed, branch};
14 end
```

- an integer value that estimates the number of reduced labels up to its call and,
- a vector of LSRs denoting the P2P segment that is candidate to be tunneled.

To give an explanation of the algorithm, the same configuration explained before with the formulas is considered.

Initially, UQDS procedure receives $N6$ as j and the segments with ids A, B, C, D, E as the set N, step (1) in Fig. 2.6. Since in Fig. 2.6 LSR $N6$ only has LSR $N5$ as neighbor (line 3 of the algorithm) and all segments in the set N are forwarded through it (lines 4 and 5), the procedure is invoked recursively with $j = N5$ and the same set N, step (2) in Fig. 2.6 (line 7).

In this second invocation, the set N is split in two because segments A, B and C are forwarded through LSR $N3$ and segments D and E through $N4$, step (3) in Fig. 2.6. This makes LSR $N5$ to query both LSRs ($N3$ and $N4$) and wait until they answer him in order to respond back to LSR $N6$. This process is repeated until LSRs $N1$ and $N2$ are reached, i.e. where the set N can not be split in subsets with less than two segments ids, step (4) in Fig. 2.6.

The selection process is started in LSRs $N1$ and $N2$. $N1$ computes its reduction in two because it may stack at most two segments (A and B). The same occurs with LSR $N2$ and segments D and E, step (5) in Fig. 2.6.

When LSR $N3$ receives $N1$ response, $N3$ computes its reduction as $N1$ reduction (valued in two) plus three entries for segments A, B and C. In the same way, LSR $N4$ computes its reduction in four. Both LSRs inform $N5$ about their reductions and the path they have follows, step (6) in Fig. 2.6.

LSR $N5$ receives both reductions and saves the best one (from $N3$). Since $N5$ chooses $N3$ as the best one, it answers to $N6$ with a reduction of 10 (five

until $N3$ and five more for $N5$ itself) and the path computed by $N3$ adding LSR $N5$: $N1 \rightarrow N3 \rightarrow N5$, step (7) in figure Fig. 2.6.

Finally, $N6$ computes the final reduction and path from $N5$ answer in 15 and $N1 \rightarrow N3 \rightarrow N5 \rightarrow N6$ respectively.

The complexity of UQDS is computed to $O(n^2)$ in the worst case and considering linear growth complexity in the *intersection* method. Taking UQDS complexity together with RLS complexity, the MCSF complexity is computed to $O(n^3)$.

2.2.4 Simulation Results

The two proposed algorithm (LSF and MCSF) were tested and compared using several P2MP configurations. Each test varies the network load and for each network load the number of labels used in the network was computed. Each test has the same number of P2MP demand plus a new one. All demands request the same bandwidth, but they differ in the ingress LSR and the multicast egress LSR set.

Since the number of labels increases as the number of P2MP LSPs increases, the results are focused on reduction factors experienced when LASPARED methods are used.

A random network is generated according to Siganos *et al.* power laws [SFFF03]. The generated network consists of 50 LSRs and 150 bidirectional physical links (rank exponent around -0.75). In this case, the multicast egress LSR set is selected randomly. A multi-objective optimization algorithm is used to compute the 'best different ways' to setup each P2MP configuration in the generated network [DFM04]. For each test carried out the optimization algorithm computes a set of feasible solutions to accommodate all flows, therefore the average of all label space reduction factors for each solution in the set is taken.

Fig. 2.7 illustrates the best reduction factor found using the LASPARED methods discussed in this chapter when network load was increased from one P2MP LSP to 32 P2MP LSPs. The reduction factor is computed as the division between the number of labels left apart by the total number of labels when no LASPARED solution is contemplated. In figure Fig. 2.7, it can be seen that the reduction factor of the LASPARED methods described follow a logarithmic curvature. This means that the total number of labels is not increased a lot when more P2MP LSPs are added, i.e. a near-static number of labels per LSR can be used.

Moreover, the simulation results shows that MCSF achieves a better reduction with lower complexity than LSF in most common cases. The reduction factor was improved in 10% in relation to the LSF. Results also show that MCSF is far away from low reduction factors achieved by traditional label space reduction methods such as MP2P trees.

2.3 Asymmetric Merged Tunneling

So far, neither asymmetric tunneling gets advantage of labels merging feature, nor labels merging of asymmetric tunneling. In this section, a mixed version of both methods that preserves their advantages is proposed.

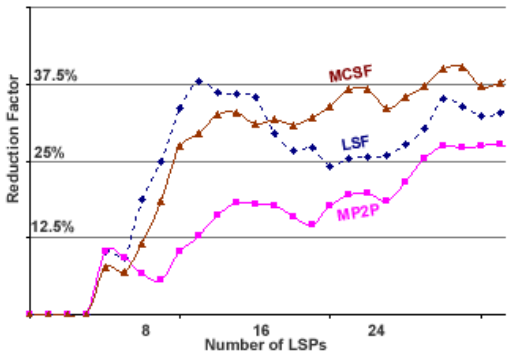


Figure 2.7: Maximum reduction factor achieved by different methods (AT-Model, LSF and MP2P) when the network load increases (number of LSPs)

Our proposal may be seen as a merging of asymmetric tunnels into a single MP2P connection, therefore decreasing even more the number of labels used in the network. Also, it may be seen as a way to create MP2P trees where the root LSR may not be the egress LSR of a set of LSPs, i.e. MP2P trees anywhere in the network.

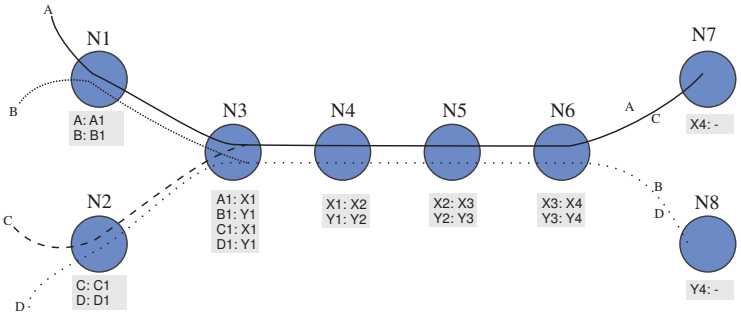
Assume there are four LSPs (e.g. see figure Fig. 2.8):

- $A : \alpha_{0...A} \rightarrow \mu_{0...M} \rightarrow \lambda_{0...L} \rightarrow \alpha'_{0...A'}$,
- $B : \beta_{0...B} \rightarrow \mu_{0...M} \rightarrow \lambda_{0...L} \rightarrow \beta'_{0...B'}$,
- $C : \gamma_{0...C} \rightarrow \nu_{0...N} \rightarrow \lambda_{0...L} \rightarrow \gamma'_{0...C'}$, and
- $D : \delta_{0...D} \rightarrow \nu_{0...N} \rightarrow \lambda_{0...L} \rightarrow \delta'_{0...D'}$

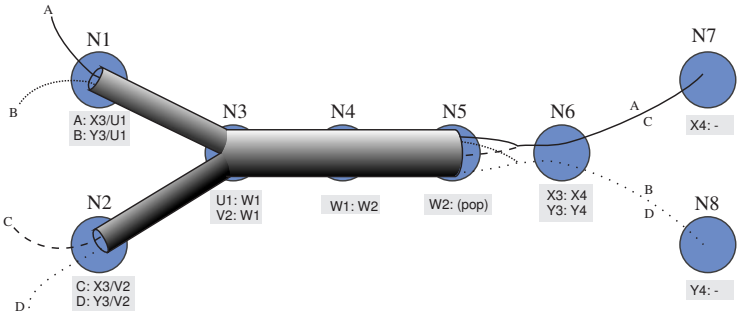
Then, an AMT [SFM05b] tree can be built when:

1. μ_0 pushes a new label into LSPs A and B packet stacks,
2. ν_0 pushes another new label into LSPs C and D stacks,
3. since labels are upstream assigned, λ_0 could ask both μ_M and ν_N to forward packets with the same label. Then, λ_0 may regard both flows as the same (using one NHLFE).
4. finally, λ_{L-1} does a pop of packets stack, so λ_L may receive packets with the original label (given by α_A, β_B, γ_C and δ_D) and, hence, forward them to its correct destination (i.e. α'_0, β'_0, γ'_0 or δ'_0).

The most closely related work to this type of reduction is presented by Gupta *et al.* in [GKR05] (and similarly in [GKR03] and [GKT03]). They studied the trade-off between label space sizes and stack depth in some special network configurations. They focus in network configurations in which all LSRs are interconnected either: *a*) along a path, or *b*) along a tree. Comparing it with our



(a) MP2P Solution



(b) AMT Solution

Figure 2.8: AMT Example

contributions, our problem is more general since we face the problem of *defining* the AMT trees in the network, as mentioned before, while Gupta *et al.* assume that the trees are already given.

The problem of obtaining the best label space reduction using AMTs is clearly harder than any considered before, since there are more decision problems to tackle at the same time. Since an AMT is regarded as a tree-like structure, deciding the boundaries for an AMT is more complex than deciding the boundaries for an AT.

2.3.1 The Brute-Force Model

In this section, our first thought for modeling the LASPARED problem with the considered constraints is presented. We denote it as the *Brute-Force model* (or BF-model for short), since the computation takes the whole set of LSP routes and the complete network as input.

Parameters and Variables

Let $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{P}$ be the set of indexes for: tunnels, LSRs, and LSPs path respectively. The symbol $\mathbb{B}$ is used to denote the set $\{0, 1\}$. We use the following indexes in the model.

- $i, j, k \in \mathcal{N}$. Nodes in the network.

- $p \in \mathcal{P}$. An LSP path routed in the network.
- $t \in \mathcal{T}$. A tunnel.

Since the number of tunnels (or AMTs) is unknown *apriori*, the set $\mathcal{T}$ must be dimensioned to a sufficiently large number.

Let $L^p_{(i,j)} \in \mathbb{B}$ be a (given) binary model parameter with a value of one if the link (i,j) forwards packets of the path p, and 0 otherwise. Since each link that a path uses makes an LSR use an additional label, the number of labels used in a network without AMTs equals $\sum_{p,(i,j)} L^p_{(i,j)}$, e.g. 16 in all the subfigures of Fig. 2.4.

The variables in the model are the following.

- $u^{p,t}_{(i,j)} \in \mathbb{B}$. Set to 1 when link (i,j) of path p is *stacked* in AMT t, 0 otherwise.
- $v^t_{(i,j)} \in \mathbb{B}$. Set to 1 when AMT t is using link (i,j), 0 otherwise.
- $x^t_j \in \mathbb{B}$. Set to 1 when LSR j is the receiving LSR (i.e. the last LSR of a tunnel) for AMT t, 0 otherwise.
- $y^p_t \in \mathbb{B}$. Set to 1 when AMT t is stacking path p, 0 otherwise.

Henceforth, the membership symbol ($\in$) is going to be omitted for notation simplicity, unless it is strictly necessary.

When a tunnel t is used, note that the number of *labels used only by the tunnel* t is $\sum_{(i,j)} v^t_{(i,j)}$, e.g. in Fig. 2.3 on page no. 28 this expression has a value of five.

Note that the last link in the figure has no shade ($N5 \rightarrow N6$ in Fig. 2.3), however we count it as part of the tunnel. We name this special link the *receiving link*, and the last LSR the *receiving LSR* (variable x^j_t) of the tunnel. Even though there is no assigned label in the receiving link of a tunnel (as shown in the figure), it has to be considered as part of it since all the "stacked" LSPs must use this link in order to comply with MPLS tag-forwarding.

Similarly, the expression $\sum_{p,(i,j)} u^{p,t}_{(i,j)}$ adds up to the number of *labels that are not being used* by LSPs p since they are being tunneled by t, e.g. in Fig. 2.3 this expression has a value of 12.

Optimization Model

With these parameters and variables defined, the optimization problem can be represented as follows: given a set of paths $\mathcal{P}$, the best reduction possible is computed by finding the values of $u^{p,t}_{(i,j)}$, such that they minimize the total number of labels, ε, in the network:

$$\varepsilon = \sum_{p,(i,j)} L^p_{(i,j)} - \hat{\Delta}$$

where

$$\hat{\Delta} = \underbrace{\sum_{p,t}\left(\sum_{(i,j)} u^{p,t}_{(i,j)} - y^{p,t}\right)}_{\beta} - \underbrace{\sum_{t}\left(\sum_{(i,j)} v^{t}_{(i,j)} - x^{t}_{j}\right)}_{\alpha}$$

The expression Δ corresponds to the number of *reduced labels* by the usage of AMTs. Within Δ, the expression outlined by α counts the number of used labels for *tunnels*; while the expression outlined by β counts the number of labels for *paths* that are "covered" by a tunnel. In other words, subexpression β counts the number of path hops that are stacked (hence, not used anymore). Because there is no reduction at the receiving LSR, the subtraction of $\sum_{p,t} y^{p,t}$ is done once in order to make Δ values accurate. In the same way, since tunnels do not have stacked labels in the receiving LSR link, x^t_j is subtracted from α. The relationship between these two subexpressions can be seen as: while subexpression β saves path labels, subexpression α pays for those savings by using tunnels. Hence, the difference between them, expression Δ, gives the overall number of labels reduced when AMTs are built in the MPLS network.

Overriding fixed values in previous formula, a Zero-One Programming model can be formulated as:

Maximize:

$$\hat{\Delta} = \sum_{p,t,(i,j)} u^{p,t}_{(i,j)} - \sum_{p,t} y^{p,t} - \sum_{t,(i,j)} v^{t}_{(i,j)} + \sum_{t,j} x^{t}_{j} \tag{2.1}$$

subject to:

$$u^{p,t}_{(i,j)} - u^{p,t}_{(j,k)} + v^{t}_{(j,k)} \leq 1, \qquad \forall i,j,k,p,t \tag{2.2}$$

$$2 \cdot y^{p,t} - \sum_{(i,j)} u^{p,t}_{(i,j)} \leq 0, \qquad \forall p,t \tag{2.3}$$

$$\sum_{i} v^{t}_{(i,j)} - \sum_{k} v^{t}_{(j,k)} - x^{t}_{j} \leq 0, \qquad \forall j,t \tag{2.4}$$

$$\sum_{j} x^{t}_{j} \leq 1, \qquad \forall t \tag{2.5}$$

$$\sum_{t} u^{p,t}_{(i,j)} \leq 1, \qquad \forall i,j,p \tag{2.6}$$

$$u^{p,t}_{(i,j)} - L^{p}_{(i,j)} \leq 0, \qquad \forall i,j,p,t \tag{2.7}$$

$$u^{p,t}_{(i,j)} - v^{t}_{(i,j)} \leq 0, \qquad \forall i,j,p,t \tag{2.8}$$

$$u^{p,t}_{i,j} - y^{p,t} \leq 0, \qquad \forall i,j,p,t \tag{2.9}$$

The objective function, (2.1), minimizes the number of labels used in the network by maximizing the number of labels reduced.

In (2.2), the model indicates to tunnel all LSPs that were tunneled previously in link (i,j) in the link (j,k) , only if a path is being tunneled in link (j,k). By adding this equation over index j, it can be demonstrated that the model avoids P2MP tunnels, if the paths in $\mathcal{P}$ are P2P connections.

It should be noted that since route paths are given, the model does *not* actually deal with LSP routing. Moreover, comparing it with most commonly used

networking routing models, the 'source node' (first pushing LSR) and 'destination node' (receiving LSR) of a tunnel are not known, hence, they are part of the problem itself. In our model, (2.2) can be seen as a sort of node-link routing constraint in those types of routing models.

The remaining equations are easier to read. Since a tunnel requires at least 3 nodes, (2.3) does not allow solutions to tunnel less than 2 links per path. By taking (2.8) into (2.3), it is easy to see that the total length of a tunnel is at least 3 nodes.

(2.4) gives a definition of x_j^t by allowing it to take the value of 1 only when node j has an incoming link but no outgoing link. Considering this, (2.5) states that there can only be 1 receiving LSR and therefore only one tunnel per each t index.

(2.6) prevents two tunnels from stacking the same path over the same link. (2.7) restricts $u_{(i,j)}^{p,t}$ search space according to the values of parameter $L_{(i,j)}^{p}$, i.e. assures that only those links used by paths may be tunneled. (2.8) and (2.9) give a definition of $v_{(i,j)}^{t}$ and $y^{p,t}$ respectively assuming that $u_{(i,j)}^{p,t}$ is known.

Reduction using MP2P

To measure the improvement between the reductions achieved when a single stacked label is used (AMT) compared to when there is no stacking solution (MP2P) (currently not found in the literature) the best reduction for MP2P must be computed. The model described in the previous subsection can be simplified for this special case.

To compute the reduction for MP2P, the model is relaxed so *only* the AMTs created are rooted at egress nodes of any LSP route. This is formalized by the following restriction over the model:

$$x_j^t \leq E_j,\ E_j = \begin{cases} 1, \text{ if LSR } j \text{ is an egress LSR} \\ 0, \text{ otherwise} \end{cases} \tag{2.10}$$

In addition, since MP2P computation involves only the LSPs that have the same egress LSR e in common, the set $\mathcal{P}$ could be partitioned into several subsets $\mathcal{P}_e$ (each of them containing only the paths ending at e) and we then apply the model for each subset $\mathcal{P}_e$, one set at a time. This may reduce the computational time required to solve the ILP model. Note that the objective function still holds for the reduction using MP2P.

Model Shortcomings The model presents two limitations. First, the set $\mathcal{T}$ must be dimensioned *before* solving the model. This implies an estimation of the number of tunnels that the optimal solution would have. A large set makes the model size larger, making the solver to spend more resources and time in solving it. A small set would lead to non-optimal solution. Therefore, dimensioning the set properly is a delicate. Second, the model considers label merging as tree structures. This is, all the AMTs are trees with non-intersecting branches. As discussed previously for MP2P, this could represent a small decrement in the number of labels reduced.

2.3.2 The Decompose & Match Framework

Looking at the BF-model, we see that it uses 4-dimensional variables. Thus, it is not hard to see that any ILP solver will need a long time and large space to give a solution for large networks; especially considering the quadratic space in terms of LSRs. Most of the times, this makes the model costly in terms of computational time and resources. Therefore, we analyzed the problem and propose in this section a fast and optimal way to do it: *the Decompose and Match framework*.

The framework described here to reduce the label space is divided in two parts: the *decomposition algorithm* and the *matching model*. The *matching model* is a path-based ILP model. Because of its path-based nature, the model needs a set of AMTs already defined (i.e. precomputed) as input, in addition to the LSP paths. With these parameters, the model matches the given AMTs with the given paths such that the maximum number of labels that can be saved is computed. The optimal matching may not include all the precomputed AMTs, since an LSP cannot be stacked by two AMTs at the same time. Clearly, computing *all* the feasible AMTs in a network would require an exponential algorithm, because computing all the feasible paths in a network is an exponential process. Instead, the *decomposition algorithm* computes a subset of all the feasible AMTs, containing the optimal solution for the problem.

The Decomposition Algorithm

As mentioned before, it should be pointed out that the optimal solution to the LASPARED problem uses a *subset* of the AMTs that will be computed in this phase by the decomposition algorithm. The smaller this feasible optimal set, the better the runtime performance.

Definition 6 (Segment) *We consider a* segment *as a sequence of* 2 *or more network links (formally an ordered set of* 2 *or more link elements) denoting a route in a network using at least three consecutive nodes.*

In this sense, all the given LSP paths are segments, but not the opposite. If s_k is a segment, $\|s_k\|$ represents the number of links that it comprises.

Definition 7 (COnverging Segment - COS) *We say that a segment s_k is* Converging *considering a set of paths $\mathcal{P}$ if $\forall p_i \in \mathcal{P}$, then $N_{p_i} \cap s_k = \emptyset$ or $s_k \subseteq N_{p_i}$.*

In other words, let us consider a segment s_k and all the LSPs $\hat{P} \subseteq \mathcal{P}$ that use at least one of the links in s_k. A segment is said to be *converging* when all the LSPs $\hat{P}$ use all the links in s_k, i.e. when *all LSPs converge* into (or, no LSPs diverges from) the path followed by s_k. For example, in Fig. 2.4, the segment $N1 \rightarrow N2 \rightarrow N3$ is a converging segment since LSPs A, B and C includes it and D is completely disjoint to it.

Definition 8 (MAximum COnverging Segment - MACOS) *Let s_k be a converging segment, let $s_k^{\rightarrow}$ be a segment formed by the links in s_k plus one more downstream link, and $s_k^{\leftarrow}$ be a segment formed by the links in s_k plus one more upstream link. Then, s_k is a* Maximum Converging Segment *if neither $s_k^{\leftarrow}$ nor $s_k^{\rightarrow}$ are converging segments.*

For example, in Fig. 2.4, the segment $N1 \rightarrow N2 \rightarrow N3 \rightarrow N4$ is a maximum converging segment since $N1 \rightarrow N2 \rightarrow N3 \rightarrow N4 \rightarrow N5$ ($s_k^{\rightarrow}$ in our definition) is not a converging segment (because LSP D contains a part of it).

To assure the correctness of the algorithm, an intermediate theorem has to be demonstrated.

Theorem 1 (MACOS Optimality) *Let s_k be a MACOS considering the paths $\mathcal{P}$, and let $\hat{s}_k$ be a segment formed by the links in s_k minus one link ($\hat{s}_k \subsetneq s_k$), then either a tunnel constructed following the links of $\hat{s}_k$ is not optimum, or $\hat{s}_k$ is another MACOS.*

Proof. Let L and $\hat{L}$ be the set of indexes of all the LSPs that are forwarded through the links in s_k and $\hat{s}_k$ respectively. Let us consider the relationship between L and $\hat{L}$. On the one hand, if $L \neq \hat{L}$ then our proof concludes trivially since $\hat{s}_k$ is another MACOS. On the other hand, assuming $L = \hat{L}$, $\hat{s}_k$ is not an MACOS (by definition). In this case, the number of labels reduced with a tunnel following the links of $\hat{s}_k$ is $(\|L\| - 1) \cdot \|\hat{s}_k\|$, since $L = \hat{L}$. Similarly, the number of labels that can be reduced using s_k is $(\|L\| - 1) \cdot \|s_k\|$. This reduction is greater than the reduction offered by $\hat{s}_k$, because $\|\hat{s}_k\| < \|s_k\|$. Since the LSPs contained in $\hat{s}_k$ are exactly the same as those in s_k (MACOS definition), we conclude that constructing a tunnel following the links of $\hat{s}_k$ is not optimal. ■

The above theorem extends our space of optimal solutions.

Corollary 1 (Joint MACOS Optimality) *Let $\mathcal{S} = \{s_0, s_1, \ldots, s_k\}$ be the set of all MACOS found for a set of LSP routes $\mathcal{P}$. Let $\mathcal{S}^\star$ be the set of all the segments resulting from joining two or more consecutive MACOS in $\mathcal{S}$. Let x be a segment not in $\mathcal{S}^\star$, i.e. $x \notin \mathcal{S}^\star$. Then, a tunnel following the links in x is not optimal.*

The proof follows the same sequence as the previous one. The previous corollary leads us to consider consecutive MACOS as part of our optimal solution. So far, we have proved how to discard non-optimal solutions. As a consequence, the resulting search space has been shrunk to combinations of MACOS, whose computation is shown shortly.

Since two paths might be intersected in more than one segment, an ordinary intersection operation between two paths may lead to non-existent routes. Therefore, it is necessary to define operators that compute, not a single non-existent route, but a set of existent segments shared between two paths. We define a multi-intersection binary operator ($\dot{\cap}$) such that if $s_i \dot{\cap} s_j = \{s_0^{'}, s_1^{'}, \ldots s_n^{'}\}$, then each s_i' represents an ordinal intersection, i.e. a consecutive sequence of links forming a path inside both s_i and s_j paths (or segments, in general). Similarly, we define a multi-difference binary operator ($\dot{-}$) such that if $s_i \dot{-} s_j = \{s_0^{'}, s_1^{'}, \ldots s_n^{'}\}$, then each s_i' represents a consecutive sequence of links forming a path for one of the two LSP routes s_i or s_j, but never both.

With this new definition of the intersection and difference operator between segments, we are ready to propose and demonstrate an efficient way to compute the claimed set of AMTs:

Theorem 2 *The optimal solutions expressed by MACOS Optimality (theorem 1) and Joint MACOS Optimality (corollary 1) lemmas are computed by:*

$$\begin{aligned} \forall p_i, p_j \in \mathcal{P}, p_i \neq p_j,\ s'_k \in (p_i \dot{\cap} p_j),\ \|s'_k\| \geq 2 \\ \rightarrow s'_k \in \mathcal{S}', \end{aligned} \tag{2.11}$$

$$\forall s'_i \in \mathcal{S}', \rightarrow s'_i \in \mathcal{S}, \tag{2.12}$$

$$\begin{aligned} \forall s'_i, s'_j \in \mathcal{S}', s_k \in (s'_i \dot{-} s'_j),\ \|s_k\| \geq 2 \\ \rightarrow s_k \in \mathcal{S}, \end{aligned} \tag{2.13}$$

Proof. Let s_k be a MACOS (or a set of consecutive MACOS) and (a, b) a link that either follows or precedes s_k. Let $L = \{l_0, l_1, \ldots, l_n\}$, with $n \geq 2$, be a set of LSPs going through all links in s_k. Since s_k is a MACOS, then the LSP routes forwarded by link (a, b) must be $L' \neq L$. At this point we have to consider two cases: at least one LSP diverges in (a, b), i.e. $L' \subset L$, or at least one LSP is added in (a, b), i.e. $L' \supset L$. A third case is when both of the above cases happen simultaneously and it can be demonstrated using Case 1 or Case 2.

Case 1 - $L' \subset L$. Let l_i be one of the LSP routes that diverges, i.e. $l_i \in L - L'$. Then, it is clear that $\forall l_j \in L, l_i \cap l_j = x$, in this case (2.11) and (2.12) compute the value.

Case 2 - $L' \supset L$. Let l_i be one of the LSP routes that is added at (a, b), i.e. $l_i \in L' - L$. Let l_j be one of the LSP routes that goes through links in s_k, i.e. $l_j \in L$. Then, s_k can be found in one of the $l_j \dot{-} l_i$ results, in this case (2.11) and (2.13) compute the value. ■

Up to this point, we have computed a set of P2P segments that could be feasible optimal solutions. To extend the P2P segments into AMTs, we need to make combinations of these segments taking into account that the combined segments: *a*) must have the same end node and, *b*) the segments may be intersected at most in one place - the same conditions for MP2P trees. This may be done with the following recursive formula (assuming initially $\mathcal{T} \leftarrow \mathcal{S}$):

$$\begin{aligned} \forall t_i, t_j \in \mathcal{T}, \sup t_i = \sup t_j,\ \|t_i \dot{\cap} t_j\| = 1 \\ \rightarrow (t_i \cup t_j) \in \mathcal{T} \end{aligned} \tag{2.14}$$

Since routes are considered as ordered sets, the expression $\sup t_i = \sup t_j$ means "if t_i and t_j end at the same place". Due to this last equation, the algorithm runs in exponential-time in terms of the number of segments. However, it is possible to use a polynomial time algorithm if the decision of which P2P segments can be merged as AMTs is left to the ILP solver. Our simulations showed that the exponential-algorithm is the best suitable for the framework, since the number of combinations it has to perform in the last step is restricted only to MACOS ending in the same LSR.

The LSP-AMT Matching Model

Assuming that a set of AMTs has been computed considering a set of given LSP routes, the main issue is:*which subset of non-interfering AMTs should be used in order to achieve the best reduction? Which LSP routes should be tunneled on each AMT?* In other words, the problem has been simplified so we need to find only the best non-interfering *matchings* between pairs of LSPs and AMTs which

reduce, as much as possible, the number of labels used. In this section we solve this question by proposing a Zero-One Integer Programming model, called the *LSP-AMT Matching* model (LAM-model).

The list of indexes of the model is:

- $e \in \mathcal{E}$. A link in the network.
- $p \in \mathcal{P}$. A given LSP path.
- $t \in \mathcal{T}$. A pre-computed AMT by the *decomposition algorithm.*

The model uses the following list of parameters.

- $L_{t,e} \in \mathbb{B}$. Set to 1 if link e is used by the feasible pre-computed tunnel t, 0 otherwise.
- $R_{p,t} \in \mathbb{N}^{+0}$. A natural number parameter set to the number of labels that can be reduced by using AMT t with LSP p. These values can be easily found as the number of common links between them minus one[1]. Note that if tunnel t cannot reduce the label space for any LSR in the path p, $R_{p,t} = 0$.

In order to simplify notation, let us assume that $L_t^* = \sum_e L_{t,e} - 1$.
The model uses two decision variables.

- $x_{p,t} \in \mathbb{B}$. Set to 1 when the path p is used in AMT t, 0 otherwise.
- $z_t \in \mathbb{B}$. Set to 1 when AMT t is used for any path in the network, 0 otherwise.

The model is formulated as follows.
Maximize:

$$\hat{\Delta} = \underbrace{\sum_{p,t} (R_{p,t} \cdot x_{p,t})}_{\beta} - \underbrace{\sum_{t} L_t^* \cdot z_t}_{\alpha} \tag{2.15}$$

subject to:

$$\sum_t L_{t,e} \cdot x_{p,t} \leq 1, \qquad \forall e, p \tag{2.16}$$

$$x_{p,t} - z_t \leq 0, \qquad \forall p, t \tag{2.17}$$

The objective function, (2.15), computes the number of labels reduced. The equation has the same two subexpressions (*viz.* α and β) explained in the previous section.

The constraint imposed by (2.16) allows a path to be stacked using only one tunnel in every link. (2.17) assures that the quantity L_t^* is paid for every tunnel used, independently of how many LSPs are stacked with it.

[1]The constant value of one (1) must be subtracted because the last hop does not count in the reduction

About MP2P Label Space Reduction using D&M

As we outlined in §1.2, many authors working on MP2P reductions have addressed the importance of computing the optimal label space reduction using MP2P trees. One of our last contribution in this chapter concerns with computing this value using our framework.

In order to do this, a slight simplification must be made to the decomposition algorithm. The simplification consists on just considering a subset of LSPs and AMTs that end at the *same* LSR of the network. As stated in §1.2.5, the reduction of MP2P can be computed separately in sets of egress LSRs; therefore the set of paths $\mathcal{P}$ can be partitioned into disjoint subsets $\mathcal{P}_d$ taking the egress LSR d of each path as the discerning criteria. Then, by taking directly (2.14) with an initial value $\mathcal{T} \leftarrow \mathcal{P}_d$ a set of MP2P trees are computed for destination d. The process must be repeated for each destination $d \in \mathcal{N}$.

Framework Complexity and Performance

It may seem that an optimization model that computes the AMTs directly from the LSP routes (BF-model) is a bit more complicate to formulate but equally efficient in performance as the framework presented in previous section (D&M). However, this is not true. In this subsection the performance of the D&M framework is compared with both the BF-model and the model proposed by Saito *et al.* for MP2P label space reduction in [SMY00].

In order to compare the solutions offered to the LASPARED problem, throughout this subsection the variables $s, t,$ and n denote the number of LSP routes, AMTs and LSRs in the network respectively.

Regarding the work of Saito *et al.* in [SMY00], the number of variables used is in $O(tn^3)$ because: a) their model needs to set the used links for a MP2P and b) in order to do a fair comparison, the model for all the egress LSR is considered at a time. The third equation in §III.B of [SMY00] rises the number of constraints to $O(stn^4)$ since the restriction holds for every link, ingress node, MP2P solution and egress node in the comparison scheme.

Regarding the BF-model, variables index over all the network nodes and the different LSP are needed to take into account the LSP routes. Moreover, an additional index is needed to control the relationship between route's segment and its AMT. Therefore, the space used by the variables is in $O(stn^2)$. Because the BF-model needs to preserve the stacked LSPs between links in a AMT, the BF-model needs a sort of three-node flow conservation restriction. This restriction increases the number of constraints in $O(stn^3)$.

The *LAM-model* presented is composed by two binary variables. One of them uses $O(st)$ and the other one $O(t)$. In addition, the number of restrictions used by the model is $s \cdot (l + t)$, where l stands for the number of used links in the network; usually $l \leq n^2$. This lead us to $O(st)$ in the number of constraints, assuming that the number of feasible AMTs is greater than the number of used links in the network. In the same way, the space of the input parameters is $(s + l) \cdot t \in O(st)$. Clearly the LAM-model is better than both the BF-model and Saito's model in terms of space and, hence, runtime performance.

The complexity of the *decomposition algorithm* is a price it should be payed for the efficiency of the model, and the framework in general. Despite this, the simulation experiments (commented in next section) were executed very fast in

comparison to those of the BF-model. We note that the framework could run very fast - around five seconds to run all the framework even for networks with 32 LSRs and 256 LSPs - while the BF-model only could run up to a network with seven LSRs and 10 LSPs in the same machine (Solaris with 1GB of physical available memory) using the same solver (CPLEX). Saito's work also showed a good runtime performance, but not as good as ours. Moreover, it should be kept in mind that Saito's work achieves LASPARED by means of MP2P *only* - not using the stack, or AMTs - which decreases the possibilities of reducing the label space; as it will be seen in next section. Although processing time is dependent on the machines and solver characteristics, these values become very small for hourly or daily operations for a ordinary ISP network.

2.3.3 Simulation Results

In this section we present a set of simulation results that aim at showing the benefit of using AMTs. In our simulations, we considered the following aspects:

About the Network Topology. The network topology used is based on the Australian ISP topology discovered with the Rocketfuel engine [SMWA04], showed previously in Fig. 1.5. The topology used here differs from the original in that ours has one node for every different location, i.e. a node may represent a set of different interconnected nodes within the same physical location in the original topology. Our topology has 28 LSRs, in which the nine with the least degree are always selected as edge LSRs.

About Routing. In order to ensure valid LSP paths over any generated network, the LSP routes were computed using a multi-objective evolutionary algorithm in which several QoS metrics were taken as objective functions for placing LSP routes [DFM04]. This ensures that our LSPs routes may obey, to some degree, network operators' concerns. Explaining this routing algorithm is out of the scope of this dissertation, but the reader can see its specification in [DFM04]. It should be pointed out that, since the routing algorithm is multi-objective [ZT99], a set of possible routes may be computed for a single set of demands. In this case, the average of the reductions is taken into consideration.

About the Analyzed Metrics. The following metrics were considered in the simulation results.

1. *Number of labels used*: number of labels used when the label space reduction methods were used (MP2P and AMT).

2. *Average packet overhead with AMTs*: this may be measured as the number of stacked hops over the number of total hops in an AMT solution. Note that the average packet overhead of MP2P solutions is constant, since the stack sizes are not increased.

The rest of this section is divided as follows. The first subsection evaluates the gain of labels versus the overhead due to stacking. The second subsection shows that AMTs helps reduce more labels at the core nodes. While the analysis presented in these first two subsections is made varying the number of LSPs, the last subsection presents detailed results for a fixed high-number of LSPs.

Overall Reduction and Overhead

In this subsection, the minimum number of overall labels in the network is computed for different scenarios. Each scenario is comprised of different number of routed LSPs in the same topology.

In Fig. 2.9 we show at the top the number of labels used by each method when the number of LSPs is varied from 10 to 300. At the same time, the bottom figure shows the average size of the incurred MPLS header.

The growing ratios are linear with the number of labels in all cases, but differ in their factor according to the LASPARED method used. As expected, AMTs use at least the same number of labels that MP2Ps. When the network load is high (300 LSPs), the label space reduction is 70.6% for AMTs, while MP2P is just 48.75%.

The cost of reducing 21.85% (70.6% − 48.75%) the label space with AMTs is an overhead in the MPLS header. The average header size, as shown at the bottom of the figure, ranges from 5 bytes to 7 bytes in total (1 to 3 bytes of overhead, respectively) more than the normally used for MPLS tag-forwarding.

To analyze the overload caused in the traffic, we carried out a similar analysis to the one performed by Van Caenegem *et al.* in [CCPD06]. We experimented with several traffic distributions. The traffic distributions consider three different sizes of payloads: 40 bytes, 520 bytes and 1500 bytes. The percentage of packets with payloads of 40 bytes can be 40%, 50% and 60%, while the percentage of packets with payloads of 1500 bytes can be 2.5%, 12.5% and 22.5%. The percentage of packets with payloads of 520 bytes is set to the complement of the combination. For each traffic distribution, we computed the overload caused by the header size of MPLS. Fig. 2.10 shows these values.

It should be remarked that in the simulations carried out before, packet headers changed their size suddenly from 4 bytes to 8 bytes (and vice-versa) depending on the hop they took. This made the average header size vary from 5 to 7, as mentioned previously. However, we consider the case in which all LSPs are stacked at every point for a worst-case analysis. In other words, the worst-case analysis contemplates a fixed header size of 8 bytes at all hops for all LSPs, implying the maximum label space reduction possible using AMTs.

The typical traffic distribution of the traffic in the Internet can be considered as: 50%;37.5%;12.5% for payload sizes of 40, 520 and 1500 respectively, making an average payload size of 402.5. Compared to the worst-case, we conclude that the traffic overload is increased by 1%, if AMTs are used. In addition, even if the traffic distribution is changed to 60%;37.5%;2.50% (average payload size of 256.5), the traffic overload is increased by 1.56% only.

What Happens in the Core?

As shown in §1.2.5, the larger percentage of labels reduced by MP2P is located in the LSRs close to the egress nodes. We show in this subsection that AMTs ease this drawback.

The simulation results previously presented are split into 5 different groups, each group considers only the LSRs of a particular rank. As mentioned before, the rank of an LSR is the minimum number of hops to the closest edge LSR in the topology. For every group, its average percentage of label space reduction for MP2P and AMT are computed. Since the fifth group is composed of only one

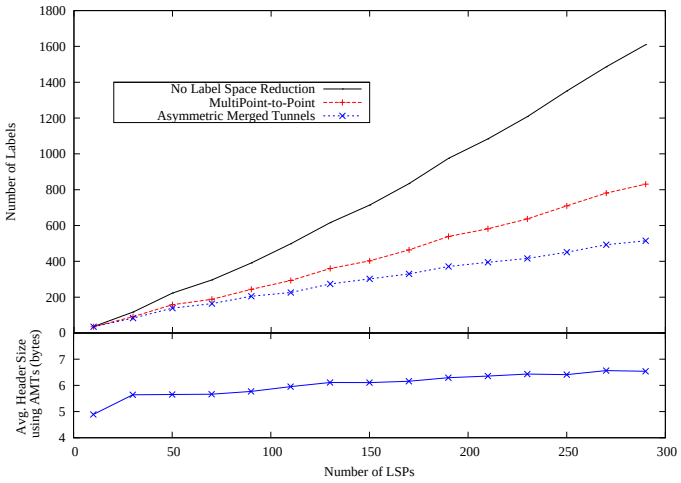


Figure 2.9: Number of Used Labels and Overhead caused by Stacking in Function of the Number of LSPs.

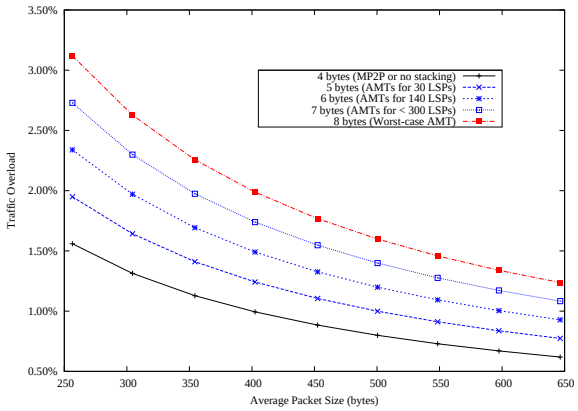


Figure 2.10: Overload caused in the network traffic because of MPLS headers.

node, only the first four groups are discussed and, hence, plotted in Fig. 2.11-Fig. 2.14.

Fig. 2.11 shows the label space reduction ratio for edge LSRs; as expected, the gain of using AMTs with respect to MP2P is low. The gain improves as both the number of LSPs augments and the rank increases, as seen in Fig. 2.12, Fig. 2.13 and Fig. 2.14.

To prove this observation, the correlation coefficient between the average gain obtained by using AMTs and the rank is computed. The coefficient was low positive (close to 0.3), which shows that the gain obtained by AMTs is not strongly related to the LSR ranks.

Sometimes some ranks may need to be sacrificed in order to obtain a better overall label space reduction using AMTs. For instance, between 110 and 190 LSPs, the solution obtaining the minimum label space requires that LSRs 18 and 23 (from rank three) create new tunnels that do not decrease their label space sizes. In fact, as it can be appreciated in Fig. 2.14, the reduction becomes negative (the number of labels for those LSRs is increased) for the AMT solutions in that range. However, it can be concluded that as the number of LSPs increases, the price that the latest ranks have to pay for an overall reduction is normalized.

This situation could be undesirable in scenarios in which the maximum number of labels used by any node in the network plays an important role, such as in AOLS. Even though the modifications in the ILPs to aim at this particular objective is small, its complete study goes beyond the purpose of this dissertation, and it is left for future research.

A Detailed Test

We performed a final test, this time with 500 LSPs routed in the same topology. In Fig. 2.15 we show the topology with the LSRs colored according to the label space reduction ratio in this test. Fig. 2.15 can be compared with Fig. 1.5 in order to appreciate the overall distribution of the label space reduction. The label space reduction of every LSR is shown in Table 2.1.

Let us denote by $R_{(i)}$ the set of LSRs with rank i.

Furthermore, let us split the set $R_{(0)}$ in two disjoint sets $R'_{(0)}$ containing those LSRs with degree one (i.e. LSRs $0, 11, 15, 16, 7$), and $R''_{(0)}$ containing the remaining ones (i.e. LSRs $17, 19, 22, 27$). We notice that the LSRs in $R'_{(0)}$ do not present any gain at all (0.00%) with the use of AMTs. This fact can be easily explained since all those LSRs have degree one. An LSR having degree one is never used as a transit LSR, i.e. all forwarded demands are either originated or ended there. The number of labels used for the demands *originated* in $R'_{(0)}$ cannot be decreased even though AMTs are used, since for every originated demand one different label is needed. Similarly, the number of labels used for the demands *ended* in $R'_{(0)}$ cannot be decreased using AMTs, since an AMT would always use the LSR's incoming links as the AMT receiving links. On the contrary, the LSRs in $R''_{(0)}$ experience a gain resulting from stacking of the demands that are traversing them.

Complementary, in order to corroborate our results, we carried out simulations over a set of 10 random network topologies generated accordingly to the work of Siganos *et al.* in [SFFF03]. Half of them containing 16 nodes and the

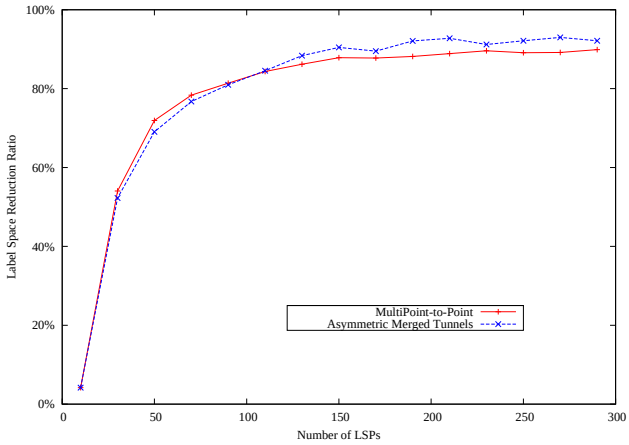


Figure 2.11: Label Space Reduction Ratio for MP2P and AMT at Rank 0

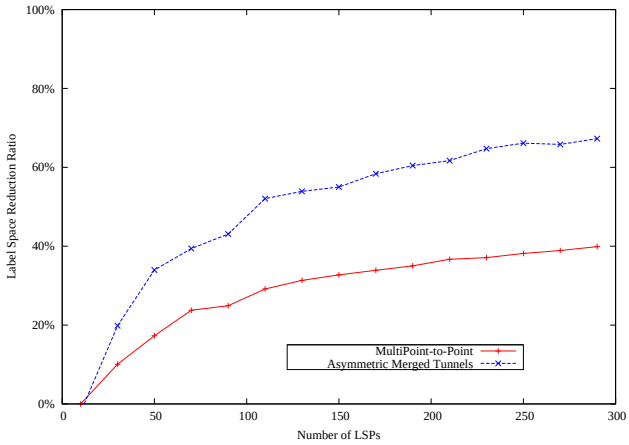


Figure 2.12: Label Space Reduction Ratio for MP2P and AMT at Rank 1

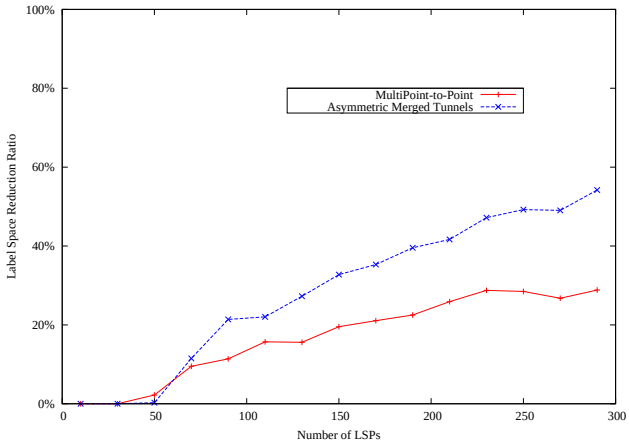


Figure 2.13: Label Space Reduction Ratio for MP2P and AMT at Rank 2

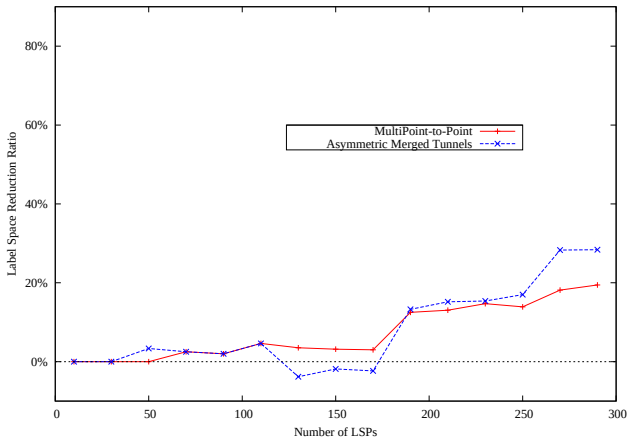


Figure 2.14: Label Space Reduction Ratio for MP2P and AMT at Rank 3

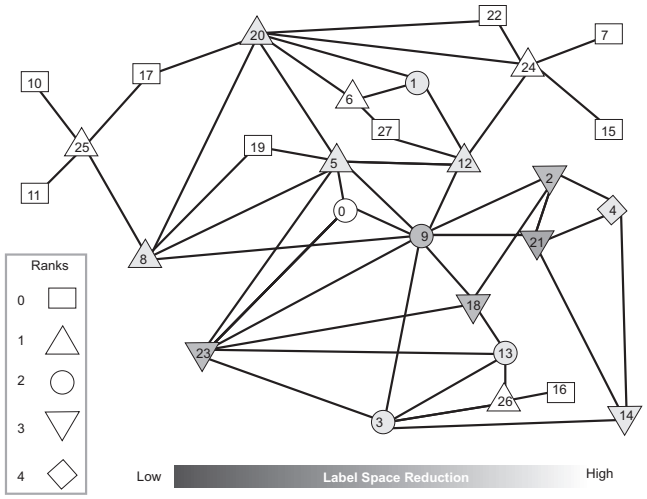


Figure 2.15: Simplified version of the Australian Rocketfuel ISP topology. Nodes ranked according to closest egress proximity and colored according to the percentage of labels saved using AMT.

Table 2.1: AMT vs MP2P Label Space Reduction in the Australian Topology loaded with 500 LSPs

Rank	**Node**	**Total**	**MP2P**	**AMT**	***Absolute Improvement***
	10	52	98.08%	98.08%	0.00%
	11	52	98.08%	98.08%	0.00%
	15	52	98.08%	98.08%	0.00%
	16	58	98.08%	98.08%	0.00%
0	17	139	79.86%	87.05%	7.19%
	19	100	73.00%	84.00%	11.00%
	22	120	78.33%	84.17%	5.83%
	27	99	77.77%	84.84%	7.07%
	7	52	98.08%	98.08%	0.00%
	12	247	43.32%	70.45%	27.12%
	20	372	47.31%	69.89%	22.58%
	24	284	44.37%	75.70%	31.33%
	25	233	48.08%	77.25%	29.18%
1	26	114	49.12%	85.96%	22.96%
	5	228	27.19%	52.19%	25.00%
	6	138	52.17%	76.09%	23.91%
	8	241	43.57%	73.44%	29.88%
	0	41	17.07%	75.61%	58.53%
	1	135	51.85%	74.81%	22.96%
2	13	74	41.89%	70.27%	12.5%
	3	82	42.68%	70.73%	28.05%
	9	147	12.93%	25.85%	12.93%
	14	22	50.00%	54.54%	4.54%
	18	44	13.63%	43.18%	29.54%
3	2	24	25.00%	37.50%	12.50%
	21	26	26.92%	34.62%	7.69%
	23	78	17.95%	28.21%	10.25%
4	4	18	38.89%	55.56%	16.67%
Total		3272	53.33%	70.67%	17.34%

other half 24 nodes. The network load was varied from 2 LSPs to 128 LSPs. All these simulations showed similar results, hence they are omitted from this dissertation.

2.4 From MPLS to GMPLS

As different switching layer flavors are deployed, a common and flexible control plane is desired. GMPLS is a generalized version of MPLS that aims at decoupling completely the forwarding and control planes giving support to different technologies.

To be able to match with several switching technologies, GMPLS standard had to classify the forwarding plane technology as either:

- Packet Switch Capable (PSC) interfaces, such as IP
- Layer-2 Switch Capable (L2SC) interfaces, such as Frame Relay, Ethernet and ATM
- Time-Division Multiplexing (TDM) interfaces, such as SONET/SDH
- Lambda Switch Capable (LSC) interfaces, based on PXC at a fine granularity, such as WDM
- Fiber Switch Capable (FSC) interfaces, based on PXC but at a coarser granularity

The establishment of LSPs that span only Packet Switch Capable (PSC) or Layer-2 Switch Capable (L2SC) interfaces is defined for the original MPLS control planes. As discussed along this part, the flow tagging is performed by imposing a header containing a label, a *logical label* with local meaning.

GMPLS extends these control planes to support each of the five classes of interfaces (i.e., layers) defined previously [Man04]. Since GMPLS is a control plane for non-packet networks as well (i.e. Time-Division Multiplexing (TDM), Lambda Switch Capable (LSC) and Fiber Switch Capable (FSC)), clearly, the support of these is done through *physical labels*, instead of logical labels (as usually MPLS does for PSC and L2SC). This is, information in these types of networks are not labeled. For instance, a wavelength color is used as a physical label to identify an LSP in a LSC network, or a time-slot is used as a physical label to identify an LSP in a TDM network.

Therefore, label stacking *per se* (in the way it has been described before) is not completely supported by all forwarding planes. However, the concept of stacking in GMPLS means, sometimes, to build a hierarchy of technologies. For example, at the top of a hierarchy there are FSC interfaces, followed by LSC interfaces beneath it, followed by TDM interfaces, followed by L2SC, and finally by PSC interfaces at the bottom.

Matching the contributions presented in this chapter with GMPLS can be regarded (in some cases) as deciding how to "bind" the physical/logical labels in the layer above. For instance, considering an IP-over-WDM architecture - comprising PSC interfaces at the IP (bottom) layer and LSC interfaces and the WDM (top) layer - the proposals would aim at reducing the number of physical

labels in the LSC interfaces (or lightpaths) when the path of the PSC demands are given[2].

Label merging depends upon the forwarding plane capabilities as well. For instance, LSC cannot perform label merging *per se*, since it would imply to merge different wavelength colors into one. However, with the aid of OPS label merging can be considered as feasible (which is the focus of the present document).

As a conclusion, the contributions presented in this document apply to GMPLS , under some restrictions depending on the type of technologies used.

2.5 Chapter Remarks

In this chapter the Asymmetric Tunneling concept for is presented in detail. The detailed discussion is based on proposing a set of tunneling facts for MPLS. Moreover, two algorithms - the Longest Segment First (LSF) and the Most Congested Space First (MCSF) algorithms - were proposed aiming to find the best way to reduce the label space. Analyzing the algorithm, simulations shows better performance and reduction of MCSF respect to LSF. On the other hand, simulation results also show great improvements in the number of labels been dropped off specially respect to traditional label merging schemes (MP2P).

Furthermore, the MP2P reduction is enhanced by means of a single stacked label, thus creating a tree-tunnel shape reduction. This type of reduction is denoted as *Asymmetric Merging Tunnels* (AMT for short). Initially a single optimization model for solving the LASPARED problem using AMTs is proposed, named the *BF-model* and discussed. Due to its cost, a second solution for the problem is discussed as well: *Decompose and Match Framework* (D&M),

Concerning simulation results, ATs shows a better performance that MP2P. For instance, while MP2P can reduce the number of labels used in 25% approximately, ATs may reduce them in 37% approximately.

Simulation results are presented for AMTs as well. Since AMTs is a method that takes advantage of both ATs and MP2Ps, its performance is better than any of them: almost 50% of them can be reduced using AMTs.

[2]Other constraints must be taken into account (e.g. wavelengths per fiber, capacity of a wavelength, wavelength conversion, etc.) depending on the forwarding plane capabilities.

Part II

Label Space Reduction in AOLS Networks

Chapter 3

All-Optical Label Swapping and Stacking in AOPS

This chapter is devoted to analyze the AOLS technology developed in the LASAGNE project. An overview of the contributions in the LASAGNE project is made in the first section of this chapter. In the second section, a modified AOLS architecture is proposed that enables label stacking, hence allowing to map all the solutions presented in the previous part to AOLS.

3.1 The LASAGNE Project

The LASAGNE project was originally conceived for OBS networks; however its success became for its adaptation to OPS networks, which clearly need more label processing that the former technology.

The LASAGNE project has as purpose the design of a device that is in charge of all-optical label swapping, named an AOLS-block (see Fig. 3.1). An AOLS-block is needed for each wavelength port in the OPS node that needs packet routing. An AOLS-block is in charge of reading the incoming label of a packet, replace it by the proper outgoing label (label swapping) and convert the packet to its respective outgoing wavelength color. It is worth noting that all these operations are performed optically in the LASAGNE project.

Optical Correlator Block. When a packet enters the AOLS-module, its payload is separated at 40Gbs [BPY+02]. What remains of the packet, i.e. the optical label, is fed to the *optical correlator* block. Within the *optical correlator* block, the incoming label is replicated several times, so every AOLXG [MRM+02] has a duplicate of the incoming label. An AOLXG is an optical device, implemented with a Semiconductor Optical Amplifier-based Mach-Zender Interferometer (SOA-MZI) that compares two labels and emits a high-intensity light pulse upon matching.

Local Address Generation Block. In order to perform the matching of the incoming label with the different recognizable labels of the switch, all these recognizably local labels are generated in parallel by the *local address generation*

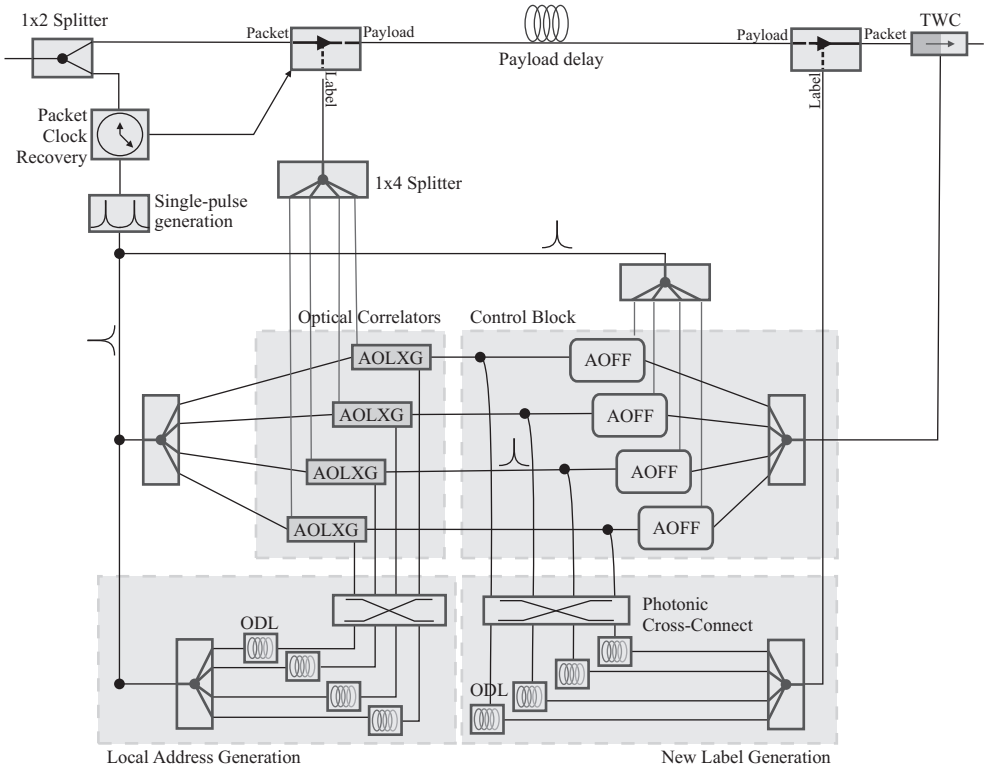


Figure 3.1: AOLS block able for label swapping

block - at the same time the incoming label is replicated. Each of these local labels is given to a different AOLXG. As a consequence, each AOLXG matches, in parallel, the same incoming label of the packet with a different locally "stored" label. Upon matching, the proper correlator transmits a high intensity light pulse. So far, the optical label has been identified. The high-intensity light pulse is sent to both the *new label generation* block and to the *control* block.

New Label Generation Block. The *new label generation* block produces the corresponding output label of the packet. This new label is inserted behind the payload.

Within the LASAGNE AOLS-block architecture, two label-generation blocks are needed: the *local address generation* block and the *new label generation* block. As mentioned before, the first one is in charge of generating the set of recognizable incoming labels, which feed later the correlators. The second one is in charge of generating the set of outgoing labels. To generate a label (either incoming or outgoing), an ODLs is used. An ODL is a very simple device comprised of several FDLs, couplers and splitters.[1]

Control Block. The *control* block drives the wavelength converter, so the output packet (including the new label) can be tuned to its proper outgoing wavelength frequency. As a consequence, both the payload and the header are wavelength converted. The *control* block is implemented with AOFFs. Depending on the matching address (indicated by one of the pulses coming from the correlators), the appropriate flip-flop will emit a Continous Wave (CW) signal at a certain wavelength. The frequency emitted by each AOFF is fixed. After the new label has been inserted, the packet is converted to the wavelength generated by the flip-flop.

Finally, the packet is routed by means of an AWG. Therefore, the wavelength on which the packet leaves the AOLS-block determines the outgoing port of the node.

Two PXC are used inside the AOLS-block to provide label swapping (inside the *new label generation* block) and wavelength conversion (inside the *local address generation* block) flexibility. The first PXC - in the *new label generation* block - switch the matching pulse from any AOLXG to any ODL. This offers a switching matrix between incoming (matching pulses) and outgoing labels (ODLs). The second PXC - in the *local address generation* block - switch the generated addresses between AOLXG correlators. This switching affects (by selecting) the fiber line through which the matching pulse is propagated. As a consequence, since each AOFF generates a CW fixed wavelength, the PXC selects the CW frequency generated by the AOFF fed by the matching pulse. These two PXC are part of the network control plane and are low-speed dynamically configurable.

[1] Upon the arrival of one pulse, the pulse is split in the number of ones that are needed for the generated label. For instance, to generate the number five (101 in binary), the pulse is split in two pulses (one for each one in the binary string). Then, different delays (using FDLs) are given to each one of the split pulses, so the third pulse has a greater delay than the second one, and so on. Finally, all the split pulse are coupled again in one light stream.

3.1.1 Label Spaces and Contention Resolution

In AOLS label spaces can be defined using one of the following policies.

- *Global.* There is a single space of labels for all the traffic in the switch. Forwarding decisions are taken solely using this label. Since there is only one label space per switch in this case, the number of labels needed is bigger than with any of the other options.

- *Per Fiber.* Separate label spaces are given for each fiber. Forwarding decisions are taken using the incoming fiber port and the label.

- *Per Wavelength Color.* Separate label spaces are given for each wavelength color. Forwarding decisions are taken using the incoming fiber port and the label.

- *Per Wavelength.* Different label space are used per each wavelength, i.e. wavelength color + fiber port. Therefore, forwarding decisions are taken considering the tuple wavelength color, incoming fiber port and label. It provides the smaller label spaces.

Although a *per wavelength* label space lead to the least number of AOLS blocks, it decreases the most the OPS performance when contention resolution is needed. Packet contentions are usually present at the output of the switch. Contention occurs in an OPS when two two packets are competing for the same output wavelength in the same output fibre port at the same time. Since all wavelengths belonging to a fiber lead to the same node, the most used method for contention resolution in OPS is to use a different wavelength in the same fiber to forward the packet in mention. However, in this case, a *per wavelength* label space would lead to an incorrect forwarding since packets may arrive on any wavelength (as a result of a previous contention resolution). The same would happen if a *per wavelength color* label space is used. In this contention resolution scheme is more appropriate to use a *global* or *per fiber* label space.

The use of wavelengths for contention resolution implies the use of several wavelength tunable converters. To avoid the use of wavelength tunable converters, a different approach is often considered for contention resolution. When two packets are competing for the same wavelength in the same fiber, instead of dropping one of the packet, the packet is sent using another fiber port; a principle known as *deflecting routing* or *hot-potato routing*. Neighboring nodes must be capable of handling the packet in question in order to reach its original destination. In this contention resolution scheme a *per wavelength color* or ever a *per wavelength* label spaces can be used.

Since an analysis of different contention resolution schemes in OPS is out of the scope of this dissertation, henceforth, as a worse case scenario, label spaces on a *global* basis are the subject of this dissertation.

It should be pointed out that the number of AOLS blocks is independent of the policy to define the label space. While the number of labels depends on how many demands are switched in the OPS, the number of AOLS blocks depends on how many wavelengths and fiber ports are used by the OPS. However, both of them define the cost and the flexibility of the architecture.

3.1.2 Label Stripping

Label stripping can be seen as a special case of label stacking, discussed previously. In label stripping, packets header are comprised of a stack full of labels. At each hop, the node strips off one label, i.e. pop the stack, and process it. The content of the stripped label is used for forwarding, as usual, but labels are never swapped.

Since each node pops the stack once, there is one different label in the stack per each hop. In order to improve resource utilization, a label is used to indicate the link that must be taken to forward the packet. Therefore, each node must handle a minimum number of labels equal to the number of neighbors it has. The number of AOLXG is drastically reduced.

On the other hand, since the stack size should be equal to the number of hops a packet would traverse, the size of the stack is augmented. This would incur in a waste of bandwidth, since the allocated space for the stack cannot be recovered even though the stack contains only one optical label.

3.1.3 Performance Overview

In this subsection a subset of the results presented by Van Caenegem *et al.* in [CCPD06] are presented. The simulations were performed using the European Network shown in Fig. 3.3 and the network demands of [MCL+02].

Initially, the dimension of the AOLS-block is studied considering the different types of labels spaces previously mentioned, these are: *global*, *per wavelength*, *per wavelength color* and *per fiber*. Sizing a AOLS-block implies counting:

- the number of PXC ports used in the *new label generation* module,
- the number of PXC ports used in the *local generation address* module,
- the length (distance) of all the incoming ODL,
- the length (distance) of all the outgoing ODL,
- the number of AOLXG correlators used,
- the number of outgoing ODL,
- the number of bits used to encode labels

The results are grouped together differently in Fig. 3.4 and Fig. 3.5, with the purpose of depicting different trade-offs.

The Cost of Contention Resolution using Wavelengths

Fig. 3.4 shows the reduction percentage of the number of used optical components when the label spaces are unable to support contention resolution.

For instance, the first bar (darker color) in Fig. 3.4 represents the reduction ratio of the optical components when the *per wavelength color* label space is used and the *global* label space, which is its contention-resolution counterpart. In the same figure, the second bar (lighter color) resumes the optical components reduction ratio when a *per wavelength* label space and, its counterpart, *per fiber* label space are used.

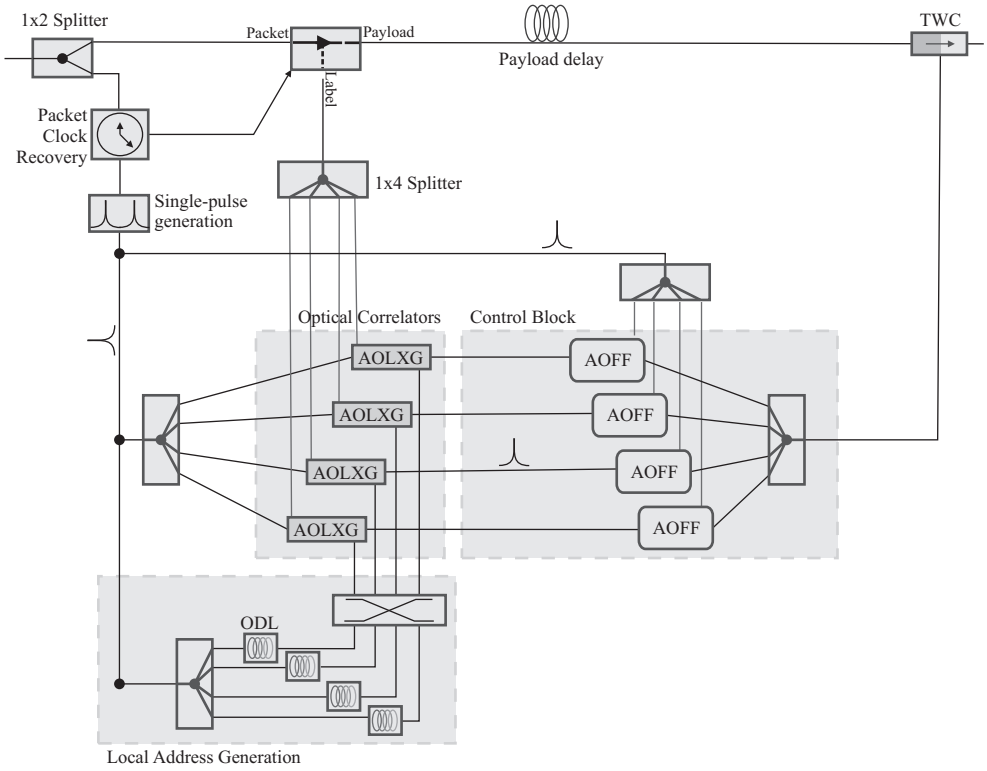


Figure 3.2: New Label Generation block allowing label stripping

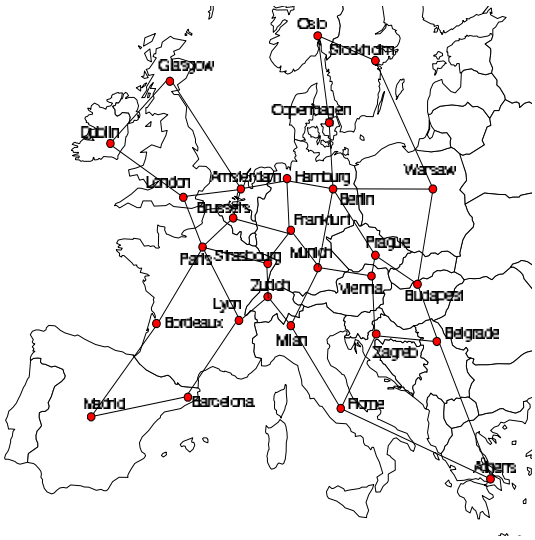


Figure 3.3: European Network

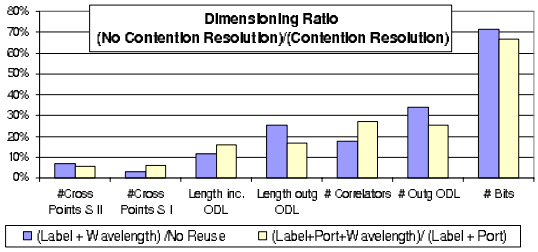


Figure 3.4: Dimensioning of label spaces considering Contention vs. No contention resolution ([CCPD06, Fig. 12])

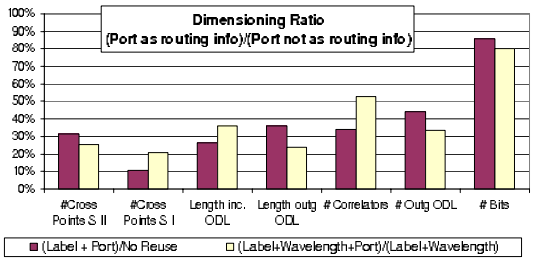


Figure 3.5: Dimensioning ratio II ([CCPD06, Fig. 13])

Adopting a *Per Fiber* Label Space

To measure the impact of the incoming port carrying routing information, the results of the label spaces types where the incoming fiber port is used as routing information (i.e. *per fiber* and *per wavelength*) is considered, and then divided by the results of similar label space types where it is not (*global* and *per wavelength color*).

It is expected that the dimensions would be smaller in case that incoming fiber port is used as routing information, as it can be seen in Fig. 3.5.

Stripping vs. Swapping

As mentioned before, although labels using stripping are short, the 'total' labels to be transported (end-to-end label) could be longer than any when labels are swapped instead. This is because in the former the end-to-end label is a concatenation of local labels. This subsection is devoted to analyze this aspect.

First of all, let us assume that an optical header contains: a label-field (comprises the forwarding information), guard bands (to separate the different information fields), and CoS-field (Class of Service: to indicate the traffic priority). When labels are stripped off, the header may contain a single CoS-field for the whole header or one CoS-field for each label. Guard bands must be used between labels (in case of label stripping), between a label and a CoS-field, and to delimit the header.

Assuming that the length of a label is 8-bits (256 flows handled by an OPS) long for label swapping and 3-bits (a connectivity degree of eight in the network) long for label stripping, that both the guard bands and the CoS-field have a

length of 3-bits each, and that the maximum number of hops in the network is eight; the total number of bits used in a header is shown in the Table 3.1.

Table 3.2 gives an overview of the label overhead introduced by the different switching strategies.

Assume that 50% of the payload sizes are 40-bytes long, 37.5% are 520-bytes long, and 12.5% are 1500-bytes long. The overhead caused in the network is then 3.14%, 8.13% and 12.79%. In this case, the overhead introduced by the label stripping strategy worst case is only four times bigger than that of the label swapping strategy.

Complementary results are going to be given at the last chapter of this dissertation.

3.2 Implementing Label Stacking in AOLS*

Enabling an AOLS block with stacking and swapping implies that the system should be able to generate one or two new labels maximum. In the case of stacking, it should be able to insert the two new labels before the packet. In the case of swapping, the system must behave as usual. Since deciding how many labels are going to be inserted depends on the content of the incoming label, to provide this flexibility is not straightforward.

The easiest way to provide two or one label insertion in the system is by allocating a fixed space for two labels even though only one is placed. The *Packet/Payload separation* circuit must be able to extract only the top label of the packet, regardless of how many they are. The remaining label, if any, should be treated as part of the payload.

Generating one or two labels can be implemented with a slight modification of the *new label generation block*, as seen in Fig. 3.6 [SCC+07b].

The high-intensity pulse of the AOLXG that matched the incoming label is split in two. One of the two pulses is delayed for a fraction of time equal to the duration of one label. Both pulses are switched - one after the other due to the induced delay - using the PXC in order to generate two different labels. Both labels are generated out of each pulse using the same set of ODLs mentioned before. Finally, the labels are merged into one optical stream.

In the case only one label is needed, the PXC should drop the pulse that was not delayed. In the event of popping the stack, the PXC should drop both pulses.

Penultimate Hop Popping in AOLS As mentioned before, GMPLS allows non-labeled packets forwarding at the last hop. For this, the label stack has to be popped (hence becoming empty) in the previous hop to the last LSR. This special case is known as the PHP. When the last LSR in the path receives an

	Guard Bands	CoS-field	Label-field	Total
Swapping	3×3 bits	1×3 bits	1×8 bits	20 bits
Stripping (CoS per header)	10×3 bits	1×3 bits	8×3 bits	57 bits
Stripping (CoS per label)	17×3 bits	8×3 bits	8×3 bits	99 bits

Table 3.1: AOLS Header Sizes

Strategy	Swapping	Stripping	
		CoS per header	CoS per label
Payload\Header size (bytes)	2.5	7.125	12.375
40	5.88%	15.12%	23.63%
520	0.48%	1.35%	2.32%
1500	0.17%	0.47%	0.82%

Table 3.2: Network Overload due to AOLS Headers

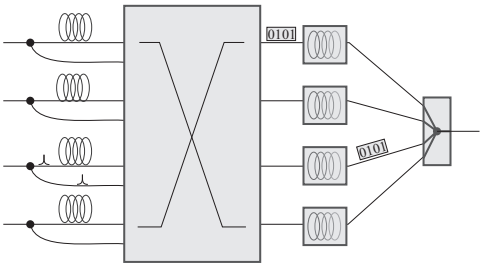


Figure 3.6: New Label Generation block allowing label stacking

unlabeled packet, the packet is treated as an IP packet (or the corresponding routing layer) in order to determine its destination.

However, because of routing in the AOLS architecture is based solely in the coded label, AOLS packets must always carry a label; including the last hop. This slight difference may increase the number of labels used in an AOLS-driven network with respect to a traditional GMPLS network.

On the other hand, if label stacking is considered, the PHP option can be used in the tunnel. Indeed, not considering the option would increase the label space, as mentioned before.

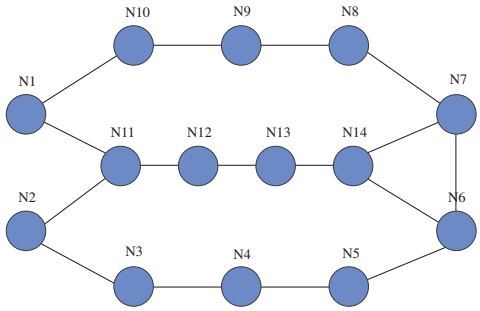
3.3 Label Space Size vs. MLU: An Example

In this subsection we present an example showing how the MLU affects the label space, and how this repercussion is eased when label merging and stacking are considered.

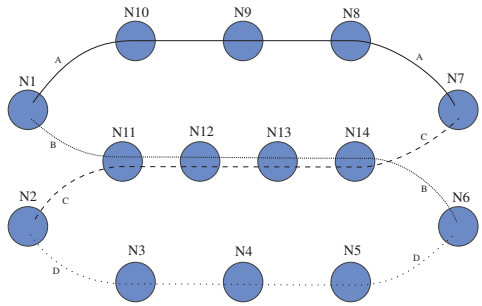
Let us consider the physical fiber topology of Fig. 3.7(a) with 14 nodes, in which nodes N1 and N2 are ingress and nodes N6 and N7 are egress. Let us assume that we need to route one unit of traffic from both ingresses to both egresses. More precisely, we denote by A the connection needed from N1 to N7; B the connection from N1 to N6; C the connection from N2 to N7 and; D the connection from N2 to N6. The solutions shown in this subsection do not contemplate the possibility of splitting the demanded bandwidth across several paths.

A classical Traffic Engineering (TE) routing algorithm could aim at routing traffic such that the MLU is minimized while bounding the delay[2]. A typical TE solution for this example can be seen in Fig. 3.7(b). The solution has the minimum delay (or hop count) and the minimum MLU. In this case, the number

[2]For simplification, we assume that hop count is equal to the delay



(a) Fiber Topology.



(b) 22 labels and MLU of two units of traffic along three links.

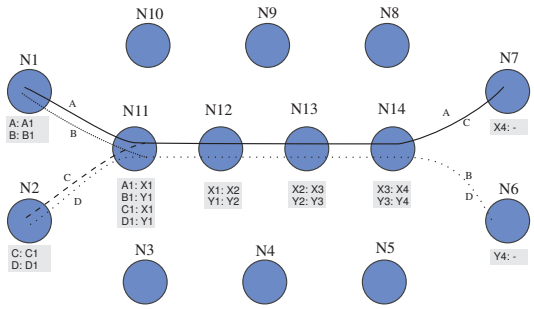
Figure 3.7: Routing and Traffic Engineering in AOLS.

of used labels equals the hop count, i.e. 22, and the MLU does not exceed two units of traffic along three links (N11-N12, N12-N13 and N13-N14).

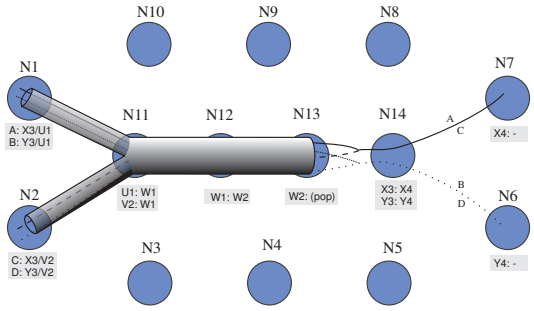
Considering the *label merging* feature, a different routing solution leads to the usage of fewer labels. For instance, in Fig. 3.8(a) connection A is using a different route, so its labels can be merged with connection C from node N11 to N7. Similarly, connection B is merged with connection D. Even though the number of labels is reduced in this case down to 16, the MLU has been increased up to four units of traffic along the same three links.

Label stacking, together with label merging, gives much more options to play with. In Fig. 3.8(b) another solution is given to the problem reducing the label space to its minimum. In this case, 12 labels are needed while the maximum link utilization is preserved to four units of traffic along the same segment.

In case the MLU is limited to (or links capacities are fixed to) two units of traffic, the solutions in Fig. 3.8(a) and Fig. 3.8(b) are not feasible. However, the solution in Fig. 3.7(b) can use 19 labels if one tunnel is created, as seen in Fig. 3.9(a). In addition, in Fig. 3.9(b), a different routing solution that reduces the label space down to 16 can be seen as well. While the solution in Fig. 3.9(a) preserves the TE routing (achieving the best link utilization and delay), the solution in Fig. 3.9(b) increases the number of links reaching the MLU to eight but reduces even more the label space.

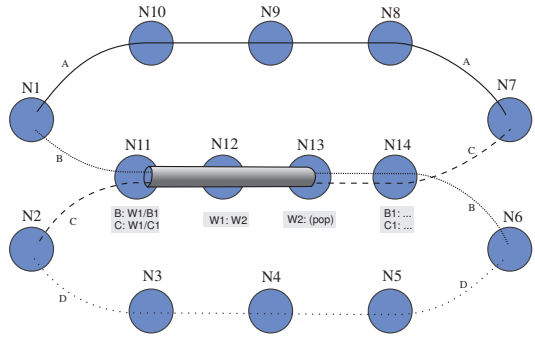


(a) 16 labels and MLU = 4u at three links (no stacking).

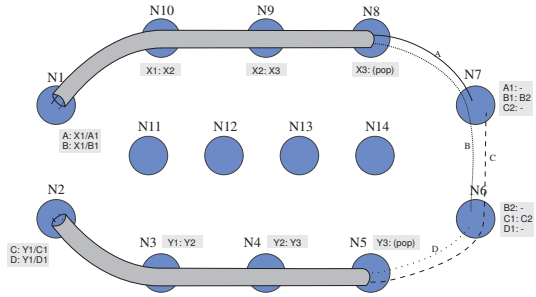


(b) 12 labels and MLU = 4u at three links.

Figure 3.8: Example of Link Utilization vs. Label Space Size.



(a) 19 labels and MLU = 2u at three links.



(b) 16 labels and MLU = 2u at eight links.

Figure 3.9: Example of Label Space Size when MLU is bounded.

3.4 Modeling Routing & Label Space Reduction in AOLS*

Without considering any label space reduction method (neither merging nor stacking), the total number of labels used in a network is equal to the total number of hops needed to route flows. Therefore, the longer the routes are, the more overall labels are needed.

In the previous part tackling the label space reduction problem in GMPLS flow routing was resolved by an algorithm considering several QoS metrics. As a matter of fact, routing was performed *before* - and regardless of how - labels were assigned. As a counterpart, the routing algorithm aimed at finding relatively short paths. Routing and labels binding were performed separately since it was not worth to sacrifice the QoS level for smaller label space.

In OPS using AOLS, the context is different. A slightly longer route could result in a imperceptible lower QoS, but in great CAPital EXpenditures (CAPEX) savings if label space reduction methods are considered. For instance, considering merging, employing two long "mergeable" routes may represent more CAPEX savings than employing two "non-mergeable" short ones. The overall number of used labels of the MP2P connection, in the former solution, could be less than the two P2P connections, in the later solution. Therefore, the routing problem is henceforth considered together with the label space reduction problem.

As discussed in [CCPD06], the dimensions of the AOLS-block are intrinsically related to a) the number of incoming labels, b) the number of outgoing labels, and c) the number of bits used to code optical labels in the network. These parameters are considered for analysis in this section through mathematical modeling of the problem.

At each of the following subsections, the routing problem is considered together with one of the different methods presented before in GMPLS. The complexity of the label space reduction methods used at each subsection is greater than the previous.

3.4.1 AOLS Routing

Initially, a simple routing model is considered without using any label space reduction method. As mentioned before, reducing label spaces in this context can be translated to routing using the shortest path: a CSPF algorithm. Although these algorithms have been extensively studied before, an ILP model is proposed with the purpose of extending it in further subsections to include label space reduction methods.

The ILP models considered throughout this section are path-based. Therefore, all feasible paths in the network are initially generated using a simple algorithm.

The following is a list of all the indexes used in the model, so far.

- $e, i, j \in \mathcal{N}$ Optical nodes in the network
- $l \in \mathcal{L}$ A generated path in the network. Sometimes l_M and l_O are used as well.
- $m \in \mathcal{M}$ A flow demand.

The parameters used in the model are:

- $R^l_{(i,j)}$ Set to 1 if link (i,j) is used by the generated path l
- S^l_j Set to 1 if node j is the source of the generated path l
- D^l_j Set to 1 if node j is the destination of the generated path l
- C^m Set to the bandwidth demand of m
- Δ^m_j Set to 1 if node j is the source of demand m, -1 if it is its destination, and 0 otherwise.
- F Set to the number of wavelengths per fiber
- W Set to the capacity of each wavelength
- L The maximum number of bits used to encode labels.

The solely variable used by this model is $\alpha^{l_M,m}$, set to 1 when demand m is routed using path l_M.

The objective function is to reduce the number of labels (or hops in this context) of every routed demand.

$$\min \sum_{l,m} R^l \cdot \alpha^{l,m}, \quad \text{where} \tag{3.1}$$

R^l is the number of hops used by route l, i.e. $\sum_{(i,j)} R^l_{(i,j)}$

Subject to:

$$\sum_l \alpha^{l,m} = 1, \qquad \forall m \tag{3.2}$$

$$\alpha^{l,m} \leq 0, \qquad \forall l,m \text{ s.t. } \sum_j \Delta^m_j \cdot (S^l_j - D^l_j) \leq 1 \tag{3.3}$$

$$\sum_{l,m|R^l_{(i,j)}=1} C^m \cdot \alpha^{l,m} \leq F \times W, \qquad \forall i,j \tag{3.4}$$

The first three equations assure that the path used to route a flow demand is consistent with the source and destination nodes of the demand. The last equation limits the capacity that can be used in every fiber link.

The labels length is bounded with the expressions.

$$\sum_{i,l,m|R^l_{(i,j)}=1} \alpha^{l,m} \leq 2^L, \qquad \forall j \tag{3.5}$$

$$\sum_{i,l,m|R^l_{(j,i)}=1} \alpha^{l,m} \leq 2^L, \qquad \forall j \tag{3.6}$$

3.4.2 Aggregating AOLS flows

The number of labels can be drastically reduced if it is noticed that several demands between a pair of nodes are routed, by the previous model, through the same paths. Therefore, it is normal to consider that all these demands following the same end-to-end paths can be labeled equally. This is, assign one label per each different used path in the network regardless of how many demands are using it. It can be assumed that differentiating the flows is performed outside the optical domain, once the signal is given to the underlying electronic switch or local area network, for instance.

To model this, a new variable is introduced: δ^l, set to 1 if path l is used to route any demand.

The objective function is:

$$\min \sum_{(i,j),l} R^l_{(i,j)} \cdot \delta^l \tag{3.7}$$

subject to all the constraints previously presented plus an additional one linking the two variables $\alpha^{l,m}$ and δ^l:

$$\alpha^{l,m} - \delta^l \leq 0, \qquad \forall l, m \tag{3.8}$$

Since labels are assigned to paths instead than to demands, the expression bounding the labels length shall be modified.

$$\sum_{i,l|R^l_{(i,j)}=1} \delta^l \leq 2^L, \qquad \forall j \tag{3.9}$$

$$\sum_{i,l|R^l_{(j,i)}=1} \delta^l \leq 2^L, \qquad \forall j \tag{3.10}$$

3.4.3 Label Merging in AOLS

Label merging in MPLS was deeply analyzed in [SFM07]. A polynomial-time algorithm solves the label bindings if the routes are given. However, since the demand routes are considered as not fixed in AOLS, an ILP is needed to perform labels merging together with routing [SCC+07a].

In the last objective function the decision variable (δ^l) was multiplied by a constant factor ($\sum_{(i,j)} R^l_{(i,j)}$) in order to calculate the number of used labels per each employed path l. This is because the number of labels used by a path is fixed. In order to perform label merging in this model, it is considered that not all links of a path use labels. When a link of a path is not using a label it is because another "mergeable" path can share its label with the former path.

In order to decide which paths would use a label in a link, a new variable is needed in the model: $r^l_{(i,j)}$. The variable is set to 1 when a label is employed at link (i, j) for path l. As mentioned in previous subsection, it should be recalled that labels are assigned to paths instead than to demands.

The objective function is therefore:

$$\min \sum_{(i,j),l} r^l_{(i,j)} \tag{3.11}$$

It is clear that the variable takes the value of zero (0) in all its links when the path is not used. However, such constraint can be reformulated in order to restrict more the search space as follows.

If the paths that are going to be used could be known in advance, determining how labels can be merged for those paths is a deterministic polynomial-time decision [SFM07]. Given a set of paths with the same egress, the same label can be allocated in a particular link to a set of paths if the paths follow exactly the same route from the link to the egress LSR. It should be noted that given a particular link and a set of paths with the same egress LSR, there could be several groups of paths that cannot use the same label. For instance, in the previous example shown in Fig. 1.3, the link $N11 \rightarrow N10$ has two groups of paths that cannot be merged even though all the paths have the same egress node. The first group is conformed by paths B and C, and the second by paths D and E.

Since the paths that are going to be used to route the demands are not known in advance, the ILP must know how to merge any of the feasible paths to conform a MP2P connection in the network. For this, a new parameter is introduced in the ILP: $M^{e,l}_{(i,j)}$. Given a link (i,j) and an optical node e (identifying a MP2P connection in the network), the value of $M^{e,l}_{(i,j)}$ is the same for all the paths l that can use the same label in the link (i,j). In other words, the parameter $M^{e,l}_{(i,j)}$ groups (with its value) the paths l that can be merged in a particular link (i,j) for a MP2P rooted at e.

Computing this parameter is performed in polynomial time using an algorithm similar to the Full Label Merging algorithm [SFM07], if all possible paths in the network are considered. Given a link (i,j) and a root e, the values of $M^{e,l}_{(i,j)}$ are assigned consecutively to each set of mergeable paths l. For notation simplicity, let us assume that $\hat{M}^{e}_{(i,j)}$ is the maximum value (i.e. group number) in a particular link of a MP2P connection.

Considering a link (i,j) of a MP2P connection with root e, the number of labels used should be (at least) one for all the paths belonging to the same group. This is expressed with the following constraint.

$$\sum_{l|M^{e,l}_{(i,j)}=k} \left(\delta^l - \|\mathcal{L}\| \cdot r^l_{(i,j)}\right) \leq 0, \qquad \forall e,i,j,k \in \mathbb{N}, 1 \leq k \leq \hat{M}^{e}_{(i,j)} \tag{3.12}$$

In the previous constraint, for a given group k, if any of the paths δ^l is used, labels in that link $(r^l_{(i,j)})$ are needed. Note that the rate between all the used paths in group k $(\sum_{l|M^{e,l}_{(i,j)}=k} \delta^l)$ and the number of labels used in link (i,j) for group k $(\sum_{l|M^{e,l}_{(i,j)}=k} r^l_{(i,j)})$ is $\|\mathcal{L}\|$. Since

$$\|\mathcal{L}\| \cdot r^l_{(i,j)} \geq \sum_{l'} \delta^{l'},$$

for the set of paths of particular group in a link, an optimal solution contains only one label ($r^l_{(i,j)}$) in the group regardless of how many paths of that group are being used.

The number of incoming and outgoing labels is restricted with the expressions.

$$\sum_{i,l} r^l_{(i,j)} \leq 2^L, \qquad \forall j \tag{3.13}$$

$$\sum_{i,l} r^l_{(j,i)} \leq 2^L, \qquad \forall j \tag{3.14}$$

3.4.4 AMT in AOLS

In this subsection a model considering AMT with full label merging and routing is proposed.

In this model, there are two types of paths that need to be differentiated. While the first type is the same used so far to route the demands, the second ones is the path used by a branch of an AMT. For this purpose, the paths used to route demands are denoted with the index l_M and the paths used by a branch of an AMT are denoted with the index l_T.

In order to handle the AMTs, two new variables are needed: γ^{l_T} and β^{l_M,l_T}. Variable γ^{l_T} is set to 1 when an AMT uses the path l_T as a branch to perform stacking on a set of demands path. Variable β^{l_M,l_T} is set to 1 when the branch following path l_T of an AMT performs stacking over demands routed through the path l_M. In other words, variable γ^{l_T} "summarizes" variable β^{l_M,l_T}.

The number of labels that are saved by using the branch l_T with the path l_M is gathered by the parameter Δ^{l_M,l_T}.

Similarly to the previous model, the variable $r^{l_T}_{(i,j)}$ is used to determine which links are using labels in an AMT branch.

The objective function is formulated as follows.

$$\min \sum_{(i,j),l_M} R^{l_M}_{(i,j)} \cdot \delta^{l_M} - \sum_{l_M,l_T} \Delta^{l_M,l_T} \cdot \beta^{l_M,l_T} + \sum_{(i,j),l_T | D^{l_T}_j = 0} r^{l_T}_{(i,j)} \tag{3.15}$$

The expression adds the number of total labels without AMTs (as consequence of routing), substracts the number of labels saved because of AMTs and adds the number of labels used by using AMTs.

The MP2P constraint of previous subsection is replaced by a similar one concerning AMT.

$$\sum_{l_T | M^{e,l_T}_{(i,j)} = k} \left(\gamma^{l_T} - \|\mathcal{L}\| \cdot r^{l_T}_{(i,j)} \right) \leq 0, \qquad \forall e,i,j,k \in \mathbb{N}, 1 \leq k \leq \hat{M}^e_{(i,j)} \tag{3.16}$$

The number of AMTs stacking one path per link is constrained as.

$$\sum_{l_T | R^{l_T}_{(i,j)} = 1} \beta^{l_M,l_T} \leq 1, \qquad \forall i,j,l_M | R^{l_M}_{(i,j)} = 1 \tag{3.17}$$

The number of incoming and outgoing labels is bounded with the following two expressions.

$$\sum_{i,l_M|R^{l_M}_{(i,j)}=1}\left(\delta^{l_M}-\sum_{l_T|R^{l_T}_{(i,j)}=1}\beta^{l_M,l_T}\right)+\sum_{i,l_T|D^{l_T}_{j}=0}r^{l_T}_{(i,j)}\leq 2^L,\quad \forall j \tag{3.18}$$

$$\sum_{i,l_M|R^{l_M}_{(j,i)}=1}\left(\delta^{l_M}-\sum_{l_T|R^{l_T}_{(j,i)}=1}\beta^{l_M,l_T}\right)+\sum_{i,l_T|D^{l_T}_{j}=0}r^{l_T}_{(j,i)}\leq 2^L,\quad \forall j \tag{3.19}$$

The new variables are linked using the following set of constraints.

$$\beta^{l_M,l_T}-\delta^{l_M}\leq 0, \qquad \forall l_M,l_T \tag{3.20}$$

$$\beta^{l_M,l_T}-\gamma^{l_T}\leq 0, \qquad \forall l_M,l_T \tag{3.21}$$

3.5 Solving the Routing and Label Space Reduction Problem using Heuristics*

The problem is solved in two-steps by two separate algorithms proposed in this section. The first algorithm routes a traffic demand matrix aiming at setting up paths such that they share the maximum number of links. Once all demands (if possible) are routed, the second algorithm creates a set of tunnels reducing the label space.

3.5.1 The Path-Interfering Routing Algorithm (PIRA)

As routing solution, we propose a modification of the Constraint Shortest-Path algorithm (CSPF). The modification consists in how the weight of links is computed. Traditionally, the weight of a physical link is set to its (propagation) delay or proportional to its available bandwidth. In this section, we propose a new dynamic weight that aims at favoring links using more labels. Our intention is that, after many iterations of the routing algorithm, there are going to be segments in the network that are highly loaded with paths. These segments will later cause a high label space reduction when tunnels are placed to cover them.

Given a network $G^*(V,E^*)$ and a set of paths P, we extend G^* to a *directed multigraph* $G(V,E)$ as follows.

Every node and link in G^* are also considered in G. We name these links: *physical links*. The bandwidth of a physical link in G is set to the available bandwidth of its corresponding link in G^*. In addition, its weight is fixed to one.

Given a pair of non-adjacent nodes $i,j\in V$ and the set of paths P, it is worth noting that:

- There could be several paths that are forwarded through the same segment from i to j.

- Considering the set of paths P, there could be more than one segment that forwards information between i and j.

Regardless of which paths they belong to, we denote $s_{i\to j}^{(k)}$ the k-th different segment from i to j considering P. We create one different link from i to j in G for every $s_{i\to j}^{(k)}$. We name these links: *induced links*. The bandwidth of an induced link in G is set to the minimal available bandwidth of the links in G^* conforming its corresponding segment. In addition, its weight is set to

$$\frac{|s_{i\to j}^{(k)}| - 1}{|P(s_{i\to j}^{(k)})| + 1}, \quad \text{where}$$

$|s_{i\to j}^{(k)}|$ is the length of the segment and $|P(s_{i\to j}^{(k)})|$ is the number of paths in P that traverse the segment. The idea behind the weight is that a new path "pays", at every hop, only for the share of the label that it uses. One (1) is subtracted from the numerator because the last hop of $s_{i\to j}^{(k)}$ never incurs in labels reduction [SSFM07]. One (1) is added to the denominator in order to consider the new path as well.

When CSPF takes an induced link to route a demand, the link should be interpreted instead as the set of physical links in G^* representing it. If a cycle is created in the physical links, it is broken. It is worth noting that, even though the number of links contemplated increases, the complexity of the CSPF-procedure should not be worse than that run with a full mesh graph.

3.5.2 The Most-Profitable Tunnel First Algorithm (MPTF)

We consider that path routes were already computed. Taking into account these path routes, we compute the set of tunnels to consider as described by Theorem 1.

It is worth noting that the number of labels that ϕ can reduce by covering α is $|\alpha \cap \phi| - 1$, however ϕ needs $|\phi| - 1$ labels despite the number of paths it covers. This gives us an idea of the *profit* gained by setting up a tunnel. More concretely, in this section, the profit of a tunnel is the number of labels that it reduces by covering all the non-covered segments of the paths.

Note that our metric (i.e. the profit) considers the "savings" due both covered length and covered broadness, gathering in one metric the best of both LSF and MCSF.

At each iteration, a tunnel - name it ϕ - with the *best profit* is always selected. Once a tunnel is selected, all the non-covered segments of paths that ϕ can cover - name them $P(\phi)$ - are marked as covered. These covered segments will not be considered in further iterations. The profit of all remaining tunnels must be updated. The algorithm iterates until no tunnel remains.

At the beginning of the algorithm, since all paths are completely uncovered, the profit of a tunnel ϕ is set to:

$$\Pi(\phi) \leftarrow 1 - |\phi| + \sum_{\alpha \in P(\phi)} (|\alpha \cap \phi| - 1)$$

Before the next iteration takes place, the profit of all the others tunnels θ overlapping with ϕ is updated accordingly to:

$$\Pi(\theta) \leftarrow \Pi(\theta) + |\phi \cap \theta| \cdot \delta_{\phi,\theta} - \sum_{\alpha \in P(\phi)} (|\alpha \cap \theta| - 1), \quad \text{where}$$

$\delta_{\phi,\theta}$ is set to one (1) if tunnels ϕ and θ end at the same node, zero (0) otherwise.

Neither ϕ nor any other tunnel $\phi' \subset \phi$ are considered any more in further iterations. The complexity of the algorithm is bounded by $O(t \cdot \log t)$, where t is the number of feasible tunnels.

3.6 Simulation Experiments

In this section, we present a set of simulations that shows the discussed trade-off in this chapter.

3.6.1 Heuristic Performance

The European network consisting of 37 nodes is used in our simulation experiments, see Fig. 3.10. The link capacity is varied from 100 units to 3000 units of traffic, creating different simulation scenarios limiting the MLU. We use the term link capacity and MLU to denote the same throughout our analysis.

The first 30% of the nodes with lowest connectivity degree are selected as edge (ingress/egress) routers. For each pair of edge routers, a number of X, Y and Z demands of one, three and 12 units of traffic is randomly generated. X, Y and Z follow an uniform distribution with parameters [0-20], [0-12] and [0-4] respectively. As a result, we generate an average of 200 demands with average demanded bandwidth of 6864 units of traffic.

In this subsection we tested several combinations of routing heuristics with label space reduction heuristics. Namely, we considered as routing heuristics: CSPF and PIRA (proposed in this chapter). As for label space reduction heuristics, we considered: MultiPoint-to-Point [SMY00, AT03] (MP2P, i.e. label merging without stacking), LSF, MCSF and, MPTF (proposed in this chapter).

In total, eight routing-tunneling solutions are contemplated. In Fig. 3.11(a), simulation results show that the best label space reduction is achieved by PIRA-MPTF when the MLU limit is high, and CSPF-MPTF when the MLU is low. In general, PIRA obtains better reductions when used with high MLUs and with any stacking heuristic (i.e. LSF, MCSF and MPTF). In the same way, MPTF obtains the best reduction regardless of the routing heuristic.

The use of the stack reduces the label space four times (CSPF-MP2P vs. CSPF-MPTF) in average when the capacity of the links is just enough to route traffic (around 1000 units of traffic), and almost six times if the capacity is doubled (CSPF-MP2P vs. PIRA-MPTF).

It is worth noticing that the number of labels increases at the beginning, when the MLU limit is low. This behavior is explained by the fact that the routing heuristics (either CSPF or PIRA) employ more, and longer, paths to accommodate the traffic in these scenarios, in order to avoid the violation of the MLU limit. In case of PIRA, this behavior is stronger and particulary emphasized (see peaks in figure) when the MLU limit is set to 800 and 1400 units of traffic.

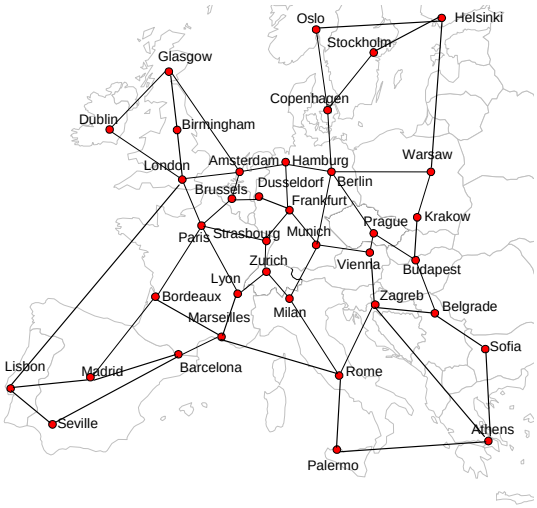
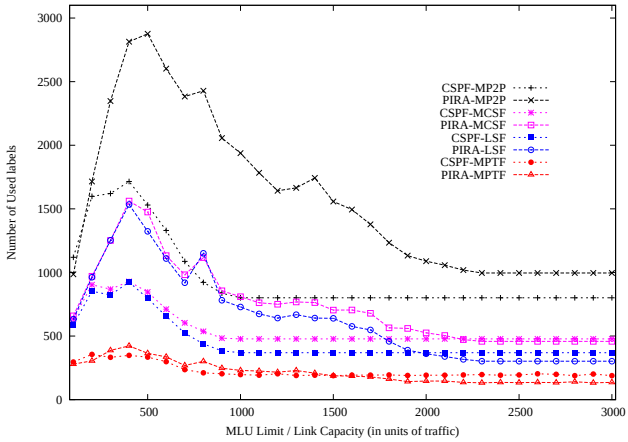


Figure 3.10: European network with 37 nodes.

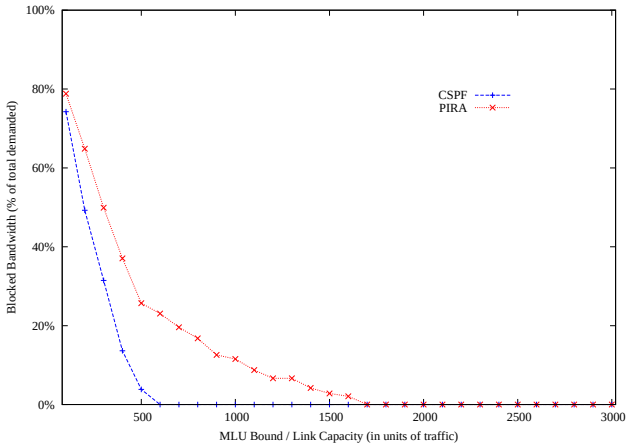
As expected, PIRA creates more link bottlenecks than CSPF in order to provide more tunneling possibilities, see Fig. 3.11(b). Therefore, the usage of PIRA is advisable when the minimum spare capacity of the links doubles the MLU. In our simulations, we noticed that the usage of PIRA in this circumstances profits with a 30% in the label space reduction.

We proceed to focus now in the particular routing solutions when the capacity of the links is high. While the MLU utilization of CSPF is 934 units of traffic, PIRA's is 2236; this is 2.5 times more. However, we notice that this case occurs in few links. In Fig. 3.12 we show the number of links in PIRA whose used capacity is above (or below) a given percentage of the MLU of CSPF (934 units of traffic). For instance, there are six links in PIRA that are using between 50% and 75% *more* capacity than the minimum MLU considering CSPF routing. It turns out that while 25 links (out of 114) requires a higher capacity, 74 are not used by PIRA (16 of them are not used by CSPF either). This suggests the idea of either: *a*) rewiring the network (instead of increasing the existing links capacities) or, *b*) employing the unused links to create lightpaths providing the extended capacity to overused links. These ideas will be studied in further contributions.

We compare now the best solution without stacking (CSPF-MP2P) with the best solution using the stack (PIRA-MPTF). The maximum number of labels that CSPF-MP2P uses is 20. This makes all AOLS-blocks to be sized for coding labels of five bits long. By using stacking (PIRA-MPTF), the maximum number of labels becomes 11, making the AOLS-blocks to be sized for coding labels of only four bits long. In Fig. 3.13 we show the number of links that are using a number of labels within a given range. It shows that 11 links are causing the five bits long labels without stacking. However, using the stack, only four links are forbidding us from using three bits long labels. Even though we did not optimize the maximum number of labels per link, it is not difficult to see that rerouting the traffic in the three links of PIRA-MPTF would be easier than rerouting the traffic of the 11 links in CSPF-MP2P in order to reduce one bit the label encoding size.



(a) Label Space Size vs. Link Capacity.



(b) Blocked Bandwidth.

Figure 3.11: Heuristics Overall Performance.

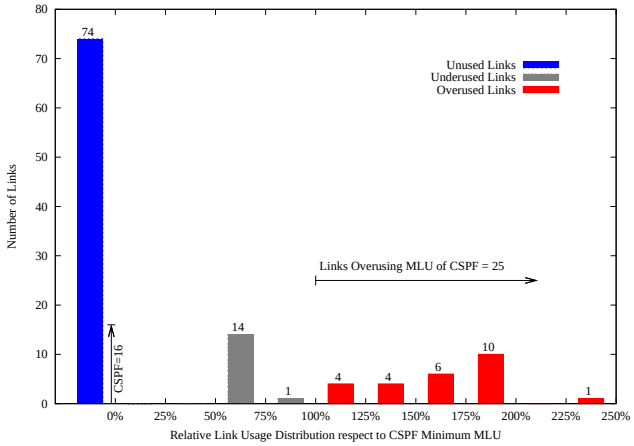


Figure 3.12: Distribution of overused link capacities of PIRA respect to CSPF MLU.

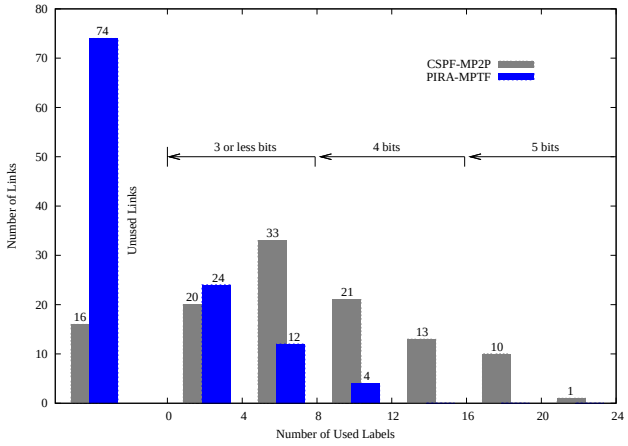


Figure 3.13: Distribution of the label space link-by-link using PIRA-MPTF.

Considering the header in [RKM+05], we compute the overhead due the coding of the extra label. In the case of no label stacking, the five bits long label yields to a 17 bits header. In the case of label stacking, the four bits long labels yield to a header of 23 bits. Assuming a classical packet distribution (as in [CCPD06]), the average overhead caused in the traffic of the network is just 0.18% more due stacking.

It is worth noting that if *label stripping* using CSPF is considered, packet header must code eight labels (minimum distance in CSPF paths) in the header and the network must be able to handle 102 labels (the number of used links by CSPF). If PIRA is considered, packet header must code 17 labels, and the network must be able to handle 40 labels. However, the traffic overhead is 2.02% and 4.5% more, respectively. Therefore, label stacking offers a better trade-off.

3.6.2 How Far From Optimum?

Each one of the two heuristics presented in this chapter affect the overall optimality of the solutions. Therefore, we compare our results with the optimal in two steps:

Minimize Labels considering CSPF/PIRA Routes

We route the traffic using CSPF and PIRA and then we relax the proposed ILP so it computes the best label space reduction using CSPF/PIRA paths. We consider the network scenario in which the MLU limit is the highest. By solving the relaxed ILP using the CSPF routes, the minimum number of used labels considering stacking is 180. Considering PIRA, the minimum number of used labels decreases down to 109. In this way, we can claim that: *a*) PIRA-routing leads to paths aiming at better label space reduction in general and, *b*) MPTF performs 21.11% far from its optimal value considering PIRA. Henceforth, we only consider PIRA-MPTF.

Minimize Labels with Variable Routes

We solve the complete ILP model proposed in §3.4.4 with a smaller network shown in Fig. 3.14. Edge nodes and demands are generated following the same parameters of the European network. The MLU is set to the double of the minimum needed by CSPF. We found that PIRA-MPTF label space size is 26.7% more than the optimal. A similar test to the previous one exposes that 68% of the error is caused by the selection of the routes. The reason is the tendency of PIRA for using more paths than the optimal. This behavior is mainly explained by the link congestions caused by PIRA and by the usage of long routes.

3.7 Chapter Remarks

In this chapter, the label space reduction problem in AOLS was tackled. First of all, an overview of the state of the art in AOLS is given. Second, an extension of the current architecture, given in the LASAGNE project, is made in order to allow the stacking (pushing and popping) of one label completely optically.

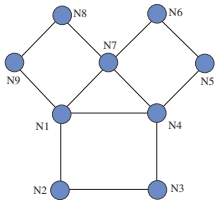


Figure 3.14: Network simulated for ILP

Due the cost expenditures of handling a label in AOLS, the routing problem is considered together with the label space reduction problem. So, the purpose is to route traffic and create tunnels in such a way that the minimum number of labels is used. As a consequence, it was foresen that the routing solution would create bottlenecks. Therefore, as a trade-off metric, the maximum link utilization in the network is analyzed as well.

The results show that, due the flexibility of routing, MP2P achieves greater reduction ratios than without such flexibility. AMTs, on the other hand, are not able to gain much. However, AMTs use better the spare capacity in the links than MP2Ps in order to decrease the label space.

Chapter 4

Reducing Label Swapping by G^+ Optical Bypassing*

Until this point, label merging and label stacking have been the two main points of discussion for reducing labels in either GMPLS or AOLS technologies. Both of them have a common assumption: the routes demand are given, hence fixed. It is clear that the longer the routes are, the more overall labels are needed in the network.

However, in AOLS systems it could be desirable to sacrifice (up to some point, of course) QoS routing parameters in order to achieve cheaper AOLS implementations. The most straightforward solution to reduce labels would be a CSPF algorithm routing all demands. Such solutions would lead to feasible and cheaper solutions in terms of overall AOLS cost [CCPD06].

WDM offers a broader option, if consider. It can be foreseen that many demands share path segments between two non-neighboring WRS. With the aid of an additional PXC, wavelengths can be set to *bypass* the node without any processing at all, creating lightpaths. As a consequence, lightpaths are setup in order to reduce the number of subwavelength switchings in the network (hence optical label processing).

The main contribution in this chapter concerns the G^+ [SCdO$^+$07, SCF$^+$06, MSdO$^+$06, MCS$^+$07], *Enhanced Grooming*, architecture as a means to reduce subwavelength switching. The chapter is devoted to quantify how much subwavelength switching can be reduced by employing G^+ instead of the traditional WRS architecture. Therefore, no special analysis is done concerning the integration with OPS (or AOLS) in this chapter, but in the next one.

This chapter is organized as follows. First of all, lightpaths, and the architecture supporting them, are explained as a method for *bypassing* the OPS. Secondly, the proposed architecture is detailed. In the next four following sections, an Integer Lineal Programming model (ILP) model and an heuristic are proposed and evaluated for using G^+ with the purpose of reducing subwavelength switching. Finally, chapter conclusions are given.

4.1 Optical Bypassing using Lightpaths

As mentioned, lightpaths are end-to-end optical connections established between pairs of WRSs in an optical network. In order to allow for a large set of lightpath configurations in a network, a typical WRS architecture must both optically forward lightpath traffic to other WRSs and transmit (or receive) optical traffic over (or from) a lightpath. Fig. 4.1 shows the classical WRS architecture aimed at working with lightpaths at wavelength granularity, which works as follows:

- Every single fiber is first demultiplexed into a set of W wavelengths.
- All the wavelengths coming from different fiber ports traverse a PXC device. This device may either redirect wavelengths to the wXC (for instance an OPS) or optically switch wavelengths fibers and/or colors to outgoing fibers.
- The wavelengths going to the wXC are processed and forwarded accordingly to the wXC specification.
- The wXC can either take this traffic out of the optical switch to local electronic equipments attached to low-speed ports, or forward it through another wavelength. This decision is made based on the headers.
- The incoming traffic to the optical node arriving from low-speed ports is processed locally together with dropped wavelengths traffic in the wXC.
- Outgoing optical packets are groomed into wavelength capacities by the wXC and then retransmitted over a wavelength.
- All outgoing wavelengths are multiplexed to their corresponding fiber.

For short, this classical architecture is abbreviated as 'G' in order to differentiate it from the proposed architecture, G^+ (to be explained in §4.2.2). For more details about classical WRS architecture, the reader is referred to [ZZM03].

Henceforth, it is assumed that an OPS instantiates the wXC. The OPS (hence, AOLS) is needed only when subwavelength switching is needed. The advantage of using lightpaths over OPS is that many optical packet processing are omitted.

Since one label identifies one demand in one node, the problem of reducing labels in AOLS is translated to reducing the number of demands that a node need to process; or in other words, minimizing the number of subwavelength switching in the network. As mentioned in the introductory chapter, the problem is best known as the Grooming, Routing and Wavelength Assignment (GRWA) problem in optical networks [ML01, Som06, Muk97], generally defined as:

> GROOMING, ROUTING AND WAVELENGTH ASSIGNMENT PROBLEM -
> GIVEN: A traffic demand matrix (Λ) and a description of the network topology including: *a*) the graph of the topology ($G'(V, E')$), *b*) the wavelengths that each link $(i, j) \in E'$ can be demultiplexed into (W), *c*) a minimum percentage of routed bandwidth ($B \in [0, 1]$), and *d*) a maximum number of virtual hops to be employed ($K \in \mathcal{N}$),
> QUESTION: Is there a set of optical routes (e.g. lightpaths) fitting within the network such that B% of the demanded bandwidth is routed using less than K virtual hops?

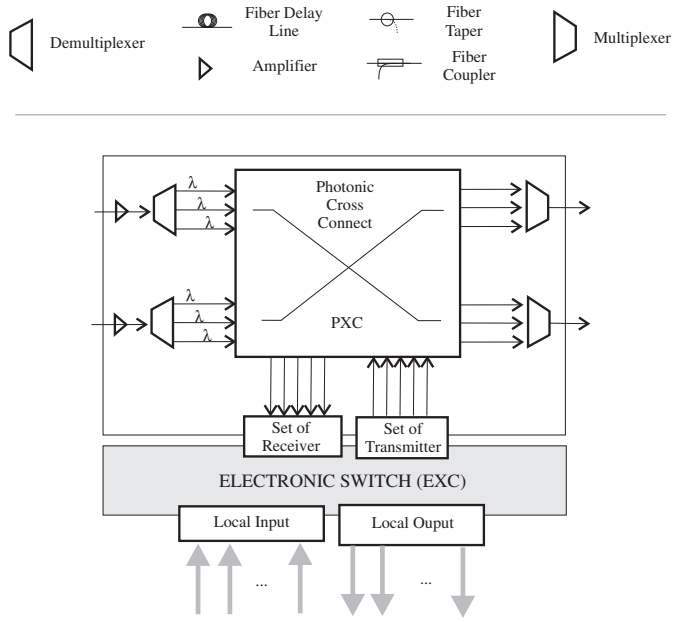


Figure 4.1: Classical WRS architecture

The problem has been solved under many different network scenarios using lightpaths [ZM02, ZZM03]. Therefore, it is not intended to analyze a new network scenario for the GRWA problem, but to propose a different architecture capable of reducing even more the number of subwavelength switchings (or virtual hops) in an OPS.

4.2 G^+ Network Architecture

In this section, the basics of G^+ are explained. In §4.2.1, a different node architecture for metro ring networks (RingO) is explained. Using these, a hybrid architecture, G^+, is proposed in §4.2.2 for handling lighttours. In §4.2.4, the proposed architecture is compared with light-trail's. Finally, assumptions for solving the GRWA problem using G^+ are presented in §4.2.5.

4.2.1 The RingO Architecture

RingO is a ring optical packet WDM network designed for metro applications. Although the RingO architecture has been recently improved [CFF+04], this dissertation uses one of its first versions, summarized here, since it fits better with G^+ needs.

In RingO, there is only one fiber connecting all nodes. The fiber can be demultiplexed into $|W|$ wavelengths. Every node receives its traffic in the ring by dropping one fixed wavelength of the fiber; no other node is allowed to drop that particular wavelength in the ring. All nodes that want to transmit to a specific node, tune their transmitter to that particular wavelength. Therefore, RingO allows only up to $|W|$ nodes in the network.

To prevent traffic contention in the ring, RingO works under the following assumptions: *a*) the network transmits optical packets in fixed time-slots, *b*) packets are optically coded with a fixed length, *c*) higher layers are able to segment and reassemble optical packets, and *d*) the tuning times of the transmitter are smaller than the slot duration.

Furthermore, RingO nodes are equipped with a λ-monitor device that allows it to check whether packets are being transmitted over a particular wavelength (λ) or not. The λ-monitor device is completely optical and very simple, and hence, fast. It only senses light passing through. If a λ-monitor device advises that a wavelength is free in a particular time slot for packet insertion, the RingO node may optically inject more traffic into it. Note that the existing traffic flow in this wavelength channel is not disturbed.

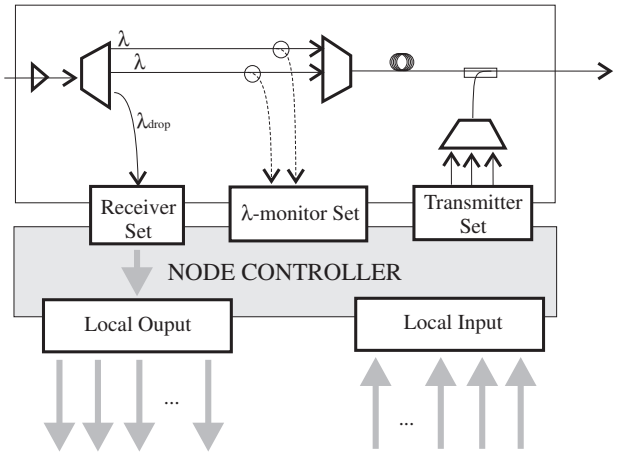
The architecture of a RingO node can be depicted in Fig. 4.2(a), which works as follows:

- The incoming fiber signal to a node is first amplified and then demultiplexed into a set of W wavelengths.
- One of the wavelengths (λ_{drop}) is dropped since it is traffic for that node.
- The rest of the wavelengths are tapped so that a fraction (10%) of the light passing through it is sampled.
- The tapped fraction of the power in every wavelength is analyzed by the λ-monitor device (in the λ-monitor set). The device only detects the received average power on a slot-by-slot basis in order to determine whether packets are passing through that particular wavelength.
- The incoming traffic to a node arriving from low-speed port(s) is queued locally.
- The node controller, based on the information the λ-monitor device provides, may inject more optical packets (queued previously in the electronic domain), by means of a tunable transmitter, into that free slot.
- The transmitters are connected to a multiplexer so that new traffic can be injected directly into the fiber.
- The existing data carried by the fiber is delayed. Therefore, the node controller decisions can be synchronized and no contention occurs in the optical domain.

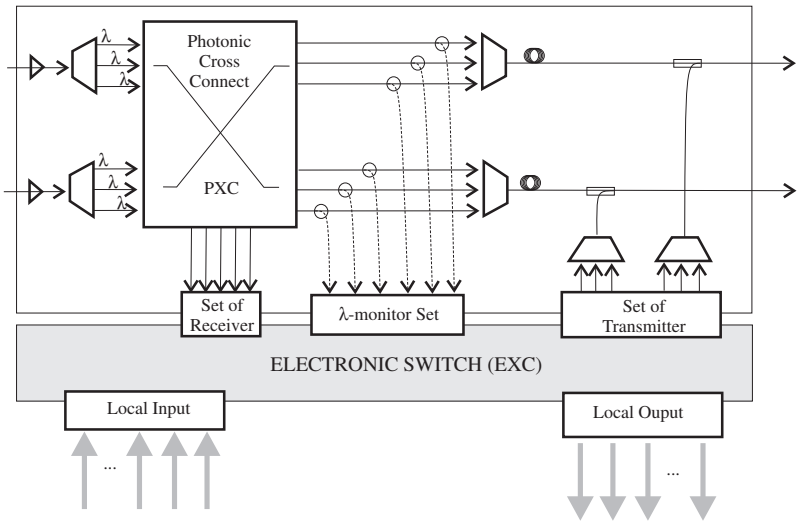
With RingO, existing traffic going through a wavelength does not need to be buffered electronically (no need for OEO conversion for existing traffic) in order to add more low-speed traffic. Simply, a node looks for a space (a time slot) between optical packets to allocate more traffic in the wavelength.

The RingO architecture also efficiently combines the advantages of electronic and optical technologies. The aggregated bandwidth is managed in the photonic (optical) domain, while packet queueing is managed in the electronic domain (making the proposed architecture free from expensive optical buffers).

Since this chapter focuses on showing the improved performance achieved by lighttours in the GRWA problem, the analysis of some details of the RingO architecture, such as optical signal codification, control plane or medium access



(a) RingO architecture



(b) G^+ architecture

Figure 4.2: G^+ and related architectures.

control protocols, are out of scope. The reader is referred to [CFF+04] for these details.

4.2.2 The Proposed WRS Architecture: G^+

Considering both the G architecture supporting lightpaths and the RingO architecture, a hybrid architecture (named G^+) is proposed with the objective of improving current traffic grooming techniques.

G^+ takes advantage of the wavelength switching flexibility in G for mesh networks and the way traffic is added to wavelengths (without incurring in any processing for the existing traffic in the wavelength) in RingO. This is accomplished by using the RingO node architecture as a base and adding a PXC device just before the λ-monitor. The G^+ WRS architecture is shown in Fig. 4.2(b).

The G^+ architecture works under the same assumptions as RingO's. WRSs transmit information using optical packets of fixed length, each corresponding to a time-slot (hence time-slots are also fixed). In G^+, packet headers are not read en route and hence, optical packet forwarding decisions are not made separately for each packet in the same wavelength. This simplifies the node architecture since the switching decisions are made at wavelength granularity and there is no need for optical buffers or any other scheme for solving contention in the optical domain.

A G^+ WRS works as follows:

- Every single fiber signal is first amplified and then demultiplexed into a set of W wavelengths.
- All the wavelengths coming from different fiber ports traverse the PXC device. This device may either redirect wavelengths to the wXC (for instance an OPS) or switch wavelengths from one fiber to another fiber.
- The wavelength going to the wXC (dropped wavelengths) are processed accordingly.
- The wXC can either take this traffic out of the optical node to local electronic equipments attached to low-speed ports, or groom it with more traffic for further optical forwarding using another wavelength. The forwarding decision is made using the underlying higher layer protocol information.
- The incoming traffic to the wXC arriving from low-speed ports is processed together with dropped wavelength traffic.
- The remaining wavelengths in the optical domain are tapped and a fraction of the light is redirected to a λ-monitor device (in the λ-monitor set).
- The λ-monitor device determines whether optical packets are transiting through a wavelength or not. This is normally performed using a DC-coupled photodiode array, a capable PXC, or a capable Amplifier.
- The wXC, based on the information the λ-monitor device gives, may inject more optical packets (processed previously in the wXC) into that free slot.

- The OPS interconnections lead to a set of multiplexers, so that new traffic can be injected directly into an outgoing fiber using an appropriate color.
- The existing data carried in the fiber is delayed so that the wXC decisions are synchronized and no contention occurs in the optical domain.

4.2.3 Lighttours Properties

As stated initially in this chapter, the proposed enhancement allows a transparent WRS to aggregate traffic over optical routes without incurring in any processing (like RingO does) for the existing traffic in the optical route. For example, if an optical route follows the path $s \rightarrow \alpha_0 \rightarrow \alpha_1 \ldots \alpha_n \rightarrow d$ - where s is the starting WRS of the optical route and d the destination or final WRS - then, the proposed scheme allows WRSs $\alpha_i,\ 0 \leq i \leq n$, to add more traffic without "breaking" the optical route (if the wavelength has enough bandwidth). Therefore, the G^+ "lightpaths" are able to make a *tour* over different traffic source WRSs and inject (en-route) their optical packets in the same optical route. We name the optical route resulting from the G^+ architecture a *Lighttour*. Lighttour properties are discussed through the rest of this section. The following example is given to illustrate some of its properties.

In Fig. 4.3(a), a network topology consisting of 4 WRSs and 5 bidirectional fiber links is shown. Fiber links can span one wavelength each. Each wavelength channel has capacity OC-12[1], but all source WRSs ($s1$, $s2$ and $s3$) only need a capacity of OC-3 to transmit to WRS d. Assume that the destination WRS d has only one available receiver. For this scenario, the best solution that classical grooming (G) may offer is to create 3 lightpaths (Fig. 4.3(b)), one each from $s1$ and $s3$ to $s2$, and another from $s2$ to d. This implies that WRS $s2$ need to switch subwavelength traffic for the demands coming from $s1$ and $s3$ (therefore, requiring two labels at $s2$, if AOLS is considered). In contrast, using G^+ (Fig. 4.3(c)) the demands can be routed with a single lighttour. Using this single lighttour ($s1 \rightarrow s2 \rightarrow s3 \rightarrow d$), the demands from $s2$ and $s3$ follow only a portion of the lighttour, i.e., they are not routed end-to-end in the lighttour. This leads to an improvement in the way the resources are used.

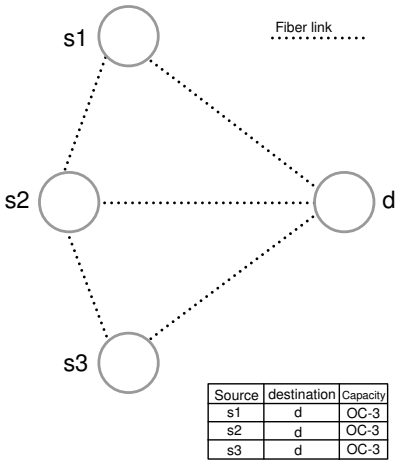
Next, three properties that differentiate lighttours from lightpaths are discussed.

Fact 8 (Asymmetric Bandwidth Utilization in a Route) *The bandwidth utilization of a lighttour increases over the route it takes since more traffic is added along its route. On the other hand, lightpath utilization is the same in all the wavelengths it takes.*

Fact 9 (One-to-Many Lighttours–Virtual-Links Mapping) *Since lighttours forward traffic from many WRS to a single one, a single lighttour could be seen as multiple virtual links in a virtual topology.*

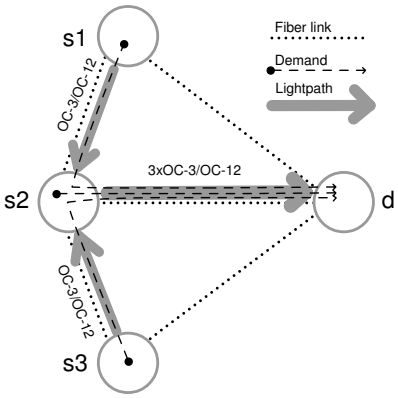
For instance, in Fig. 4.3(d), the virtual topology created by a single lighttour can be seen. It consists of three virtual links, since its corresponding lighttour connects three source WRSs to the same destination.

[1]OC-1 (optical carrier one) is equivalent to a SONET line with transmission speed of 51.84 Mbps using optical fiber.

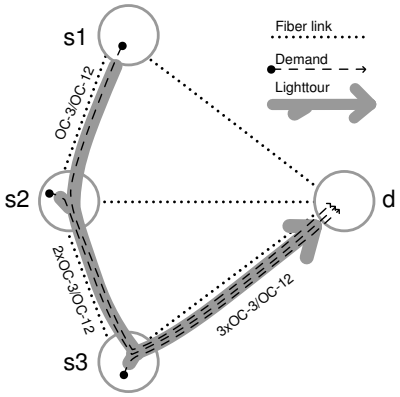


Source	destination	Capacity
s1	d	OC-3
s2	d	OC-3
s3	d	OC-3

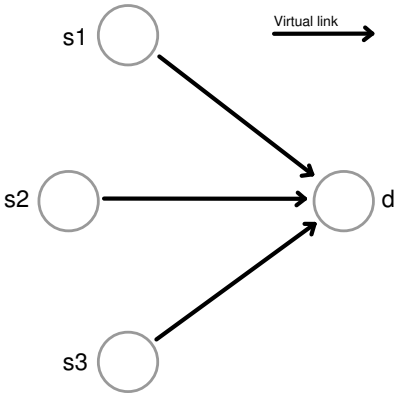
(a) Physical topology and demands.



(b) Classical grooming solution.



(c) G^+ solution.



(d) Lighttour virtual topology.

Figure 4.3: Difference between classical grooming (G) and G^+ solutions.

Although a proper discussion on synchronization issues is not addressed in this dissertation, it should be noted that the transmission delay of lighttour packets injected in the middle of a wavelength (e.g., while WRS seeks for free slot) is the same as that caused by grooming these demands at the source of a lightpath: the source node would have to queue traffic in order to transmit it through the lightpath. This is considering that the propagation time in fibers is insignificant compared to the processing time of electronic packets.

4.2.4 Related Network Architectures

Another architecture that improves wavelength utilization in WDM mesh optical networks is *light-trail* [GZ03]. Light-trails are unidirectional optical routes crossing several optical nodes, such that any node in the light-trail can send information to any of its downstream nodes without having to reconfigure. The wavelength is shared in time and the medium access is arbitrated by a control protocol among the optical nodes that try to transmit data simultaneously.

A light-trail acts as an optical bus. For instance, let us assume a light-trail is created starting from node $s1$, passing through $s2$ and $s3$, and ending at d in Fig. 4.3(a). If $s1$ and $s2$ both request to communicate with d using the light-trail, they are not able to do it at the same time. One of them, say $s1$, uses the light-trail to transmit to d; and then, when the light-trail is completely free (the transmission is ended or interrupted), the other node, say $s2$, may use it.

The main difference between lighttours and light-trails lies in the partition of the wavelength in time-slots. The fact that light-trails create an optical bus makes them depend on the control plane for multiplexing the subwavelength connections. This could delay downstream transmissions, since the trail has to be empty before transmission.

We consider that light-trails are suited for bursty traffic where transient connections are needed and the requirements for providing quality of service are low. On the contrary, lighttours are more suitable for permanent connections with higher quality of service requirement for subwavelength traffic.

4.2.5 Assumptions

In the following sections, two different proposals (model and heuristic) are given to solve the GRWA problem with the aim of reducing the number of subwavelength switching. The following assumptions hold for the solutions proposed in this chapter:

- *Wavelength Continuity:* WRSs are not able to convert a wavelength in an input fiber to another wavelength for an output fiber, i.e., there are no wavelength conversion capabilities. Assuming full wavelength conversion capabilities makes the model simple.
- *WRS Connectivity:* There is at most one fiber link, in each direction, between every pair of WRSs. Moreover, all fibers have the same capacity.
- *Wavelength Multiplexing:* Fibers have the same number of wavelengths.
- *Multi-path:* One optical route (lighttour or lightpath) uses one wavelength. However, multiple optical routes may exist between two pair of WRSs if they use different wavelength channels.

- *Physical Hops and Optical Route Length:* All optical routes have limitations in physical hops and length. The reason is simple, optical signals need amplification and optical channels induce noise in the signal. Although these two parameters are considered in §4.3, none of them affect the results in the simulations[2].

- *Contention Resolution:* Contention resolutions issues are not addressed in this chapter, but in the following.

4.3 The Multilayer G^+ Model

Most classical grooming models consider three separate pairs of node-indexes for distinguishing between fiber links, virtual links, and source/destination WRSs of a demand. These models have two main sets of variables: one for lightpath routing (virtual topology) over existing fiber links, and another for routing demands over the lightpaths [ZM02][BM00]. Therefore, most of the variables are associated with two pairs of indices. These models usually route lightpaths over fibers and traffic demands over lightpaths as two routing subproblems considered together. One advantage that lightpath modeling has is that a lightpath is mapped directly to just a single virtual link, making routing over virtual links easier [Som06].

Because of *Fact* 9 described in §4.2.2, the traditional scheme of two independents routing submodels is difficult to apply. This led us to re-consider the way in which traffic grooming had been modeled thus far. The proposed model works slightly differently. It routes traffic demands over the physical topology *if* there exists a lighttour spanning it.

In the general GRWA problem, different objectives can be considered. For instance, in a resource constrained network, it is desirable to maximize the total traffic that can be routed over the network. On the contrary, given a minimum desired throughput, it can also be desirable to minimize the number of resources for routing the traffic.

4.3.1 An ILP Formulation for Multi-hop Enhanced Grooming

In this subsection, an ILP model for the multilayer problem where demands can be routed using several consecutive lighttours (i.e., multi-hop grooming) is proposed.

The following list are the indices used by the variables of the ILP proposed.

INDICES:

i,j,k WRSs in the network.

m A demand that needs to be routed.

t A lighttour in the network.

w A wavelength of a fiber.

[2]Many models in the literature do not consider these two parameters.

The model parameters are listed next, i.e., input data or constants.
PARAMETERS:

Λ^m Traffic demand m[3].

Δ_j^m 1 if WRS j is the destination for demand m, -1 if it is its source and 0 otherwise.

$F_{(i,j)}$ The physical (fiber) distance between nodes. It is assumed that the minimum distance between any connected pair of nodes is 1. A disconnected pair of nodes has distance 0.

C Capacity of a wavelength[3]. It takes positive integer values.

R_i Number of OPS ports (or AOLS blocks) in WRS i. It takes non-negative integer values.

H Maximum number of hops allowed for a lighttour.

L Maximum lighttour distance allowed.

Next, the decision variables for the model are described.
VARIABLES:

r^m 1 if demand m has been successfully routed, 0 otherwise.

$p_{(i,j)}^{m,t}$ 1 if demand m is routed through lighttour t in fiber (i,j), 0 otherwise.

$\lambda_{(i,j)}^t$ 1 if lighttour t uses a wavelength on fiber (i,j), 0 otherwise.

$\rho^{m,t}$ 1 if lighttour t routes demand m, 0 otherwise.

q_j^t 1 if WRS j is grooming additional traffic into lighttour t, 0 otherwise. In other words, it is 1 if a demand is being partially routed over lighttour t starting at WRS j, 0 otherwise.

$d_{(i,j)}^t$ 1 if link (i,j) is the last one for lighttour t, 0 otherwise.

γ_w^t 1 if lighttour t uses wavelength w, 0 otherwise.

The formulation requires *apriori* knowledge of the number of optical routes of the optimal solution. Therefore, the dimension of index t should be bounded by a large enough number such that all optical routes can be computed. In the worst case, it can be set to $\min(|E| \times |W|, |M|)$, where E, W, M represent the set of indices for links, wavelengths and demands respectively.

All the variables used in the model are binary. Therefore,

$$p_{(i,j)}^{m,t}, l_{(i,j)}^t, d_{(i,j)}^t, \rho^{m,t}, q_j^t, r^m, \gamma_w^t \in \{0,1\} \tag{4.1}$$

[3]All capacities are expressed as multiples of OC-1.

Among all the objectives discussed for the multilayer problem, the one concerning this dissertation is considered in this chapter. The number of subwavelength switching needed to route a given minimum throughput can be minimized with:

MINIMIZE

$$\sum_{m} \left[\sum_{t} \rho^{m,t} - r^{m} \right] \tag{4.2}$$

The $\sum_{m_t} \rho^{m,t}$ term adds the number of virtual hops taken by the routed demands. Since the number of subwavelength switching for a given demand is one less than the number of virtual hops needed, the term $\sum_m r^m$ is subtracted to obtain the correct value.

SUBJECT TO:

Routing Constraints

$$\sum_{i,t} p^{m,t}_{(i,j)} - \sum_{i,t} p^{m,t}_{(j,i)} = \Delta^m_j \cdot r^m, \qquad \forall m, j, \tag{4.3}$$

$$p^{m,t}_{(i,j)} + p^{m,t}_{(j,i)} \leq 1, \qquad \forall i, j, t, m \tag{4.4}$$

$$\sum_{i} \lambda^{t}_{(i,j)} \leq 1, \qquad \forall j, t \tag{4.5}$$

$$d^{t}_{(i,j)} + \rho^{m,t} - p^{m,t}_{(i,j)} \leq 1, \qquad \forall i, j, m, t \tag{4.6}$$

$$p^{m,t}_{(i,j)} - p^{m,t}_{(j,k)} + \lambda^{t}_{(j,k)} \leq 1, \qquad \forall i, j, k, m, t \tag{4.7}$$

where (4.3) and (4.4) are the basic flow conservation constraints for a demand (regardless of the lighttour). (4.5) assures that each lighttour is linear-shaped, i.e., not a tree. (4.6) forces the use of the last link of a lighttour t for a demand m, if the demand is being forwarded in t.

The last constraint, (4.7), relates to the base of the G^+ model. It allows a demand to be routed through a portion of the lighttour. This means that if a demand is routed in a link (i, j) using a lighttour and this lighttour continues to be active in (j, k) (for any k), then the demand must be routed in (j, k) using the same lighttour. Therefore, if a demand "wishes" to get routed through a lighttour $(s \rightarrow \alpha_0 \rightarrow \alpha_1 \ldots \alpha_k \ldots \alpha_n \rightarrow d)$ starting at WRS α_k, the demand must follow the lighttour route from α_k until the end $(\alpha_k \rightarrow \alpha_{k+1} \ldots d)$ regardless of the unused portion of the lighttour $(s \rightarrow \alpha_0 \ldots \alpha_k)$.

Note that this avoids the need to handle special flow conservation constraints - except (4.5) - for the $\lambda^t_{(i,j)}$ variables. This is because every pair $p^{m_1,t}_{(i,j)}$ and $p^{m_2,t}_{(i,j)}$ is tied down in the same link if the lighttour t is "activated" in (i, j) (i.e., $\lambda^t_{(i,j)} = 1$).

Wavelength Continuity Constraint

$$\gamma^{t_1}_w + \lambda^{t_1}_{(i,j)} + \gamma^{t_2}_w + \lambda^{t_2}_{(i,j)} \leq 3, \qquad \forall i, j, t_1 \neq t_2, w \tag{4.8}$$

$$\sum_{w} \gamma^{t}_w - \sum_{(i,j)} d^{t}_{(i,j)} = 0, \qquad \forall t \tag{4.9}$$

where (4.8) and (4.9) are the wavelength continuity constraints, i.e., these two equations prohibit wavelength conversions in the network. By limiting the left hand side of (4.8) to 3, two different lighttours cannot use the same wavelength on the same link. Additionally, since the index w takes a finite set of values, the number of wavelengths used in a fiber is automatically restricted.

Constraint (4.9) allows a lighttour to use only one wavelength. It should be pointed out that $\sum_{(i,j)} d^t_{(i,j)}$ is 1 if the lighttour t is being used by any demand and 0 otherwise.

Capacity Constraints

$$\sum_m \left[\Lambda^m \cdot p^{m,t}_{(i,j)}\right] \leq C, \qquad \forall i,j,t \tag{4.10}$$

$$\sum_{i,t} d^t_{(i,j)} \leq R_j, \qquad \forall j \tag{4.11}$$

$$\sum_{(i,j)} \lambda^t_{(i,j)} \leq H, \qquad \forall t \tag{4.12}$$

$$\sum_{(i,j)} F_{(i,j)} \cdot \lambda^t_{(i,j)} \leq L, \qquad \forall t \tag{4.13}$$

where (4.10) limits the bandwidth used by a lighttour. Note that the constraint runs over all links since the used capacity of a lighttour varies from link to link (as mentioned in Fact 8 in §4.2.2). (4.11) bound the number of OPS ports per WRS.

Like other optical routes, lighttours have hop and length limitations. (4.12) and (4.13) address these two aspects, respectively.

Relationship between Variables

$$\sum_i p^{m,t}_{(j,i)} - \sum_i p^{m,t}_{(i,j)} - q^t_j \leq 0, \qquad \forall j,m,t \tag{4.14}$$

$$p^{m,t}_{(i,j)} - \sum_k p^{m,t}_{(j,k)} - d^t_{(i,j)} \leq 0, \qquad \forall i,j,m,t \tag{4.15}$$

$$p^{m,t}_{(i,j)} - F_{(i,j)} \leq 0, \qquad \forall i,j,m,t \tag{4.16}$$

$$p^{m,t}_{(i,j)} - \lambda^t_{(i,j)} \leq 0, \qquad \forall i,j,m,t \tag{4.17}$$

$$p^{m,t}_{(i,j)} - \rho^{m,t} \leq 0, \qquad \forall i,j,m,t \tag{4.18}$$

Constraint (4.14) increases the lower bound of q^t_j to 1 if WRS j is grooming traffic on t for any demand d. Since only one OPS interconnection is needed to groom several demands in a WRS, $q^t_j = 1$ regardless of how many demands are groomed by WRS j in lighttour t.

Constraint (4.15) defines variable $d^t_{(i,j)}$ by imposing a lower bound of 1 when j has no outgoing traffic but it has incoming traffic from i for at least one demand of t.

Constraint (4.16) sets an upper bound of 0 to p in those links in which there is no fiber connecting the nodes. (4.17) activates a lighttour link (i,j) if there is at least one demand using it. (4.18) registers the relationship between a routed demand and the lighttours it traverses.

4.3.2 Constraining for Classical Grooming Modeling

For classical grooming modeling, the ILP can be configured to limit the number of grooming WRSs that every lighttour has to at most one. Limiting this quantity to one creates lighttours having only one grooming WRS, the source WRS. This can be formulated by adding the following constraint.

$$\sum_j q_j^t \leq 1, \forall t \tag{4.19}$$

4.3.3 Other Common Constraints

A very common constraint when modeling grooming is to restrict routing of demands to one virtual hop. This way, no subwavelength switching is employed but a relative "smaller" maximum throughput is obtained. This can be done by adding a constraint to restrict the number of lighttours associated with each demand as follows.

$$\sum_t \rho^{m,t} \leq 1, \quad \forall m \tag{4.20}$$

The model can also be modified to allow full wavelength conversion. In order to do so, (4.8) and (4.9) must be removed, and instead, an equation limiting the number of wavelengths per fiber should be added:

$$\sum_t \lambda_{(i,j)}^t \leq |W|, \quad \forall (i,j) \tag{4.21}$$

where $|W|$ is the number of wavelengths per fiber.

4.4 Comparing G^+ and Classical Grooming: A Numerical Example

It is easy to see that G^+ will, in the worst case, behave like classical grooming with any ordinary networking objective function, since all feasible solutions for classical grooming are also feasible for G^+. However, in this section, a set of numerical solutions[4] for the ILP model proposed in §4.3 is presented in order to highlight the improvements achieved.

The simulation scenario (network topology and traffic demand matrix) in this section mirrors previous studies of the GRWA problem with ILPs (e.g. see [ZZM03] and [ZJM00b]). The network shown in Fig. 4.4 is used in the simulations. It is considered that each fiber link may be demultiplexed to at most $|W|$ wavelengths and each WRS in the network has $|T|$ OPS interconnections. The capacity of each wavelength is OC-48. Wavelength conversion is not allowed.

The traffic demand matrix is generated similarly to [ZZM03]. Each demand bandwidth may correspond to OC-1, OC-3 or OC-12. The number of OC-1 demands between any pair of nodes follows an uniform distribution between 0 and 10, OC-3 demands follow an uniform distribution between 0 and 6, and OC-12 demands follow an uniform distribution between 0 and 2. In total, 235

[4]The solver used is Xpress by Dash Optimization, release 2005B for 32-bits Linux.

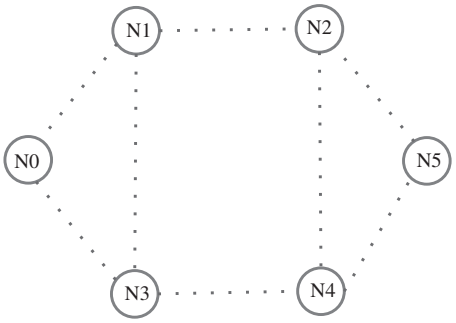


Figure 4.4: Example of physical topology.

demands are generated with a total bandwidth equivalent to OC-585: 123 OC-1 demands, 98 OC-3 demands, and 14 OC-12 demands.

To compare the results, the model is solved twice[5]: once using lighttours and another using lightpaths[6].

As explained before, even though a WRS has to wait for free slots, the proposed architecture does not incur in any extra major delay for the routed traffic. Therefore, the main cause of delay is the same as in the classical architecture: subwavelength switching processing. In this subsection, numerical examples minimizing the number of subwavelength switching in different scenarios are presented.

For consistency reasons, the single-hop routing constraint is disabled in this set of experiments. The number of interconnections with the wXC is fixed to 5 per WRS and the number of wavelengths is uniformly varied from 2 to 5. The solver is asked to solve the model routing *all* demands while using the minimum number of subwavelength switching.

Fig. 4.5 shows the minimum number of virtual hops needed to route all demands in each scenario. The number of used OPS interconnections in the network for each scenario is shown at the end of its corresponding bar.

Note that the minimum number of virtual hops is 235, since each one of the 235 demands needs at least one virtual hop to be routed. Therefore, the excess in the number of virtual hops (i.e. above 235) is due to the use of subwavelength switching. The results show that G^+ *can route a fixed amount of traffic using half the number of subwavelength switching needed by classical grooming.* For instance, for two wavelengths per fiber, G^+ needs to switch subwavelength demands less than 18 times, while classical grooming needs to do it 49 times. When the number of wavelengths is set to 3, G^+ is able to route all traffic without subwavelength switching, whereas G needs 5 wavelengths for the same purpose. When both architectures route all demands without subwavelength switching, G^+ need less optical routes (hence, less wavelengths) than G, simplifying network management.

Since the maximum number of interconnections with the OPS is almost reached for the worst case pure-lighttour scenarios (when $|W| = 2$), it is believed that less subwavelength switching can be achieved by increasing the number of

[5]Most of the solutions presented here for the lighttours scenarios are at most 2.5% away from the optimal value.

[6]By restricting the model as mentioned in §4.3.2.

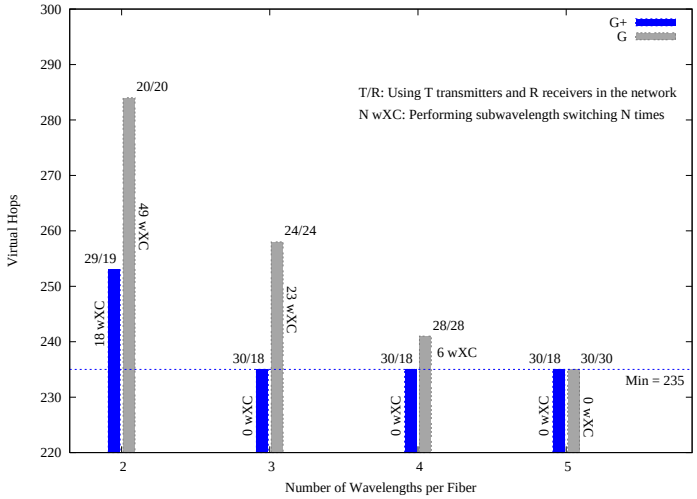


Figure 4.5: Minimum Number of Virtual Hops needed by G and G^+.

available interconnections.

It should be pointed out that while these results show that G^+ *uses* more interconnections than G, they only imply that G^+ uses the available resources more efficiently.

Other experiments show that the difference between allowing full wavelength conversions and no wavelength conversion is not significant with regards to the improvement of the objective function, i.e., the number of subwavelength switching is still reduced to half of its original value.

4.5 Heuristic

Obtaining the optimal solution using the model could be very expensive in terms of computational time. Hence, an heuristic capable of finding near optimal solutions for the GRWA problem in polynomial time, using G^+, is proposed in this section.

This section is divided in two parts. Initially, in §4.5.1, some definitions are given and later, in §4.5.2, the heuristic is described.

4.5.1 Definitions

The proposed heuristic uses a special virtual topology, also proposed in this thesis. Although the definitions given in this section apply to both architectures, specific details and properties for lightpaths are not explored. The special virtual topology and other related concepts are detailed in this subsection.

Definition 9 (eXtended Virtual Topology - XVT) *Given a physical network $G'(V, E')$ and a set of lighttours L routed over the physical topology G', an* eXtended Virtual Topology (XVT) *of G' taking L - represented now on as $G_L(V, E)$ - is a directed multi-graph created by taking all the WRSs in V, the*

links of the physical topology having free wavelengths, and the virtual links that map L to a common virtual topology.

As mentioned in Fact 9, a lighttour could be mapped onto more than one virtual link. This way, if a lighttour spans over k fibers links in the physical topology, the lighttour is mapped to k different virtual links, one for each of the feasible grooming source nodes.

Definition 10 (Fiber and Lighttours Virtual Links) *As the previous definition states, an eXtended Virtual Topology (XVT) is composed of two types of virtual links: those mapped from fibers with available wavelengths,* Free-Wavelength Virtual Links *(represented with a single arrow '$\rightarrow$' in the text), and those mapped from lighttours,* Lighttour Virtual Links *(represented with a double arrow '$\rightrightarrows$' in the text).*

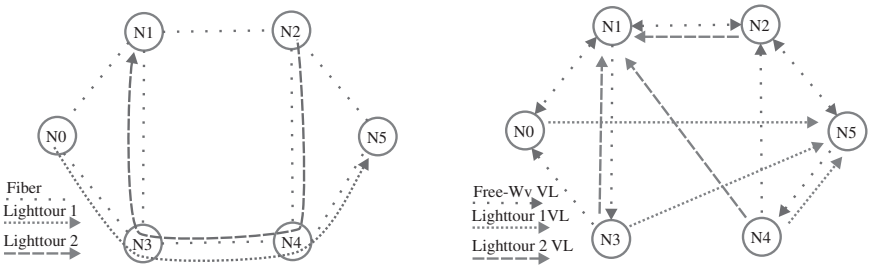
A path over an eXtended Virtual Topology (XVT) may consist of links representing either fibers or lighttours. Therefore, given a demand from s to d, finding a route over an XVT may imply taking lighttours (possibly only a portion of the lighttour) and/or using available fibers for the creation of new lighttours. This is clearly advantageous. Within a single multi-graph (the XVT), all possible means to route a demand can be considered at the same time, using only existing lighttours (a multi-hop solution), creating lighttours from raw (available) fibers (a single-hop solution), and a mixture of both (a multi-hop solution with new lighttours). Note that the creation of an XVT is $O(n+e+n \cdot l)$, where n, e and l are the number of WRSs, physical links, and existing lighttours respectively.

Since there is a direct mapping of a WRS in a physical topology to its XVT, the number of free (unused) OPS connections of a node can be easily handled. When a WRS in a physical network runs them, the XVT can cut off all incoming Free-Wavelength Virtual Links to the mapped node in the XVT. Similarly, when a WRS in a physical network runs out of OPS interconnections, the XVT can cut off all outgoing Free-Wavelength Virtual Links and also all Lighttour Virtual Links that are not currently grooming traffic in their corresponding lighttour. This prevents the creation of demand paths requesting non-available OPS interconnections.

Example

Consider the network in Fig. 4.6(a) with 6 WRSs interconnected by 8 bidirectional fiber links and 2 lighttours lt_1 and lt_2, which satisfy a set of traffic demands. Assuming that there is only one wavelength per fiber and that no WRS has run out of OPS interconnections, the XVT is shown in Fig. 4.6(b). It should be pointed out that some of the links may correspond to lighttours and others to fiber links. For example, while links $N1 \rightarrow N2$, and $N5 \rightarrow N4$ are Free-Wavelength Virtual Links, $N0 \rightrightarrows N5$, and $N4 \rightrightarrows N1$ are Lighttour Virtual Links taken from lt_1 and lt_2, respectively. In addition, note that both lt_1 and lt_2 are each mapped to 3 Lighttour Virtual Links; the former with destinations $N5$ and, the latter with destinations $N1$.

Since the proposed heuristic (explained in detail in §4.5.2) is based on the shortest path algorithm, weights are given to links for setting different heuristic behaviors. The weight of a Free-Wavelength Virtual Link is represented by w_F, while the weight of a Lighttour Virtual Link is represented by w_L.



(a) Physical topology with 2 lighttours. (b) eXtended Virtual Topology (XVT).

Figure 4.6: Network topology example.

Definition 11 (Weight of Virtual Links) *The relationship between these two weights affects the algorithm's objective as follows:*

- Case $w_F \ll w_L$*: a shortest path algorithm will aim at creating new lighttours to satisfy the demand (when possible), i.e., it will avoid taking existing lighttours to route the demand. Since new lighttours would be created for new demands, the number of subwavelength switching could be minimized.*

- Case $w_F \gg w_L$*: a shortest path algorithm will avoid (when possible) creating new lighttours to route the demand, and will use existing lighttours to route new demands. Therefore, the number of used wavelengths could be minimized.*

4.5.2 The Shortest-2-Shortest Heuristic

The GRWA is usually separated into two subproblems (most of the times closely tied), Traffic Grooming and Routing and Wavelength Assignment for the sake of simplifying complexity. Since the simulation experiments in §4.4 showed that the rate of the reducing subwavelength switching is the same with or without the wavelength continuity constraint, the Traffic Grooming and Routing subproblem is tackled. The Wavelength Assignment subproblem can be solved with one of the heuristics in [ZJM00a].

One way to route traffic with the least number of lighttours (hence, less virtual hops and less lables) is to find lighttours crossing several sources[7]. In other words, given a destination, the least number of lighttours that make a *tour* over all the sources is found.

The complete heuristic pseudo-code is given in Fig. 7. The heuristic has two nested loops. The outer loop (lines 3 to 24) selects a destination WRS d, which satisfies the greatest possible number of demands. The inner loop (lines 10 to 18) computes a path l over the XVT traversing several sources to d. The inner loop computes an extension of path l, named p, in each iteration such that a new source can be routed in l (the new source may be a source of several demands to d; the Λ' set in line 12). The path extension, p, is computed[8] (initially in line 8

[7]Fig. 4.3(c) intuitively illustrates this idea.

[8]The links in p, and those going to any WRS in p, are removed from G. This way, the next time the shortest path algorithm is run, the new path p cannot create a loop in l.

Algorithm 7: Procedure S2S(G': physical network, Λ: demands)

```
1  begin
2      Set the resulting set as empty, L ← ∅;
3      repeat
4          Create an extended virtual topology G_L(V, E) considering the physical
           topology G' and the lighttours in L;
5          Select WRS d as the destination of the greatest amount of non-routed traffic;
6          Initialize lighttour path l as empty;
7          Let N be the set of source WRSs with non routed demands to d;
8          Let p be the shortest route from a WRS s ∈ N to d;
9          Let routedBw ← 0 and bwLimit ← C;
10         while p exists do
11             Extend lighttour path l with the links in path p;
12             Let Λ' ⊆ Λ be the set of demands from s to d that can be satisfied with
               bwLimit;
13             Let e* be the available bandwidth of the link in l with the least available
               bandwidth;
14             Increase routedBw with all the satisfied demands in Λ';
15             Set bwLimit ← e* − routedBw;
16             Mark all demands in Λ' as routed;
17             Retain, in N, the sources demanding less than bwLimit units, i.e.
               N ← N − Λ';
18             Let p be the shortest route from a WRS s ∈ N to the first WRS in l;
19         if l ≠ ∅ then
20             Let l' be the Free-Wavelength Virtual Links of l;
21             Create a lighttour for each set of consecutive links in l'. Let L' be the set
               of created lighttours;
22             Reduce the available capacity of all the lighttours virtual links in l and in
               L';
23             L ← L ∪ L' ;
24     until N ≠ ∅ and l ≠ ∅ ;
25 end
```

and later in line 18 inside the loop) using a shortest path algorithm from one of the source WRSs, from the set N not yet included in the path, to the first WRS of the path l.

If there are several shortest paths going from different sources, the one demanding more bandwidth is selected. The variables *routedBw* and *bwLimit* keep track of the used and available bandwidth of the new route, respectively. For better efficiency, in each iteration of the inner loop, the set N is reduced so that only demands having less bandwidth than that available in the path (*bwLimit*) are considered.

4.6 Heuristic Performance

In this section, two sets of simulations of the proposed heuristic are analyzed. The first set of simulations, discussed in §4.6.1, aims at showing how close the heuristic solutions are to the optimal values computed by the ILP model proposed in §4.3. In all simulations shown in this section, it is assumed that WRSs perform full wavelength conversion when needed.

Since a comparison between G^+ and classical grooming was already discussed in §4.4, simulations of classical grooming are not described in this section.

4.6.1 How Good is the Heuristic?

In this subsection a subset of the scenarios analyzed in §4.4 is considered. The topology (see Fig. 4.6(a)) and traffic demand matrix are the same.

Since the objective is to route the maximum amount of traffic using the least number of subwavelength switching, the heuristic is run considering solely $w_L \gg w_F$ (scenarios varying weights are analyzed in the next subsection).

While the optimal maximum throughput is achieved using 5 OPS interconnections per node and 2 wavelengths per fiber (see Fig. 4.5), under the same circumstances, the heuristic routes 70.7% of total demanded capacity. However, if the number of wavelengths per fiber is increased to 4, the heuristic achieves 92.3% of the maximum throughput using the same number of interconnections. The remaining 8.7% is obtained if the network is provided with 3 additional interconnections per node, i.e., 8 interconnections per node in total.

Another interesting metric concerns the number of subwavelength switching. Fixing the amount of resources, while the ILP model solution routes 90% of the traffic using single-hop routing (see Fig. 4.5), the heuristic routes 63.7% using single hop and 7% using multi-hop. This means that $71\% = \frac{63.7\%}{90\%}$ of the traffic routed with the ILP model using single-hop can also be routed using the heuristic.

4.6.2 Shifting Weights

In this subsection the trade-offs brought on by the weights w_F and w_L in the heuristic are analyzed. Henceforth, a new topology is considered: the National Science Foundation network (see Fig. 4.7). For this topology, each wavelength has a capacity of OC-48.

The demand parameters follow the same type of distribution as those described in §4.4. The total demanded traffic is equivalent to OC-3399. In this

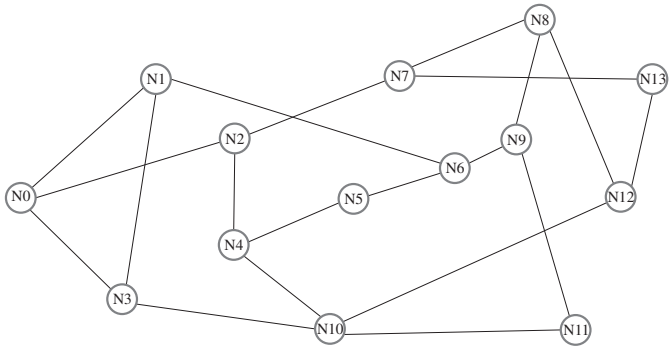


Figure 4.7: National Science Foundation network consisting of 14 nodes.

case, every WRS has 20 OPS interconnections available and every fiber can be demultiplexed in 20 wavelengths. The relationship between the weights is set to either $w_F = 1000 \times w_L$, $w_F = w_L$, or $w_L = 1000 \times w_F$.

As expected, the heuristic achieves the maximum possible throughput in the network regardless the values of the weights, but at a different cost. Table 4.1 shows this trade-off: three solutions with the same throughput are obtained with different weights. All solutions routed all demands.

By setting $w_F \ll w_L$ (first data column in Table 4.1) the heuristic uses 11 OPS interconnections less and 4 lighttours less in total than when $w_F \gg w_L$ (third data column in Table 4.1). However, the former's price is to use 24 demultiplexed wavelengths more than the later solution. Clearly, if $w_F \gg w_L$, the lighttours would be shorter and, hence, traffic would incur in more subwavelength switching, as reflected in the last two columns in the table.

Taking this trade-off into account,it can be concluded that $w_F \gg w_L$ is suitable for dynamic networks in which new connections are set and torn down quickly: since lighttours are shorter, it offers a high connectivity degree in the virtual network topology. On the contrary, $w_F \ll w_L$ is best suitable for static networks where optimal resource utilization and quality of service provisioning is highly desirable.

Table 4.1: Trade-offs for different weights, w_F and w_L.

Relation	$w_F \ll w_L$	$w_F = w_L$	$w_F \gg w_L$
Num. of Lighttours	79	81	83
Used Interconnections[1]	238	241	249
Used Wavelengths[2]	325	320	301
Max. of Virtual Hops	2	3	4
Num. of Virtual Hops	1435	1524	1637

[1] The total number of interconnections is 400 (20 per WRS).
[2] The total number of wavelengths is 400 (20 per fiber).

4.7 Chapter Remarks

In this chapter, a new WRS architecture, named G^+, is proposed. G^+ allows the setup of an optical route that may gather information from many sources while en route towards the destination, therefore reducing the number of OPS. These new optical routes are called *lighttours*.

G^+ is a hybrid architecture of RingO and WDM switching. A WRS in G^+ includes a simple optical device, named λ-monitor device, that enables the routing through lighttours.

To compare G^+ and classical grooming, an ILP model was proposed for solving the GRWA problem considering minimizing the number of subwavelength switching. The ILP model is solved in different scenarios with a given traffic demand matrix.

As a general result, using G^+, a reduction of 50% in the number subwavelength switching is achieved.

A polynomial-time heuristic was proposed as well. The heuristic results are compared with those of the ILP model. Simulation results show that for 70% of the demands, the heuristic results used a few more optical resources than the optimal solution.

Chapter 5

Conclusions & Future Work

5.1 Thesis Conclusions

In this thesis, the label space reduction problem was analyzed. Two different technologies were discussed: plain GMPLS networks and AOLS-based networks. Each of which was discussed in separate parts of this document.

In the first part, concerning GMPLS it is assumed that a set of paths are given and the contributions aim at reducing the number of labels as much as possible. Two methods were discussed: *label merging* and *label stacking*.

Label merging has been previously studied by several authors. Concerning label merging, one of thesis main conclusions is that computing the minimum number of labels using label merging can be performed in polynomial time [SFM07]. This observation nullifies most of the contributions previously published in the field.

Label stacking, by the contrary, has not been studied so deeply before. The contributions in this dissertation concerning label stacking are considerable greater in number. Initially, the *stacking problem* is described and its properties are discussed. Then, two algorithms and two mathematical models are proposed for solving the stacking problem. Finally, simulation results show that, in average, the label space can be reduced by 40% using label stacking and 20% using label merging.

In the second part, concerning AOLS the routing problem is tackled together with the label space reduction problem. The reason for considering routing as part of the problem is extending the possibilities of reducing label spaces, since an optical label implies an economic cost.

Initially, the label merging and label stacking approaches (studied in the previous part for GMPLS) are mapped to AOLS-based networks. This implies the update of the current AOLS-block architecture, the first contribution in this part. Then, since routing is considered, a new (and more complex) mathematical model is proposed as well. The results show that MP2P improves its reduction ratio, while AMTs do not gain much. However, AMTs use better the spare capacity in links for reducing label spaces, more than MP2Ps do.

The last chapter is devoted to a different approach for reducing labels (or OEO conversions in a non-AOLS-based architecture): *optical bypassing*. The traditional WRS architecture is modified, so lightpaths can be extended to *light-*

tours. The new WRS architecture is denoted as G^+, coming from *enhanced grooming*. Basically, lighttours allow the network to route traffic using less OEO conversions, which turns out to be less labels in an AOLS-based network. In order to evaluate G^+, a mathematical model and an heuristic are proposed. Simulation results show that G^+ can route traffic using half of the OEO conversions, or labels, than the traditional WRS architecture.

5.2 Future Work

The present document contains few contributions on the broad field of Label Space Reduction. The research can be extended in the following lines.

5.2.1 Faster Solutions

Approximation Algorithm using Decomposition

Considering the decomposition of LSP routes in Maximum Clean Segment (MCS), the matching "problem" can be solved using an approximation algorithm. Approximation algorithm are heuristics that guarantees a performance in a worse-case scenario. Existing approximation algorithms for well-known problems, such as Knapsack Problem and Maximum Cut, could be considered as a base for the Label Space Reduction problem.

Matching by Column Generation Methods

The simulations using the Decompose and Match framework were performed over networks using up to 24 LSRs and 800 LSPs. The running time of the framework was acceptable for daily network management. Taking into account the ratio between the number of LSPs and LSRs, it could be desirable to consider more LSPs in the simulations. However, the framework is limited by the ILP model in special.

The ILP size grows according to the number of feasible AMTs, and LSPs in the network as well. Considering column generation, an ILP model can be developed so it includes one column (one feasible matching) at a time.

5.2.2 New Trade-off: Propagation Time vs. Label Space Reduction

Creating AMTs, or tunneling in general, could expose a trade-off when protection and/or restoration mechanisms are considered (see §1.3.2). Mainly, the larger the tunnel (hence the greater the reduction), the larger the fault propagation time.

5.2.3 Problem Analysis: AMTs NP-Completeness Proof

Even though the decision problems for AMTs were clearly exposed, an NP-Complete proof was not stated. An NP-complete proof helps to relate the problem with other well-known problems in computer science. Therefore, both better heuristics and approximation algorithms could be developed.

5.2.4 Extending the Possibilities: Is It Worth Pushing Twice?

The present study considers pushing only one label. It could be interesting to measure how much can the label space be reduced by pushing another label.

5.2.5 The Effects of Existing Routing Algorithms over LaSpaRed

The routes obey to QoS parameters. Routes were computed in the first part using an evolutionary algorithm considering a bunch of metrics at a time.

It could be appropriate to analyze how classical routing algorithms (such as CSPF, MIRA, etc.) affect the performance of the label space reduction methods.

5.2.6 Label Space Reduction over a Virtual Topology

The analysis of the G^+ architecture using an OPS was not completely addressed. In §3, the LASPARED in a physical optical networks was explored. In §4, the virtual topology problem using G^+ was addressed. A better reduction of labels can be addressed if a virtual topology is considered instead of a physical one.

The use of a virtual topology implies a new trade-off: contention resolution. As mentioned before, optical packet may compete in an OPS for a wavelength port at the same time. The usual method to resolve this is by forwarding one of the packet using another wavelength embraced in the same output fiber. Clearly, with the use of a WRS not any wavelength can be picked for contention resolution. This is because some of the wavelengths are cross-connected downstream in order to head to a different destinations.

Under the WDM perspective, optical routes must be conceived as virtual links over a virtual topology. On the other hand, under the OPS perspective, in order to address packet contentions, several wavelengths are needed for connecting two neighboring OPS. A juxtaposition of both technologies leads to consider several optical routes connecting any pair of nodes in the (virtual) network. This is, any pair of nodes in the network should be interconnected by several optical routes or not be interconnected at all.

Bibliography

[ABG+01] Daniel Awduche, Lou Berger, Der-Hwa Gan, Tony Li, Vijay Srinivasan, and George Swallow. *Extensions to RSVP for LSP Tunnels*. IETF, December 2001. RFC 3209.

[AMA+99] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. *Requirements for Traffic Engineering Over MPLS*. IETF, September 1999. RFC 2702.

[AS03] L. Andersson and G. Swallow. *The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols*. IETF, February 2003. RFC 3468.

[AT03] David Applegate and Michel Thorup. Load optimal MPLS routing with N+M labels. In *Proc. IEEE INFOCOM 2003*, pages 555–565, 2003.

[BGN02] Sudeept Bhatnagar, Samrat Ganguly, and Badri Nath. Label space reduction in MultiPoint-to-Point LSPs for traffic engineering. In *Proc. IEEE European Conference on Universal Multiservice Networks (ECUMN 2002)*, pages 29–35, April 2002.

[BGN03] Sudeept Bhatnagar, Samrat Ganguly, and Badri Nath. Creating Multipoint-to-Point LSPs for traffic engineering. In *Proc. IEEE Workshop on High Performance Switching and Routing (HPSR 2003)*, pages 201–207, June 2003.

[BGN05] Sudeept Bhatnagar, Samrat Ganguly, and Badri Nath. Creating Multipoint-to-Point LSPs for traffic engineering. *IEEE Commun. Mag.*, 43(1):95–100, January 2005.

[BM00] Dhritiman Banerjee and Biswanath Mukherjee. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Trans. Netw.*, 8(5):598–607, October 2000.

[BOR+00] Daniel Blumenthal, Bengt-Erik Olsson, Giammarco Rossi, Timothy Dimmick, Lavanya Rau, Milan Masanovic, Olga Lavrova, Roopesh Doshi, Olivier Jerphagnon, John Bowers, Volkan Kaman, Larry Coldren, and John Barton. All-optical label swapping networks and technologies. *J. Lightw. Technol.*, 18(12):2058–2075, December 2000.

[BPY+02] C. Bintjas, N. Pleros, K. Yiannopoulos, G. Theophilopoulos, M. Kalyvas, H. Avramopoulos, and G. Guekos. All-optical packet address and payload separation. *IEEE Photon. Technol. Lett.*, 14:1728–1730, 2002.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.* IETF, September 1997. RFC 2750.

[CCPD06] Ruth Van Caenegem, Didier Colle, Mario Pickavet, and Piet Demeester. Benefits of label stripping compared to label swapping from the point of node dimensioning. *Photonic Network Communications Journal*, 12(3):227–244, December 2006.

[CFF+04] A. Carena, V. De Feo, J.M. Finochietto, R. Gaudino, F. Neri, C. Piglione, and P. Poggiolini. RingO: An experimental WDM optical packet network for metro applications. *IEEE J. Sel. Areas Commun.*, 22(8):1561–1571, October 2004.

[CMC+06] Ruth Van Caenegem, Jose M Martinez, Didier Colle, Mario Pickavet, Piet Demeester, Francisco Ramos, and Javier Marti. From IP over WDM to all-optical packet switching: Economical overview. *J. Lightw. Technol.*, 24(4):1638–1645, April 2006.

[DFM04] Yezid Donoso, Ramon Fabregat, and Jose Marzo. A multi-objective optimization scheme for multicast routing: A multitree approach. *Telecommunication Systems Journal*, 27(2–4):229–251, October 2004. Special issue: Networks.

[DHL+03] H.J.S. Dorren, M.T. Hill, Y. Liu, N. Calabretta, A. Srivatsa, F.M. Huijskens, H. de Waardt, and G.D. Khoe. Optical packet switching and buffering by using all-optical signal processing methods. *J. Lightw. Technol.*, 21(1):2–12, January 2003.

[Dix03] Sudhir Dixit. *IP over WDM: Building the Next Generation Optical Internet.* Wiley-Interscience, March 2003.

[GJ02] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 2002.

[GKR03] Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Exploring the trade-off between label size and stack depth in MPLS routing. In *Proc. IEEE INFOCOM 2003*, 2003.

[GKR05] Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a pez dispenser (or, routing issues in MPLS). *SIAM Journal on Computing*, 34(2), 2005.

[GKT03] Anupam Gupta, Amit Kumar, and Michel Thorup. Tree based MPLS routing. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA 2003)*, 2003.

[GZ03] Ashwin Gumaste and Si Qing Zheng. Next-generation optical storage area networks: The light-trails approach. *IEEE Commun. Mag.*, 43(3):72–79, March 2003.

[JACD02] B. Jamoussi, L. Andersson, R. Collon, and R. Dantu. *Constraint-Based LSP Setup using LDP*. IETF, January 2002. RFC 3212.

[KR05] K. Kompella and Y. Rekhter. *Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)*. IETF, October 2005. RFC 4206.

[Man04] E. Mannie. *Generalized Multi-Protocol Label Switching (GMPLS) Architecture*. IETF, October 2004. RFC 3945.

[MCE+00] B. Meagher, G.K. Chang, G. Ellinas, Y.M. Lin, W. Xin, T.F. Chen, X. Yang, A. Chowdhury, J. Young, S.J. Yoo, C. Lee, M.Z. Iqbal, T.Robe, H. Dai, Y.J. Chen, and W.I. Way. Design and implementation of ultra-low latency optical label switching for packet-switched wdm networks. *J. Lightw. Technol.*, 18(12):1978–1987, December 2000.

[MCL+02] S. De Maesschalck, Didier Colle, I. Lievens, Mario Pickavet, and Pitier Demeester. Pan-european optical transport networks: an availability-based comparison. *Photonic Network Communications Journal*, 5(3):203–225, May 2002.

[MCS+07] Jose Marzo, Luis Caro, Fernando Solano, Jaudelice de Oliveira, and Ramon Fabregat. Operational cost reduction in WDM networks using lighttours. In *Proc. IEEE International Conference on Transparent Optical Networks (ICTON 2007)*, July 2007. Invited Paper.

[ML01] Eytan Modiano and Philip J. Lin. Traffic grooming in WDM networks. *IEEE Commun. Mag.*, 39(7):124–129, July 2001.

[Moy98] J. Moy. *OSPF Version 2*. IETF, April 1998. RFC 2328.

[MRM+02] J.M. Martinez, F. Ramos, J. Marti, J. Herrera, and R. Llorente. All-optical N-bit XOR gate with feedback for optical packet header processing. In *Eur. Conf. Optical Communication (ECOC)*, volume 3, 2002. Poster session.

[MSdO+06] Jose Marzo, Fernando Solano, Jaudelice de Oliveira, Luis Caro, and Ramon Fabregat. Optimal traffic routing in WDM using lighttours. In *Proc. IEEE International Conference on Transparent Optical Networks (ICTON 2006)*, June 2006. Invited Paper.

[Muk97] Biswanath Mukherjee. *Optical Communication Networks*. McGray-Hill, 1997.

[NP04] Constantinos Neophytou and Chris Phillips. A scheme for the dynamic formation of robust multipoint to point LSPs. In *Proc. IEEE Consumer Communications and Networking Conference (CCNC 2004)*, pages 251–255, January 2004.

[Ora90] D. Oran. *OSI IS-IS Intra-domain Routing Protocol*. IETF, February 1990. RFC 1192.

[PSA05] P. Pan, G. Swallow, and A. Atlas. *Fast Reroute Extensions to RSVP-TE for LSP Tunnels*. IETF, May 2005. RFC 4090.

[RKM+05] Francisco Ramos, E. Kehayas, Jose M Martinez, R. Clavero, Javier Marti, L. Stampoulidis, D. Tsiokos, H. Avramopoulos, J. Zhang, P. V. Holm-Nielsen, N. Chi, P. Jeppesen, N. Yan, I. Tafur Monroy, A.M.J. Koonen, M.T. Hill, Y. Liu, H.J.S. Dorren, Ruth Van Caenegem, Didier Colle, Mario Pickavet, and B. Riposati. IST-LASAGNE: Towards all-optical label swapping employing optical logic gates and optical flip-flops. *IEEE J. Sel. Areas Commun.*, 23(10):2993–3011, October 2005.

[RL95] Y. Rekhter and T. Li. *A Border Gateway Protocol (BGP-4)*. IETF, March 1995. RFC 1771.

[Ros01] Eric Rosen. *MPLS Label Stack Encoding*. IETF, January 2001. RFC 3032.

[RVC01] Eric Rosen, Arun Viswanathan, and Ross Callon. *Multiprotocol Label Switching Architecture*. IETF, January 2001. RFC 3031.

[SCC+07a] Fernando Solano, Ruth Van Caenegem, Didier Colle, Jose Marzo, and Ramon Fabregat. Label merging in all-optical label swapping networks. In *Proc. Workshop in G/MPLS Networks*, April 2007. Invited Paper.

[SCC+07b] Fernando Solano, Ruth Van Caenegem, Didier Colle, Mario Pickavet, Jose Marzo, and Ramon Fabregat. Routing and label stacking in all-optical label swapping networks. In *Proc. European Conference in Networks and Optical Communications (NOC 2007)*, June 2007. Invited Paper.

[SCdO+07] Fernando Solano, Luis Caro, Jaudelice de Oliveira, Ramon Fabregat, and Jose Marzo. G^+: Enhanced traffic grooming in WDM mesh networks using Lighttours. *IEEE J. Sel. Areas Commun.*, 25(5):1034–1047, June 2007.

[SCF+06] Fernando Solano, Luis Caro, Ramon Fabregat, Jose Marzo, and Thomas Stidsen. Enhancing traffic grooming in WDM networks through lambda-monitoring. In *Proc. INFORMS Telecommunications Congress 2006*, pages 77–80, April 2006. As part of the COST 293 project.

[SFDM04] Fernando Solano, Ramon Fabregat, Yezid Donoso, and Jose Marzo. Mapping sub-flows to P2MP LSPs. In *Proc. IEEE IP Operations and Management (IPOM 2004)*, October 2004.

[SFDM05a] Fernando Solano, Ramon Fabregat, Yezid Donoso, and Jose Marzo. Asymmetric tunnels in P2MP LSPs as a label space reduction method. In *Proc. IEEE International Conference on Communications (ICC 2005)*, pages 43–47, May 2005.

[SFDM05b] Fernando Solano, Ramon Fabregat, Yezid Donoso, and Jose Marzo. A label space reduction method for P2MP LSPs using asymmetric tunnels. In *Proc. IEEE International Symposium on Computers and Communications (ISCC 2005)*, pages 746–751, June 2005.

[SFFF03] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power-laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524, August 2003.

[SFM05a] Fernando Solano, Ramon Fabregat, and Jose Marzo. A fast algorithm based on the MPLS label stack for the label space reduction problem. In *Proc. IEEE IP Operations and Management (IPOM 2005)*, October 2005.

[SFM05b] Fernando Solano, Ramon Fabregat, and Jose Marzo. Full label space reduction in MPLS networks: Asymmetric merged tunneling. *IEEE Commun. Lett.*, 9(11):1021–1023, November 2005.

[SFM07] Fernando Solano, Ramon Fabregat, and Jose Marzo. On optimal computation of MPLS label binding for MultiPoint-to-Point connections. *IEEE Trans. Commun.*, 2007. Accepted for publication.

[SMWA04] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1):2–16, February 2004.

[SMY00] Hiroyuki Saito, Yasuhiro Miyao, and Makiko Yoshida. Traffic engineering using multiple MultiPoint-to-Point LSPs. In *Proc. IEEE INFOCOM 2000*, pages 894–901, 2000.

[Som06] Arun Somani. *Survivability and Traffic Grooming in WDM Optical Networks*. Cambridge University Press, 2006.

[SSFM07] Fernando Solano, Thomas Stidsen, Ramon Fabregat, and Jose Marzo. Label space reduction in MPLS networks: How much can one label do? *IEEE/ACM Trans. Netw.*, 2007. Accepted for publication.

[SSM06] Narendra K. Singhal, Laxman H. Sahasrabuddhe, and Biswanath Mukherjee. Optimal multicasting of multiple light-trees of different bandwidth granularities in a WDM mesh network with sparse splitting capabilities. *IEEE/ACM Trans. Netw.*, 14(5):1104–1117, October 2006.

[Swa99] George Swallow. MPLS advantages for traffic engineering. *IEEE Commun. Mag.*, 37(12), December 1999.

[ZJM00a] H. Zang, J. Jue, and B. Mukherjeee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine*, January 2000.

[ZJM00b] Hui Zang, Jason P. Jue, and Biswanath Mukherjee. Capacity allocation and contention resolution in a photonic slot routing all-optical WDM mesh network. *J. Lightw. Technol.*, 18(12):1728–1741, December 2000.

[ZM02] Keyao Zhu and Biswanath Mukherjee. Traffic grooming in an optical WDM mesh network. *IEEE J. Sel. Areas Commun.*, 20(1):122–133, January 2002.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithm: A comparative case study and the strength pareto approach. *IEEE Trans. Evol. Comput.*, 3(4):257–271, November 1999.

[ZZM03] Keyao Zhu, Hui Zang, and Biswanath Mukherjee. A comprehensive study on next-generation optical grooming switches. *IEEE J. Sel. Areas Commun.*, 21(7):1173–1186, September 2003.