Universitat Politècnica de Catalunya
Optical Communications Group

In-Operation Planning in Flexgrid Optical Core Networks

Lluís Gifre

Advisors:
Dr. Luis Velasco Esteban
Dr. Ignacio Navarro Mas

A thesis presented in partial fulfillment of the requirements for the degree of

Philosophy Doctor

June 3, 2016

Optical Communications Group (GCO)

Universitat Politècnica de Catalunya (UPC)

C/ Jordi Girona, 1-3

Campus Nord, D6-107

08034 Barcelona, Spain

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola de Doctorat

Curso académico: 2015/2016

Acta de calificación de tesis doctoral

Nombre y apellidos
Lluis Gifre Renom

Programa de doctorado
Computers Architecture

Unidad estructural responsable del programa
Department of Computers Architecture

Resolución del Tribunal

Reunido el Tribunal designado a tal efecto, el doctorando / la doctoranda expone el tema de la su tesis doctoral titulada _____

_____.

Acabada la lectura y después de dar respuesta a las cuestiones formuladas por los miembros titulares del tribunal, éste otorga la calificación:

☐ NO APTO ☐ APROBADO ☐ NOTABLE ☐ SOBRESALIENTE

(Nombre, apellidos y firma) Presidente/a	(Nombre, apellidos y firma) Secretario/a	
(Nombre, apellidos y firma) Vocal	(Nombre, apellidos y firma) Vocal	(Nombre, apellidos y firma) Vocal

_____, _____ de _____ de _____

El resultado del escrutinio de los votos emitidos por los miembros titulares del tribunal, efectuado por la Escuela de Doctorado, a instancia de la Comisión de Doctorado de la UPC, otorga la MENCIÓN CUM LAUDE:

☐ SÍ ☐ NO

(Nombre, apellidos y firma) Presidente de la Comisión Permanente de la Escuela de Doctorado	(Nombre, apellidos y firma) Secretaria de la Comisión Permanente de la Escuela de Doctorado

Barcelona a _____ de _____ de _____

Agraïments

Voldria començar donant les gràcies al meu supervisor de tesis, el professor Luis Velasco, qui em va introduir en el mon de la investigació, em va encoratjar a començar el doctorat, m'ha donat suport tots aquests anys i m'ha fet confiança des del primer dia. També m'ha permès tractar de primera mà amb experts de tot el mon, tant en estades de recerca a universitats de primera línia com en els projectes d'investigació als que he pogut col·laborar.

Vull també recordar a meu ex-professor i co-supervisor de la tesis, el professor Ignacio Navarro qui per desgràcia, ens va deixar pocs mesos abans de poder finalitzar aquesta tesis. Ell sempre va mirar d'ajudar-me en el què va poder i em va proporcionar els recursos que va tenir al seu abast.

També voldria agrair molt a les companyes i companys del Grup de Comunicacions Òptiques (GCO) que sempre han estat disposats a ajudar-me quan ha calgut. El tracte acadèmic i personal amb totes elles i tots ells ha estat una experiència inoblidable que, per descomptat, ha facilitat molt el camí. Ha estat un privilegi poder col·laborar amb totes i tots vosaltres.

Moltes gràcies també a les meves germanes i amics per ser sempre allà, però en especial als meus pares, qui sempre han mirat primer per la meva educació i la de les meves germanes abans que per ells mateixos. Indubtablement, el vostre sacrifici ha estat l'origen de tot aquest camí.

Sols em queda reiterar-me en agrair-vos a totes i tots el suport i ànims que m'heu donat. M'agrada pensar que, d'una forma o altre, tots heu posat el vostre gra de sorra en aquesta tesis.

Abstract

New generation applications, such as cloud computing or video distribution, can run in a telecom cloud infrastructure where the datacenters (DCs) of telecom operators are integrated in their networks thus, increasing connections' dynamicity and resulting in time-varying traffic capacities, which might also entail changes in the traffic direction along the day.

As a result, a flexible optical technology able to dynamically set-up variable-capacity connections, such as flexgrid, is needed. Nonetheless, network dynamicity might entail network performance degradation thus, requiring re-optimizing the network while it is in operation. This thesis is devoted to devise new algorithms to solve in-operation network planning problems aiming at enhancing the performance of optical networks and at studying their feasibility in experimental environments.

In-operation network planning requires from an architecture enabling the deployment of algorithms that must be solved in stringent times. That architecture can be based on a Path Computation Element (PCE) or a Software Defined Networks controller. In this thesis, we assume the former split in a front-end PCE, in charge of provisioning paths and handling network events, and a specialized planning tool in the form of a back-end PCE responsible for solving in-operation planning problems.

After the architecture to support in-operation planning is assessed, we focus on studying the following applications:

1) Spectrum fragmentation is one of the most important problems in optical networks. To alleviate it to some extent without traffic disruption, we propose a hitless spectrum defragmentation strategy.
2) Each connection affected by a failure can be recovered using multiple paths to increase traffic restorability at the cost of poor resource utilization. We propose re-optimizing the network after repairing the failure to aggregate and reroute those connections to release spectral resources.
3) We study two approaches to provide multicast services: establishing a point-to-multipoint connections at the optical layer and using multi-purpose

virtual network topologies (VNT) to serve both unicast and multicast connectivity requests.

4) The telecom cloud infrastructure, enables placing contents closer to the users. Based on it, we propose a hierarchical content distribution architecture where VNTs permanently interconnect core DCs and metro DCs periodically synchronize contents to the core DCs.

5) When the capacity of the optical backbone network becomes exhausted, we propose using a planning tool with access to *inventory* and *operation* databases to periodically decide the equipment and connectivity to be installed at the minimum cost reducing capacity overprovisioning.

6) In multi-domain multi-operator scenarios, a broker on top of the optical domains can provision multi-domain connections. We propose performing intra-domain spectrum defragmentation when no contiguous spectrum can be found for a new connection request.

7) Packet nodes belonging to a VNT can collect and send incoming traffic monitoring data to a big data repository. We propose using the collected data to predict next period traffic and to adapt the VNT to future conditions.

The methodology followed in this thesis consists in proposing a problem statement and/or a mathematical formulation for the problems identified and then, devising algorithms for solving them. Those algorithms are simulated and then, they are experimentally assessed in real test-beds.

This thesis demonstrates the feasibility of performing in-operation planning in optical networks, shows that it enhances the performance of the network and validates the feasibility of its deployment in real networks.

It shall be mentioned that part of the work reported in this thesis has been done within the framework of several research projects, namely IDEALIST (FP7-ICT-2011-8) and GEANT (238875) funded by the EC and SYNERGY (TEC2014-59995-R) funded by the MINECO.

Resum

Les aplicacions de nova generació, com ara el cloud computing o la distribució de vídeo, es poden executar a infraestructures de telecom cloud (TCI) on operadors integren els seus datacenters (DC) a les seves xarxes. Aquestes aplicacions fan que incrementi tant la dinamicitat de les connexions, com la variabilitat de les seves capacitats en el temps, arribant a canviar de direcció al llarg del dia.

Llavors, cal disposar de tecnologies òptiques flexibles, tals com flexgrid, que suportin aquesta dinamicitat a les connexions. Aquesta dinamicitat pot degradar el rendiment de la xarxa, obligant a re-optimitzar-la mentre és en operació. Aquesta tesis està dedicada a idear nous algorismes per a resoldre problemes de planificació sobre xarxes en operació (*in-operation network planning*) per millorar el rendiment de les xarxes òptiques i a estudiar la seva factibilitat en entorns experimentals.

Aquests problemes requereixen d'una arquitectura que permeti desplegar algorismes que donin solucions en temps restrictius. L'arquitectura pot estar basada en un Element de Computació de Rutes (PCE) o en un controlador de Xarxes Definides per Software. En aquesta tesis, assumim un PCE principal encarregat d'aprovisionar rutes i gestionar esdeveniments de la xarxa, i una eina de planificació especialitzada en forma de PCE de suport per resoldre problemes d'in-operation planning.

Un cop validada l'arquitectura que dona suport a in-operation planning, estudiarem les següents aplicacions:

1) La fragmentació d'espectre és un dels principals problemes a les xarxes òptiques. Proposem reduir-la en certa mesura, fent servir una estratègia que no afecta al tràfic durant la desfragmentació.
2) Cada connexió afectada per una fallada pot ser recuperada fent servir múltiples rutes incrementant la restaurabilitat de la xarxa, tot i empitjorar-ne la utilització de recursos. Proposem re-optimitzar la xarxa després de reparar una fallada per agregar i re-enrutar aquestes connexions tractant d'alliberar recursos espectrals.
3) Estudiem dues solucions per aprovisionar serveis multicast: establir connexions punt-a-multipunt sobre la xarxa òptica i utilitzar Virtual

Network Topologies (VNT) multi-propòsit per a servir peticions de connectivitat tant unicast com multicast.

4) La TCI permet mantenir els continguts a prop dels usuaris. Proposem una arquitectura jeràrquica de distribució de continguts basada en la TCI, on els DC principals s'interconnecten per mitjà de VNTs permanents i els DCs metropolitans periòdicament sincronitzen continguts amb els principals.

5) Quan la capacitat de la xarxa òptica s'exhaureix, proposem utilitzar una eina de planificació amb accés a bases de dades d'inventari i operacionals per decidir periòdicament l'equipament i connectivitats a instal·lar al mínim cost i reduir el sobre-aprovisionament de capacitat.

6) En entorns multi-domini multi-operador, un broker per sobre dels dominis òptics pot aprovisionar connexions multi-domini. Proposem aplicar desfragmentació d'espectre intra-domini quan no es pot trobar espectre contigu per a noves peticions de connexió.

7) Els nodes d'una VNT poden recollir i enviar informació de monitorització de tràfic entrant a un repositori de big data. Proposem utilitzar aquesta informació per adaptar la VNT per a futures condicions.

La metodologia que hem seguit en aquesta tesis consisteix en formalitzar matemàticament els problemes un cop aquests son identificats i, després, idear algorismes per a resoldre'ls. Aquests algorismes son simulats i finalment validats experimentalment en entorns reals.

Aquesta tesis demostra la factibilitat d'implementar mecanismes d'in-operation planning en xarxes òptiques, mostra els beneficis que aquests aporten i valida la seva aplicabilitat en xarxes reals.

Part del treball presentat en aquesta tesis ha estat dut a terme en el marc dels projectes de recerca IDEALIST (FP7-ICT-2011-8) i GEANT (238875), finançats per la CE, i SYNERGY (TEC2014-59995-R), finançat per el MINECO.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The flexgrid optical network technology has become the *de-facto* candidate for future optical networks because of its inherent spectrum efficiency and connection flexibility [G.694.1]. Nonetheless, to ensure that the network can support the forecast traffic and all the failure scenarios that need to be protected against, operators add spare capacity (overprovision) in different parts of the network to address likely future scenarios.

Today, the network capacity planning process is typically an offline process, and is based on very long planning cycles (e.g. yearly, quarterly). Generally, this is due to the static and inflexible nature of the current networks. This entails inefficient use of network resources and significantly increases network capital expenditures (CAPEX). In addition, due to the fixed and rigid nature of provisioning in the transport layer, network planning and traffic engineering (TE) still require significant human intervention, which entails high operational expenditures (OPEX).

Notwithstanding, optical transport platforms are designed to facilitate the setting up and tearing down of optical connections (lightpaths) within minutes or even seconds. The optical layer might use automated TE techniques to set-up connections through routes that use currently available resources. Combining remotely configurable optical cross-connects (OXCs) with a control plane provides the capability of automated lightpath setup for regular provisioning and real-time reaction to failures thus, being able to reduce OPEX. However, to exploit existing capacity, increase dynamicity, and provide automation in future networks, current management architectures, utilizing legacy network management systems (NMSs), need to be radically transformed.

In a scenario where lightpath provisioning can be automated, network resources can be made available by reconfiguring and/or re-optimizing the network on demand and in real time aiming at improving the network's Grade of Service (GoS); this was called as *in-operation network planning* in [Ve14-1].

In this PhD thesis, we study both the feasibility of performing in-operation planning in optical networks and propose several applications of in-operation planning that take advantage of novel reconfiguration capabilities and new management architectures.

The work here presented has been developed within the Optical Communications Group and continues, in part, the work developed during the PhD thesis of Dr. Alberto Castro [Cast14-3].

1.2 Goals of the PhD thesis

The main objective of this thesis is to demonstrate the feasibility of in-operation planning in flexgrid optical networks. Two specific goals are defined to achieve this main goal:

G.1 – Architectures for in-operation planning

- To study architectures supporting in-operation planning.

G.2 – Applications of in-operation planning in flexgrid-based networks

This goal is divided into several sub-goals:

- **G.2.1**: To study the hitless spectrum defragmentation problem using spectrum shifting.
- **G.2.2**: To study the after failure repair optimization problem using multiple paths.
- **G.2.3**: To study point-to-multipoint connectivity.
- **G.2.4**: To study the feasibility of distributing contents in flexgrid-supported telecom cloud.
- **G.2.5**: To extend the in-operation planning architecture to support inventory-based planning.
- **G.2.6**: To study multi-domain connectivity in a hierarchical broker-based architecture.
- **G.2.7**: To extend the in-operation planning architecture to support monitoring-based data analytics.

A summary of the goals is presented in Table 1-1.

Table 1-1 PhD Thesis goals

	Goals	
G1 **Feasibility of in-operation planning**	In-operation planning in a separate module and synchronize databases.	
G2 **Applications of in-operation network planning**	**G2.1**	Spectrum shifting.
	G2.2	After failure repair optimization using multiple paths.
	G2.3	Single- and multi-layer point-to-multipoint connectivity.
	G2.4	Content distribution in the telecom cloud.
	G2.5	Inventory-based planning.
	G2.6	Multi-domain connectivity.
	G2.7	Monitoring-based data analytics.

1.3 Methodology

During the development of this PhD thesis, we follow a methodology where in-operation planning applications are first formulated as mathematical problems. Next,algorithms are devised and their performance is evaluated through simulation and finally, their feasibility is assessed in an experimental environment. Fig. 1-1 presents the main steps in the methodology.

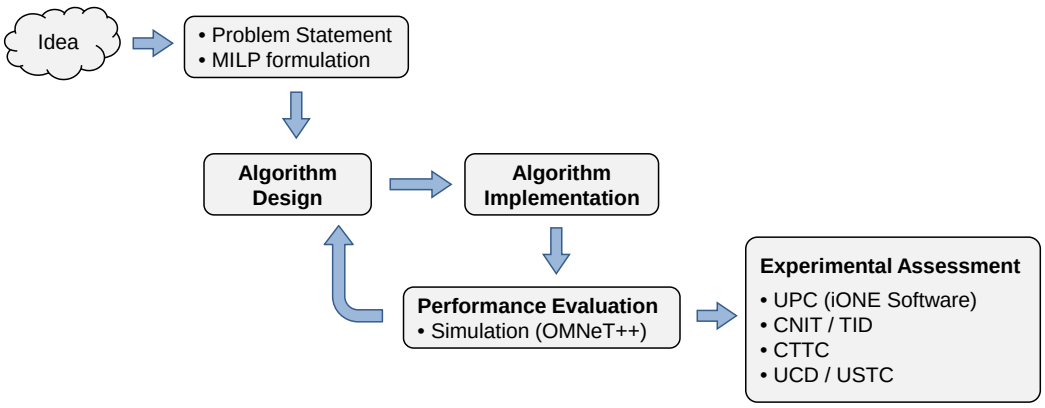


Fig. 1-1 Methodology used in this PhD thesis

The process starts when a new idea is conceived. The in-operation problem is first formally stated and a Mixed Integer Linear Program (MILP) formulation is designed. Because solving the MILP formulations would require long computation times, an exact or heuristic algorithm needs to be devised and implemented; algorithms are implemented in C++.

Comparison between solving the MILP problem using a commercial solver such as CPLEX [CPLEX] and running the algorithm is performed in terms of quality of the solutions and computation times.

Next, the performance of the algorithms are evaluated through simulation; we used the event-driven OMNeT++ simulation framework [OMNET]. Finally, once the algorithm has been validated, it is experimentally assessed in a real distributed testbed, usually involving third-party collaboration from other national and international institutions.

1.4 Thesis Outline

The remainder of this thesis is as follows.

Chapter 2 briefly introduces the main background of flexgrid optical networks, including the control and management planes, in particular focusing on in-operation planning.

Chapter 3 explores the state-of-the-art of in-operation planning for optical core networks and its applications.

Chapter 4 introduces our contribution to an architecture for in-operation planning (goal G1) assessing the separation of the module for in-operation planning and validating the synchronization of their local databases. The iONE module architecture used along this thesis to implement the Applications-based Network Operations (ABNO) components and a use case to illustrate its configuration are provided. This chapter is based on [Gi14-3], [Mar14-1], [Go15-1], [Ve15-1] and [Go16-1].

Chapter 5 applies spectrum shifting for hitless spectrum defragmentation (goal G2.1). This chapter is based on [Gi14-2].

Chapter 6 explores an after failure repair optimization mechanism using multiple paths (goal G2.2). This chapter is based on [Ve14-5] and [Gi15-2].

Chapter 7 presents our contribution to single- and multi-layer approaches for point to multi-point connectivity (goal G2.3). This chapter is based on [Gi14-1] and [Gi15-1].

Chapter 8 devises a new architecture for hierarchical content distribution in the telecom cloud (goal G2.4). This chapter is based on [Gi16-1].

Chapter 9 explores an inventory-based planning mechanism to periodically increase the capacity of the network (goal G2.5). This chapter is based on [Ve16-1].

Chapter 10 introduces our contribution regarding multi-domain connectivity services with per-domain spectrum defragmentation (goal G2.6) and monitoring-based data analytics techniques to reconfigure virtual network topologies (goal G2.7). This chapter is based on [Cast15-1] and [Gi16-2], respectively.

Finally, Chapter 11 draws the main conclusions of the thesis.

Chapter 2

Background

As a starting point, let us introduce the concepts and technologies used along this thesis. Firstly, the telecom cloud is presented; it is an infrastructure that considers both, the telecom network and the datacenters (DCs) interconnected through this network. The huge capacity and dynamicity of the required connections motivates the need of deploying flexible optical networks to interconnect DCs in a telecom cloud.

Next, the main concepts and benefits related to flexgrid-based optical networks are introduced. In particular, we use the graph theory to model optical networks, describe the type of connections that can be set-up in an optical network, and the way that their route and spectrum allocation is specified. Because of the mismatch between the capacity of demands and that of optical connections, we introduce the concept of multi-layer networks, where connections are established on the Internet Protocol (IP) / Multi-Protocol Label Switching (MPLS) layer through virtual links supported by the optical layer.

Aiming at automatically operating an optical network, a flexible control and management planes needs to be deployed. In particular, a recently standardized architecture where components have well-defined functionalities and interfaces is presented and the protocols enabling components' inter-operation are reviewed.

Finally, since dynamic operation of optical networks results in non-optimal use of resources, in-operation planning is introduced to re-optimize the network and improve its resource utilization in real time. The network's life cycle is extended with in-operation planning and the desired requirements for in-operation planning are presented.

2.1 Telecom Cloud

New generation applications, such as cloud applications and content distribution, usually entail establishing connections between geographically distant premises and DCs. These connections usually present high degrees of dynamicity in terms of variability in the capacity and directionality of the traffic, which changes daily as a result of users interaction with cloud applications during day periods, while backups are synchronized between DCs during night periods.

Telecom operators are in a vantage position to capitalize on highly-reliable cloud services since they own the infrastructure to support on-demand delivery of processing, storage, and network resources, which allow them developing a *telecom cloud* infrastructure [Ve15-2]. That connected cloud infrastructure, consisting of DCs and telecom networks, can provide highly-reliable ultra-low latency cloud services by placing computing resources closer to the end-users thus, increasing quality of experience.

Network operators can use the telecom cloud for many purposes such as to distribute live-TV [As15-1] or synchronizing replicated contents using any available approach including distributed databases. Telecom cloud infrastructures need to transfer massive amounts of data between DCs, thus the high-capacity connectivity provided by optical networks is needed. The delivery of distributed cloud services implies, not only the configuration of the per-DC cloud computing resources, but also the configuration of the networks providing the communications services.

2.2 Optical Networks

In view of the above constraints, in this section we introduce the basics of optical networks.

2.2.1 Overview

An optical network can be described as a graph $G(N, E)$, where N is the set of optical cross connects (OXCs), and E is the set of optical links connecting two OXCs. Fig. 2-1 presents an example of an optical network consisting of four OXCs and ten optical links.

OXC physical equipment consist of a set of slots where cards can be plugged in. Some of these cards contain optical ports, named as *transponders*, used to transform data flows in the electrical domain to signals in the optical domain, and vice versa. Two optical fiber links are usually set-up between pairs of OXCs, one connecting node A to B and the other one connecting node B to A.

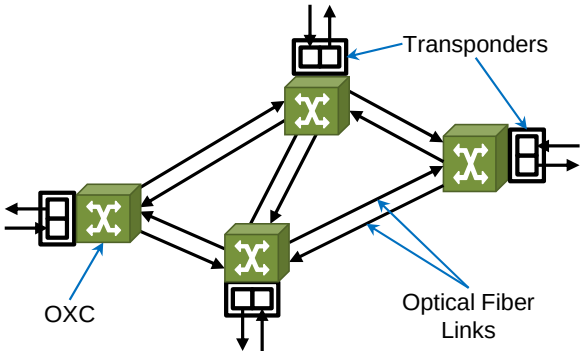


Fig. 2-1 Example of optical network.

Wavelength Division Multiplexing (WDM) networks use a fixed-size grid (e.g. 50 GHz) and allocate a single wavelength for every optical connection (*lightpath*).

In contrast, the emerging flexible grid technology (*flexgrid*) [Ji09-1], [Ji10-1], [Ger12-1], divides the optical spectrum into thinner equally-sized frequency slices (e.g., 12.5 GHz), and allows using a variable number of consecutive frequency slices to create arbitrary-sized frequency *slots* allocated to lightpaths [G.694.1].

The ability for creating arbitrary-sized lightpaths in flexgrid entails a better utilization of the optical spectrum [Wr13-1]. For instance, a 400 Gb/s connection can be served using 75 GHz of spectral width, while in a fixed grid technology four connections of 100 Gb/s, consuming 200 GHz in total, need to be established. Besides, flexgrid technology enables performing elastic operations over lightpaths, i.e. dynamically modifying the capacity of the lightpaths. This brings the possibility of adapting the traffic to the needs of each demand on every time period in terms of capacity.

When a lightpath needs to be set-up, the OXCs traversed are configured to switch the proper frequency slot from the incoming link to the outgoing link. In a flexgrid network, lightpaths must enforce two optical spectrum constrains: the *continuity constraint* and the *contiguity constraint*. The continuity constraint restricts lightpaths to use the same frequency slot in every link along the path, while the contiguity constraint ensures that the set of frequency slices in a frequency slot are contiguous in the spectrum.

As defined in [G.694.1], the frequency slot allocated to a lightpath is identified by the tuple <*n*, *m*> and it is used to tune the optical filters along the path.

- The *n* parameter is the index of the central frequency encoded as the number of central frequency granularity units (positive, negative, or 0) from the reference central frequency (193.1 THz). The nominal central frequency granularity has been defined as 6.25 GHz.
- The *m* parameter is the frequency slot width in terms of number of slices, each of 12.5 GHz.

Therefore, the frequency slot of an optical connection can be univocally computed using the following equations:

$$central\ \ frequency\ (\text{THz}) = 193.1 + n \times 0.00625 \tag{2.1}$$

$$slot\ width\ (\text{GHz}) = 12.5 \times m \tag{2.2}$$

To illustrate the use of the tuple <n, m>, Fig. 2-2 shows three connections allocated in an optical link. For instance, the connection in the middle has been allocated with a 4*12.5=50 GHz slot centered in 193.1-3*0.00625 THz.

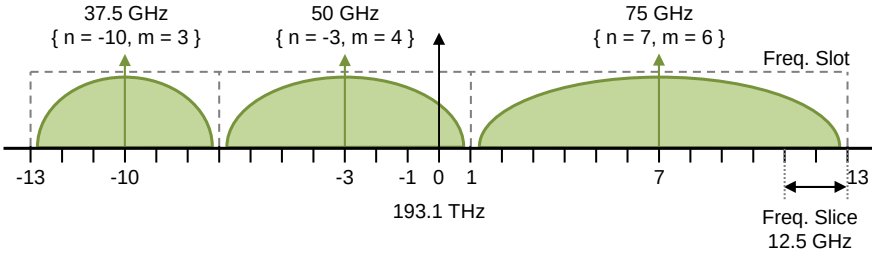


Fig. 2-2 Example of spectrum allocation.

2.2.2 Transponders

As previously stated, transponders are responsible for transforming electrical flows into optical signals and vice-versa. Therefore, they are used as end-points of lightpaths. Depending on their characteristics, different kinds of transponders are described in the literature. The selection of the kind of transponder is related to the characteristics of the lightpaths conveyed through the optical network [Ri12-1]. In optical networks where all lightpaths are of the same capacity, e.g. 100 Gb/s, Fixed Transponders (FT) can be used and just one type of transponder needs to be kept in the inventory. However, when lightpaths have different capacities, e.g. 40/100/400 Gb/s. FTs of different capacities have to be installed in the network nodes and different types of transponders have to be maintained in the inventory thus, significantly increasing network OPEX.

Moreover, since the capacity of demands and transponders must coincide, compatible transponders need to be available in the end nodes of that connection. Even though FT could be available in those nodes, if they are not of the required capacity, that lightpath could not be set-up.

As an alternative solution, Bandwidth-Variable Transponders (BVT) enhances FTs' characteristics by enabling dynamic configuration of their capacity and can be used in a range of capacities, e.g. 40/100/400 Gb/s. In such way, the inventory contains transponders of a single kind thus, notably reducing network OPEX.

However, BVTs are underutilized when lightpaths require lower capacities than that of the BVTs. For instance, a 400 Gb/s BVT can end a lightpath of 400 Gb/s at

the most; when lower capacities are requested, the residual capacity remains unused.

To increase the flexibility of BVTs and improve their utilization, a new kind of transponder, named as Sliceable BVT (SBVT), has been proposed in the literature [Ji12-1], [Nap15-1]. SBVTs are composed of a set of modules, each able to support an independent lightpath. Three constraints are imposed by SBVTs over the supported lightpaths: *i*) the total number of lightpaths cannot exceed the number of modules in the SBVT; *ii*) the total capacity of the supported lightpaths cannot exceed the total switching capacity of the SBVT; and *iii*) the spectrum allocated to each lightpath cannot overlap [Sam15-1]. For instance, an SBVT of 400 Gb/s with 4 internal modules can support four lightpaths of 100 Gb/s, or two lightpaths of 100 Gb/s plus a third one of 200 Gb/s, etc.

2.3 Connection Provisioning and Recovery

2.3.1 Connectivity at the Optical Layer

Connection provisioning is the process of establishing new optical connections in the network to serve incoming demands. Specifically for flexgrid-based networks, it consists in finding a physical route and a frequency slot, while enforcing both the continuity and contiguity constraints.

To find the optimal route and frequency slot, the Routing and Spectrum Allocation (RSA) problem, defined in [Ve14-2], must be solved for each incoming demand. Parameters n and m are determined according to the requested bitrate, as well as to other parameters, such as the distance. Fig. 2-3 depicts three lightpaths set-up on a network. Lightpath P1 (from X4 to X1) has been set-up through a four-slice slot, P2 (from X1 to X2) has been set-up through an eight-slice slot, and P3 (from X3 to X4) has been set-up through a six-slice slot.

Lightpaths allow connecting a source node with a single destination node (point-to-point, p2p connection). In contrast, when a source node needs to transfer the same data to a set of destination nodes (point-to-multipoint, p2mp connection), *light-trees* can be set-up thus, reducing the total amount of optical resources required as shown in Fig. 2-4. For instance, establishing a p2mp connection from X1 to X2 and X4 using two connections requires 32 frequency slices in total (8 slices along 4 hops), whereas using a light-tree requires only 24 frequency slices in total (8 slices along 3 hops).

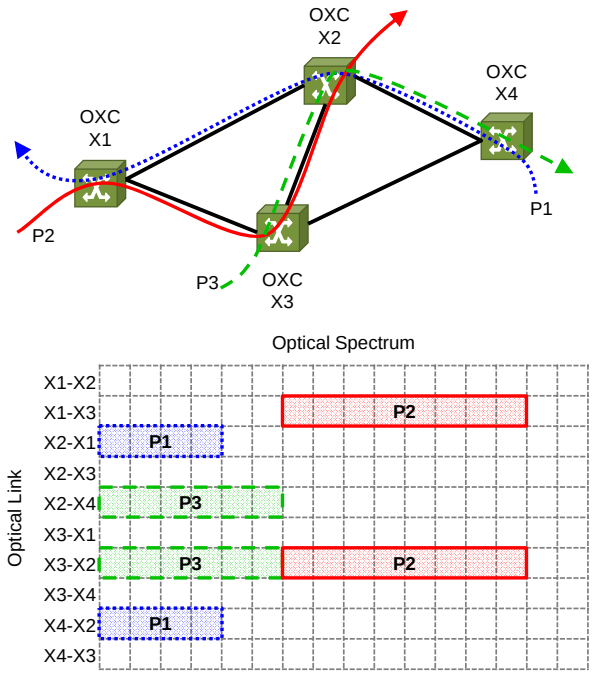


Fig. 2-3 Example 3 lightpaths set-up on a flexgrid network.

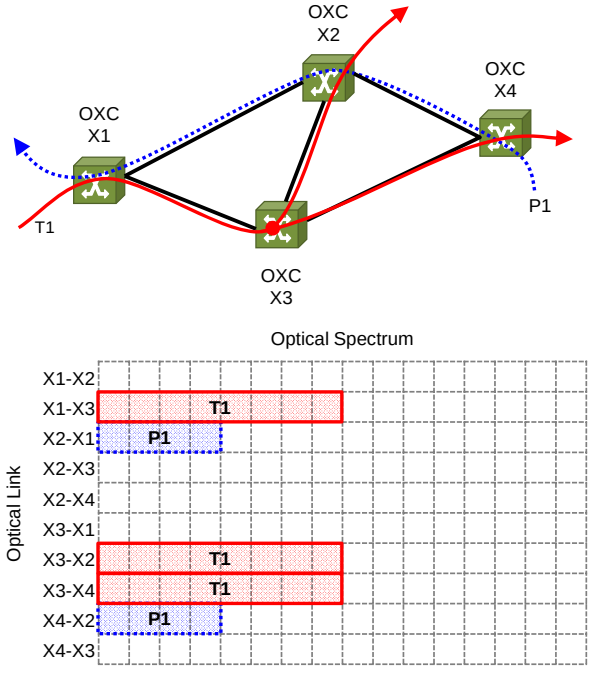


Fig. 2-4 Example of a lightpath and a light-tree set-up on a flexgrid network.

Finally, let us focus on the content distribution scenario where a set of DCs store copies of the same content. When a user needs to access that content (imagine large

size files), an *anycast* connection can be established between the user and any DC. It is worth noting that, in the context of optical networks, not only the route, but also the spectrum allocation fulfilling the continuity and contiguity constraints have to be computed. Fig. 2-5 illustrates an example of anycast provisioning where a content is replicated in three different DCs.

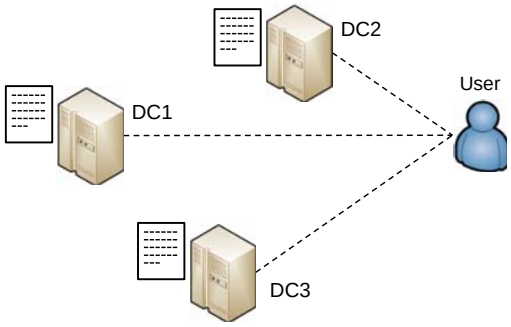


Fig. 2-5 Example of anycast provisioning.

2.3.2 Connectivity on Multi-layer Networks

Transport networks are designed to convey lightpaths with huge capacities, e.g. tenths or hundreds of Gb/s, which might be much higher than that required by many applications. To cope with that mismatch in demands' capacity, multi-layer networks can be deployed. A multi-layer network consists of two or more stacked networks; each of these stacked networks is named as a network layer.

The upper network layer consists of a set of nodes interconnected by means of *virtual links* (vlinks) supported by connections established in the immediately lower layer. Then, connections in one layer can be served through the topology created by that vlinks thus, creating a so-called Virtual Network Topology (VNT).

Fig. 2-6 shows an example of a multi-layer network where the upper layer, the IP/MPLS layer, is composed of a set of IP/MPLS routers connected by vlinks, whereas the Flexgrid optical layer consists of a set of OXCs and optical links. Vlinks in the IP/MPLS layer are supported by lightpaths in the optical layer.

Lightpaths are also named as Lambda Switch Capable (LSC) Label Switched Paths (LSP), whereas IP/MPLS flows are also known as Packet Switched Capable (PSC) LSPs.

In Fig. 2-6, routers R1, R3, R4 and R5 have been connected through a VNT supported by vlinks R1-R3, R1-R4 and R4-R5; vlink R1-R4, for instance, is supported by a lightpath crossing nodes X1, X2 and X4; its optical signal is originated in a transponder in R1 and it is converted back to the electrical domain in a transponder in R4

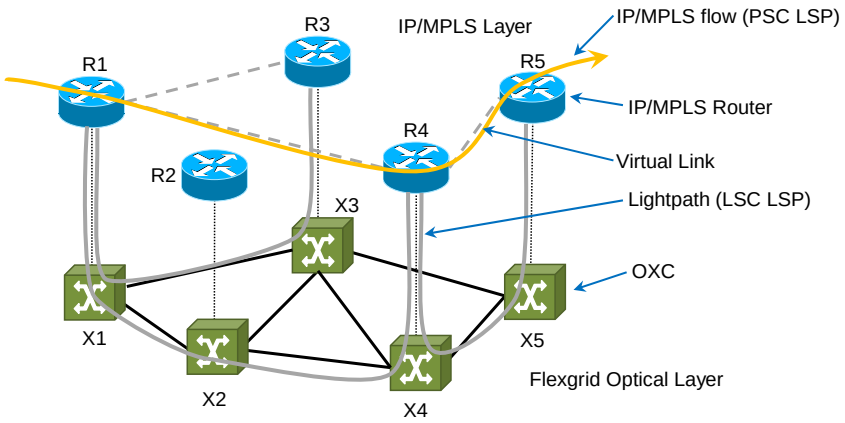


Fig. 2-6 Example of a multi-layer network.

2.3.3 Multi-domain Connectivity

Transport networks are usually organized in domains or Autonomous Systems (AS); i.e. a sub-network, or network segment, where the devices belonging to that domain share a specific characteristic, e.g. manufactured by the same vendor, using the same technology, managed by a specific operator, etc.

When the same operator manages different domains, the problem of provisioning a path that passes through different domains can be solved in the very same way as in single-domain networks, since full visibility of the network topology and resources is usually available. Fig. 2-7a represents a single-operator network with three domains and with full visibility. Path P1 is established crossing the three domains and the intra-domain path in each domain is known.

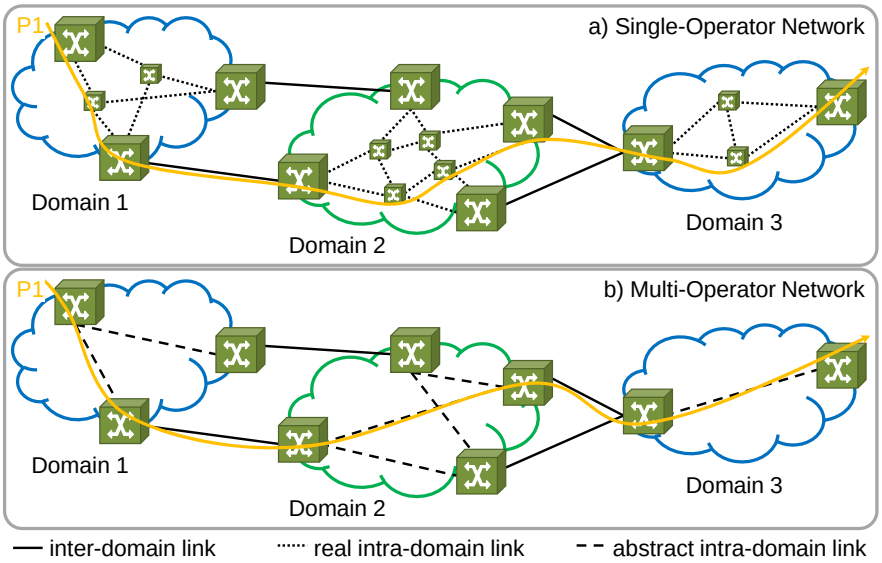


Fig. 2-7 Examples of multi-domain networks.

In contrast, when different operators manage each domain, the internal domain topologies are not visible to the rest of operators thus, *abstracted* topologies of the networks need to be used to compute end-to-end paths. Such abstracted topologies contain ingress and egress nodes in the domain, named as border nodes, as well as abstracted links representing the adjacencies with available resources between pairs of border nodes. Note that domains are connected by means of inter-domain links between border nodes on different domains. To provide end-to-end connectivity, each domain needs to provide intra-domain connections between their ingress and egress nodes according to the selected path. In Fig. 2-7b, different network operators manage each domain and the internal domain topology is unknown. Path P1 is established crossing all three domains but the intra-domain paths in each domain are computed by the operator of the domain.

2.3.4 Connection Recovery

During network operation, failures that affect physical resources supporting traffic can occur. Such failures could make unavailable a subset of active connections using those resources. To recover those failed connections, a recovery scheme can be applied. In that regard, protection and restoration schemes can be considered to recover the affected connections [Zha04-1]. In light of the significantly lower amount of resources used when restoration is implemented compared to protection, and the short recovery times that can be achieved (in the order of hundreds of milliseconds [Ve10-1]), restoration is commonly used to recover connections from failures.

2.4 Control and Management Planes

As introduced above, transport networks consist of different network segments (metro and core) and layers (IP/MPLS and optical), possibly managed by different vendors or using different technologies. Besides, operators need to provide different services including Internet access, voice communications, content distribution and Cloud computing infrastructures, among others.

To enable automatic provisioning and management of heterogeneous services over such multi- vendor/domain/layer networks, control and management planes need to be used. The control plane is in charge of automatically provisioning connections and reacting against failures in the network devices, while the management plane is responsible for providing fault, configuration, accounting/administration, performance and security (FCAPS) management services to the network. Moreover, standard interfaces need to be available to avoid vendor specific solutions.

Nonetheless, to enhance network operation, a robust infrastructure capable of performing complex operations, such as the Applications-based Network

Operations (ABNO) architecture [RFC7491], recently standardized by the Internet Engineering Task Force (IETF), or a Software Defined Network (SDN) controller, as defined by the Open Networking Foundation (ONF) [ONF], is required. In this thesis, we assume the ABNO architecture since many network operators, such as those in the Idealist project, are adopting this solution. In the next chapter, we elaborate on the reasons for this choice.

2.4.1 Applications-based Network Operations

The ABNO architecture consists of a number of standard elements with clearly defined functionality plus, a north-bound interface (NBI), a south-bound interface (SBI), and internal interfaces among modules (Fig. 2-8). This, enables distributing functional blocks across different processes for scalability purposes. The ABNO components need to work together to provide the network operations requested by the Network Management System (NMS) / Operations Support Systems (OSS) in charge of the network, or by an Applications Service Orchestrator (ASO).

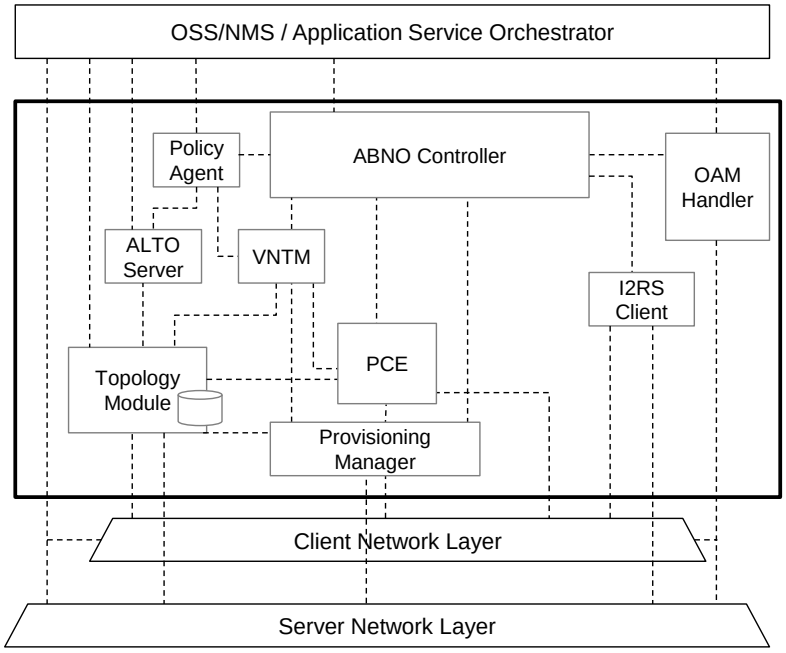


Fig. 2-8 ABNO architecture (reproduced from [RFC7491]).

A simplified ABNO architecture containing the elements required for the applications of in-operation planning developed along this thesis is shown in Fig. 2-9. Its elements are:

- The **controller** is the gateway that receives provisioning and service requests from the NMS / OSS / ASO through its NBI, translates them into simpler operations that the rest of functional components can deal with, and

invokes those components according to implemented workflows to serve the requests.

- The **policy agent** is the entity that regulates the use and access to network functionalities and resources and checks the rights of the different elements of the ABNO architecture before they commit any operation. It is responsible for propagating these policies to other components in the ABNO architecture.
- The **databases** module stores the information related to the network. The TE Database (TED) contains the state of network resources. Besides, when the Path Computation Element (PCE) is stateful (see PCE module below), the databases module contains an additional database, named as LSP Database (LSP-DB), maintaining information regarding current LSPs in the network; including the route, the bitrate, the spectrum allocation, the switching types, and the LSP constraints. The PCE updates the LSP-DB when connections are set-up, torn-down, or updated.

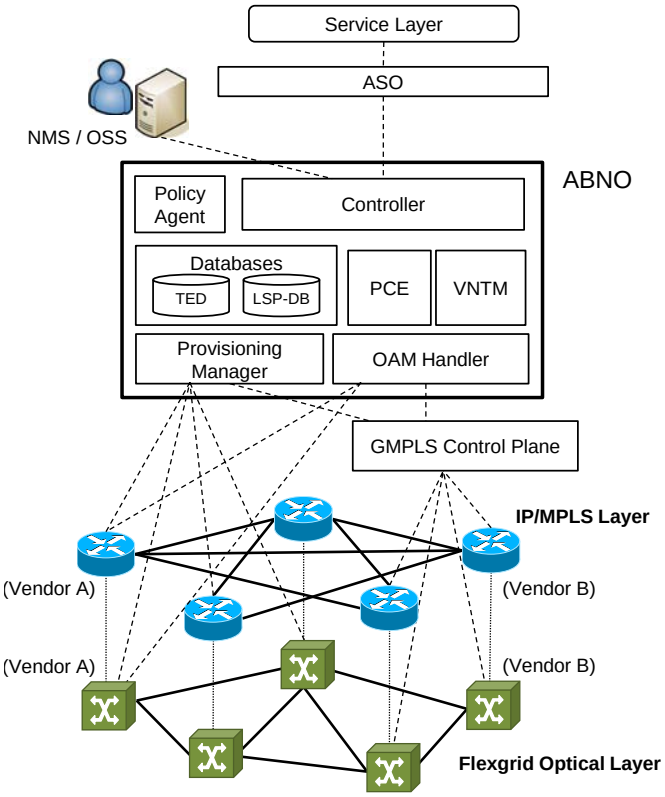


Fig. 2-9 ABNO-based Control and Management Plane Architecture

- The **PCE** [RFC4655] is the entity devoted to perform constrained path computation on a graph representing the network by solving the RSA problem in the case of flexgrid-based networks for incoming connection

requests as well as for solving optimization problems. It can be implemented with either stateless or stateful capabilities [draft-stateful]. In the stateless architecture, the PCE relies only on the TED to perform the path computation. The stateful PCE architecture extends the stateless one with the LSP-DB. Additionally, a stateful PCE may also include the active functionality that enables the PCE to modify the route and the frequency slot of already established LSPs.

- The **VNT Manager (VNTM)** [RFC5623] is in charge of creating and configuring VNTs in multi-layer networks by requesting to the PCE to set-up or tear down LSPs supporting the vlinks in such VNT [RFC5212].
- The **Provisioning Manager (PM)** is responsible for configuring the network devices to establish LSPs in the network**,** through the paths computed by the PCE. It uses ABNO's SBI to interact with a Generalized MPLS (GMPLS) [RFC3945] control plane or an SDN controller for directly programming the data path on each individual network node using protocols such as OpenFlow [OpenFlow].
- The **Operations, Administration and Maintenance (OAM) Handler** is responsible for receiving notifications from the network elements advertising for events such as failures, alarms or monitoring counters. When a message is received, the OAM Handler decides whether to invoke another component through the controller to deal with that event.

Some other elements are also included in the ABNO architecture. However, since they are not used in this thesis we refer the reader to [RFC7491] for further details. Additionally, a number of use cases are presented in [RFC7491] describing how ABNO can be applied to provide application-driven and NMS/OSS-driven network operations, such as multi-layer networking or global concurrent optimization.

2.4.2 Protocol Standards Review

In many of the studies in this thesis we assume a stateful PCE implementation because in-operation planning algorithms require to access, not only the network topology and the state of the resources in the TED, but also the information regarding the established LSPs in the LSP-DB.

As described in [Agu14-1], most of the internal interfaces between ABNO modules rely on the PCE Protocol (PCEP) [RFC5440]. For network topology synchronization between ABNO components, Border Gateway Protocol (BGP) with extensions to distribute link-state information (BGP-LS) [draft-bgpls] is proposed in [RFC7491], whereas the state of the LSPs can be synchronized by means of PCEP protocol with stateful extensions [draft-stateful]. In the latter, the Path Computation (PC) Report (PCRpt) message can be used to synchronize the LSP-DBs of two

components, e.g., the PCE and the PM. Since we assume an active implementation of the PCE, the LSP set-up/tear-down operations are requested to the PM by means of the Initiate message defined in [draft-initiate].

Regarding path computation requests, the PCE serves incoming PC Request (PCReq) messages containing the computation requests and replies with PC Reply (PCRep) messages.

In support to network re-optimization, PCEP extensions for Global Concurrent Optimization (GCO) were standardized in [RFC5557]. A GCO path computation request involves a set of connections to be jointly considered together with their respective constraints. In flexgrid networks, a GCO request can be applied to re-compute the route and spectrum allocation of a set of existing connections considering also routing of new connection demands.

When GCO is used to request an optimization procedure usually, at each request, the existing connections to be considered in that optimization need to be specified. For that end, it would be desirable to identify connections, e.g. by means of the SYMBOLIC_NAME Type Length Value (TLV) structure within the `LSP` object defined in [draft-stateful]. Unfortunately, current drafts do not support a path computation request referring to existing LSPs in the LSP-DB. A possible solution could be to explicitly identify them using its end points and their route and spectrum allocation. This is achieved by using the `END_POINTS` and record route (`RRO`) objects. However, this may entail scalability issues in cases where the number of connections included in a PCReq message is large.

Within the GCO PCReq message, a so called request list contains the parameters associated to each connection involved in the re-optimization specifying: the Request Parameters (`RP`) object identifying the request, its endpoints and relevant constraints such as `BANDWIDTH` or Include Route (`IRO`) objects. These requests can be grouped using the Synchronization VECtor (`SVEC`) [RFC6007] object to ensure that they are jointly considered in the re-optimization. Finally, Objective Function (`OF`) objects can be included to specify the desired GCO algorithm to be run.

Regarding PCRep messages, the PCE must reply to every request received in the incoming PCReq message. Each request can be replied including an Explicit Route Object (`ERO`) specifying the new spectrum allocation [RFC7698] or the `NO-PATH` object in case that no feasible solution was found. The spectrum allocation of an LSP can be encoded in the `Label` sub-object carried in the route objects, i.e. ERO, IRO and RRO, specifying the <*n*, *m*> tuple described in section 2.2 [draft-flexlabel].

The relative order for executing LSP operations is of paramount importance in re-optimization. For instance, a re-optimization algorithm should decide to reallocate lightpath A using resources of lightpath B after the latter is released. In such case, operations need to be ordered and performed sequentially, i.e. releasing the resources of B, before allocating them for A. In that regard, a TLV can be included

in the RP object of each response to specify the *delete* and *set-up* order of each existing connection. The standard defines that receiving the same set-up and delete values implies that the LSP stays unmodified. Nonetheless, that behavior has to be modified since some re-optimization applications, such as spectrum shifting, never turns down or sets up LSPs. Instead, they are updated, i.e. their central frequency is shifted.

2.5 In-operation Planning

A network optimization problem focuses, in general, on improving the performance of the network. Two main categories can be identified. *Traffic engineering* problems aim at routing traffic through currently available resources, by solving the RSA problem. In contrast, *network planning* problems consist in adding network resources where the current traffic is, e.g. optimizing resource utilization in places with high traffic demands. Specifically, network planning allows re-arranging active LSPs in a more efficient way in terms of network resources utilization, which is especially necessary under dynamic traffic. Two sub-categories of network planning can be distinguished: *off-line planning* and *in-operation planning*.

The off-line planning relates to network design problems aiming at reducing the total cost of ownership (TCO) of the network in terms of cost of deployment (CAPEX) and operational costs (OPEX). In network design problems, the optimal solution, or near-optimal solutions, have to be found since, the better the solution, the lower the cost of the network. In such cases, the algorithm solving the problem can take time to explore huge amount of different scenarios [Ru14-2].

Fig. 2-10 illustrates the classical network planning life-cycle, where a number of steps are performed sequentially. The life-cycle begins by planning and forecasting the network based on the service layer requirements, the state of the resources in the already deployed network, and inventory information. The output of this first step is a set of recommendations to satisfy the forecast traffic for a period of time. The duration of that time period depends on factors involving the operator and the traffic type.

This set of recommendations are used in the architecture and design step to define the changes to be applied to the already deployed network; these changes are later implemented in the network. During network operation, its capacity is continuously monitored and the collected data is used as input for the next planning cycle. The planning cycle may be restarted in case of network changes or unexpected increases in demands.

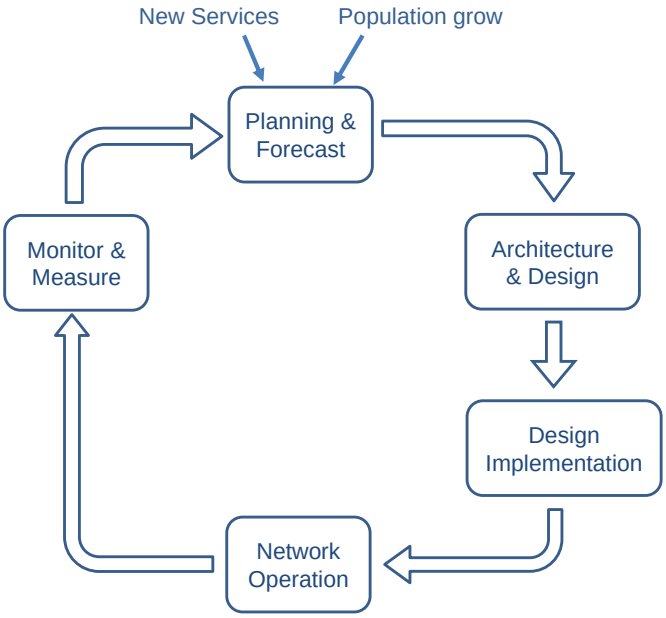


Fig. 2-10 Classical network planning life-cycle (reproduced from [Ve14-1])

In contrast to off-line planning, in-operation planning relates to problems to be solved over a network which is in-operation. Note that in-operation planning algorithms are a subset of on-line algorithms; while on-line algorithms can access to arbitrary data, such as geographic locations or service duration periods, in-operation planning algorithms are constrained to access the data available in the control plane. The main constraint of in-operation network planning algorithms is the stringent computation times in which solutions must be computed, usually in the range of hundreds of milliseconds to minimize traffic disruptions.

Traffic dynamicity entails setting up connections by allocating resources for them when they are requested and releasing the resources when the connections are torn down. As a result of this repeated operations, the available optical resources in the network could be used in a non-optimal way thus, increasing the amount of connections blocked when resources fulfilling the contiguity and continuity constraints cannot be found. This entails reducing the Grade of Service (GoS) of the network.

Assuming the benefits of dynamically operating the network, the classical network life-cycle can be extended with a new step, as shown in Fig. 2-11. This new step takes care of network re-configuration and re-optimization in real time during network operation, implementing the changes immediately in the network.

The main purpose of network re-optimization techniques is to improve network resource utilization and thus, its GoS. Solving network re-optimization problems usually entails solving combinatorial problems that can be huge. To solve them, heuristic algorithms can be used to find good enough solutions in reasonable

computation times. Note that if the solution takes too much time to be computed, the network state may be changed making the solution outdated.

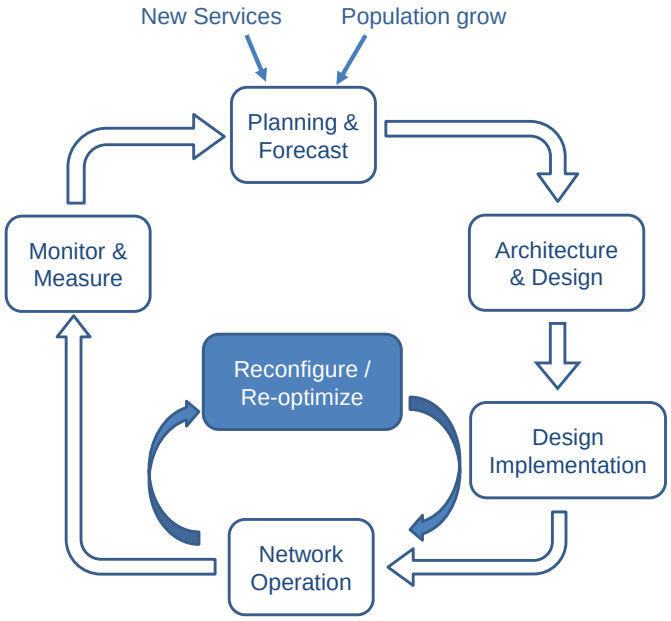


Fig. 2-11 Augmented network life-cycle (reproduced from [Ve14-1])

2.6 Conclusions

Significant background for this thesis has been presented. The telecom cloud and its relation with optical transport networks have been introduced. Next, the models used to represent optical networks along the thesis have been described, as well as the network devices that compose an optical network. Then, the different connectivity types and the main concepts on multi-layer and multi-domain networks have been introduced.

The ABNO, an architecture that can support the implementation of in-operation planning algorithms, as well as the protocols involved in ABNO operation have been reviewed. Finally, the main concepts of network planning and in-operation planning have been introduced.

Next chapter reviews the state-of-the-art regarding the objectives of this thesis.

Chapter 3

Review of the State-of-the-Art

This chapter reviews the state-of-the-art related to in-operation planning. It is organized as follows: first, previous works related to in-operation planning and multi-domain architectures are reviewed and the recent works related to telecom cloud infrastructures are explored. Next, re-optimization algorithms are reviewed including: *i*) triggering events that can be used to start a re-optimization process; *ii*) works related to connection provisioning, including p2p, p2mp, and anycast; *iii*) applications of network re-optimization; and *iv*) inventory-based in-operation planning applications. Finally, works tackling monitoring-based data analytics are reviewed.

3.1 Architectures for In-Operation Planning

Several architectures that can be used for complex computations like those needed by on-line re-optimization algorithms can be found in the literature.

Specifically, in-operation planning algorithms can be implemented inside SDN controllers. In fact, the ONF studies different use cases related to in-operation planning in [ONF-TR509]. For instance, they propose the joint optimization of packet and optical layers of a network after a failure or the addition of new elements resulted in a non-optimized use of the network resources.

Some works have demonstrated the use of SDN controllers for re-optimization. For instance, in [Che14-1], authors extended a POX-based OpenFlow controller [POX] to support bundle re-configuration; candidate lightpaths are packed in bundles, where lightpaths in a bundle are jointly re-optimized. A make-before-break scheme was used to minimize traffic disruptions. Another example of spectrum defragmentation was proposed in [Zhu15-1], where authors extended POX with a

multi-domain spectrum defragmentation algorithm; they implemented an inter-domain protocol to coordinate controllers in different domains. Besides, authors in [Par14-1] extended a OpenIRIS [OpenIRIS], a Floodlight-based SDN controller [Floodlight], with a mechanism to re-route flows traversing congested links. When packet loss was detected, new routes were computed avoiding such congested links.

Usually, in-operation planning algorithms require high computation capacities to solve optimization problems in stringent times. Centralized elements, such as an SDN controller or a PCE, can provide limited computation capabilities, so it is desirable to move highly intensive computations from those elements to specialized modules with advanced computation capabilities. However, SDN-based architectures are monolithic (i.e. consists of a single module) thus, difficulting both their extension with in-operation planning algorithms and using specialized/advanced computation capabilities. Authors in [Pao15-1] configured the PCE to be in charge of running re-optimization algorithms. However, this architecture suffers from the same problems identified for SDN, i.e. it is monolithic and hard to extend with algorithms.

For this very reason, network operators are, in general more interested in ABNO-based architectures for in-operation planning since ABNO is composed of a set of standardized modules with well-defined functionalities and interfaces. The main component of ABNO is the PCE.

Several PCE architectures, including single PCE and multiple PCE architectures are surveyed in [Pao13-1]. One single PCE can be in charge of computing the paths independently of the source node for the path or, conversely, the source nodes can be partitioned in groups and each group can be served by a different PCE.

Similarly, in multi-layer networks, the PCE can be split into two separated PCEs: the Layer 2 PCE (L2 PCE), responsible for computing PSC paths, and the Layer 0 PCE (L0 PCE) responsible for the LSC ones.

Another architecture studied in [Pao13-1] consists in a single primary PCE (also named as front-end PCE, fPCE) in charge of computing paths for all the network nodes, which is assisted by a secondary (back-end) PCE (bPCE). Authors in [StrongestD34] adapted the fPCE/bPCE architecture aiming at introducing more flexibility in terms of load balancing and resiliency and considered a number of bPCEs, where each bPCE could be specialized to compute a subset of algorithms; the fPCE selects the bPCE as a function of the path computation algorithms. Aiming at adding flexibility, authors in [Case13-1] extended the PCE architecture to enable plugging-in third-party algorithms in the form of shared libraries.

Finally, authors in [Ve14-1] proposed to use the fPCE/bPCE architecture to enable in-operation planning, where the bPCE is in charge of running re-optimization algorithms entailing intensive computations. In this regard, the ABNO architecture depicted in Fig. 2-8 needs to be upgraded as depicted in Fig. 3-1.

We rely on the ABNO architecture for developing the modules to experimentally assess the proposed in-operation planning applications and the fPCE/bPCE architecture is used as a starting point for this PhD thesis. We define workflows as the repeatable interactions between the ABNO modules. In addition, previous works inspired us to design an architecture based on highly flexible and configurable modules.

An additional advantage of the fPCE/bPCE architecture is that the bPCEs can be based on specific hardware devices, such as Field Programmable Gate Arrays (FPGAs) (see e.g. [Xilinx]) or Graphic Processing Units (GPUs) [Nvidia]. For instance, quality of transmission computation, requiring extensive computation can be delegated to be computed in a FPGA as the authors in [Az11-1], [Per11-1] demonstrated.

In this thesis, we take advantage of GPU devices to execute many threads in parallel aiming at accelerating path computations.

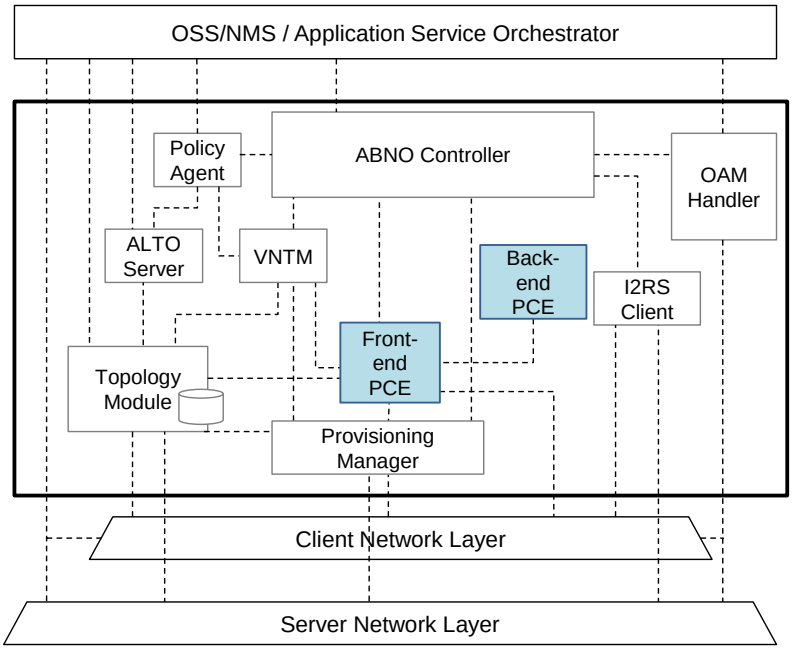


Fig. 3-1 ABNO Architecture extended with the fPCE/bPCE scheme.

3.2 Architectures for Multi-Domain Networks

Regarding multi-domain path computation, two different scenarios can be considered: *i*) single-operator, in which the topology of the different domains is fully visible; and *ii*) multiple operators, where the complete topology of each of the domains is only fully visible inside that domain.

In single-operator, multi-domain networks are usually created as a result of deploying nodes from different vendors and/or different technologies. In such case, a PCE (or SDN controller) is in charge of computing paths in each of the domains, where each PCE (named as child PCE) has full visibility of the topology and resources of the underlying domain. A parent PCE is connected to every child PCE and it is able to compute the sequence of domains [RFC6805]. For the intra-domain paths two options can be considered depending on whether the parent PCE has full visibility of whole network topology or not. In the former, the parent PCE can compute itself the end-to-end path, while in the latter, intra-domain paths computation is delegated to child PCEs.

For the second case, i.e. multi-operator multi-domain networks, several approaches have been considered in the literature. Authors in [Pao13-1] surveyed two different techniques in the case where a PCE is in charge of computing paths for each of the domains. The standardized backward path computation procedure [RFC5376] allows computing an end-to-end path through a chain of PCEs, where each PCE computes a sub-path for the underlying domain. Starting from the destination PCE, the end-to-end path computation is performed towards the source PCE. Since this procedure results in sub-optimal path computation, the backward recursive PCE-based computation procedure [RFC5441] was standardized to compute optimal constrained paths. Nonetheless, both procedures assume that the sequence of the domains has been previously determined. To decide the sequence of domains, a parent PCE can be deployed. In this case, the parent PCE gets an abstracted topology of each domain.

Similar architectures than those proposed for PCE are being considered when an SDN controller is in charge of each of the domains [OIF15-1], [ONF-TR509].

In this PhD thesis, we focus on the case of multi-operator multi-domain networks. In such scenario, the author in [Yo14-1] proposed to use a number of market-driven brokers on top of the child PCE / SDN controller in charge of each domain. Since the proposed scheme provides autonomy to the network domains whilst improving scalability, in this thesis we adopt that architecture.

3.3 Telecom Cloud

Authors in [De12-1] identified the IT and network requirements for new generation applications. In particular, they concluded that flexible optical networking technologies will play an important role in future cloud applications requiring flexible high-bandwidth connectivity services. To fulfill these requirements, telecom operators are integrating DCs and preparing their transport networks to be ready for cloud applications [Con12-1] to create a telecom cloud infrastructure [Ve15-2].

Authors in [Ericsson12-1] and [Ka12-1] identified the opportunities of telecom operators to lead the market of cloud computing services. They concluded that telecom operators could benefit of telecom cloud from two points of view. As providers, they could deliver on-demand applications and computing capacity for cloud services with the added value of managing the connectivity. As users, they can transfer their business functions to the cloud as other enterprises does thus, resulting in cost efficiency and time to market.

Several studies related to the cloud and network cross-stratum orchestration have been lately carried out. The authors in [Ve14-4] studied the cost savings obtained by inter-connecting DCs and in [Ve15-2] a hybrid architecture composed of large and micro DCs requiring high connection dynamicity was proposed. In [Mah11-1], the authors devised an architecture to provide bandwidth on demand over WDM. In [Bu14-1] an enhanced control plane following a hierarchical PCE architecture to orchestrate the IT and network components in a telecom cloud infrastructure was proposed for jointly performing network routing and IT server provisioning decisions applicable to anycast scenarios for IT resource selection.

High-level services offered by the network operators, usually involves orchestrating several components in the Telecom Cloud. The ASO was proposed in [Ve13-1] as the service layer element responsible for the coordination between applications and the ABNO controller in the transport network. It is responsible of translating service connectivity requests from the high-level services to network-level requests for the ABNO. Cloud applications can use some specific semantic that ASO adapt to the network-specific semantic, which it is based on connections [Ve14-3], [As14-1].

In this thesis, we use the above architecture for telecom cloud, where large DCs are placed in the core transport network, while micro DCs are placed in the metro networks, i.e. close to the users.

3.4 Re-optimization

3.4.1 Triggering Events

For in-operation planning, it is of paramount importance the events that trigger re-optimization workflows. Authors in [Wu11-1] surveyed the available literature regarding network re-optimization algorithms in WDM and concluded that those algorithms could be triggered by the following events: *i*) traffic pattern changes; *ii*) addition of network resources; and *iii*) removal of network resources due to a network failure or a planned network maintenance. Besides, algorithms can be triggered by periodic events. In addition, re-optimization can be done *reactively*, when it is intended to mitigate some performance degradation (e.g. a blocked connection request) or *proactively*, when it is intended to prevent future

performance degradation (e.g. when the amount of traffic conveyed reaches some threshold, so as to prevent bitrate blocking).

Many examples of on-line re-optimization algorithms triggered by changes in the traffic pattern can be found in the literature. For instance, authors in [Bou05-1] proposed rerouting existing lightpaths with the aim of increasing the network GoS. Authors in [Chu07-1] compared the performance of reactive and proactive rerouting schemes in the presence of wavelength converters. Different strategies to identify when a re-optimization needs to be performed and which demands have to be selected were proposed and compared in [Ah11-1].

In this thesis, we rely on the triggering events described in this section to decide whether the proposed in-operation planning mechanisms need to be run.

3.4.2 Provisioning-triggered Operations

Path Provisioning

The RSA problem needs to be solved to set up an optical connection on a flexgrid optical network. In dynamic scenarios, fragmentation of the spectrum appears in flexgrid networks as a result of connection's dynamicity and spectral bandwidth diversity thus, increasing the blocking probability of connection requests and degrading the network GoS. A possible solution is to install spectrum converters in the OXCs. However, as a consequence of its high cost they are rarely used in optical networks.

To improve network efficiency, defragmentation can be applied by re-configuring selected connections, thus compacting the utilized resources and facilitating that incoming connection requests can be served. Several re-optimization strategies (including rerouting and spectrum reallocation) have been proposed so far for optical networks and can be applied to the specific case of flexgrid networks [Wan13-1], [Agr09-1], [Cast12-1], [Cast13-1]; those strategies can be divided into proactive and reactive strategies considering the way they are triggered [Wan13-1]. The proactive strategy focuses on minimizing spectrum fragmentation itself at a given period of time, whereas the reactive strategy focuses on making enough room for a given connection request if it cannot be established with current resources allocation. When a re-optimization algorithm is reactively triggered after a connection request cannot be served it is also known as *provisioning-triggered* strategy.

Periodic defragmentation might require long computation times as a result of the amount of data to be processed and therefore, it is essentially performed during low activity periods, e.g. during nights [Agr09-1]. Conversely, provisioning-triggered defragmentation might provide solutions in shorter times and can be run in real time since it involves only a limited set of already established connections [Cast12-1], [Cast13-1]. To mitigate the fragmentation of the spectrum, the authors in

[Cast12-1] devised an algorithm based on re-allocating connections and solved it in reasonable computation times, e.g. < 1 s. A similar defragmentation algorithm based on lightpath re-allocation was proposed in [Zhu15-1].

Note that re-allocating a connection might entail traffic disruption. To avoid disruption, the standardized *make-before-break* technique can be applied; it allows establishing the new path for rerouting and transferring traffic from the current path to the new one before the old path is finally torn down. In this way, reallocation procedures do not cause any working traffic disruption, but they require using additional network resources.

Authors in [Qin13-1] proposed and experimentally evaluated a novel technique to perform hitless spectrum defragmentation of multiple channels without causing errors on connections lying between the initial and final spectrum locations by using a fast wavelength auto-tracking scheme. Defragmentation is performed in the order of tenths of microsecond. To avoid traffic disruption during the defragmentation operation, they used a small link-layer buffer to store the data transferred.

In parallel, the authors in [Cu13-1] proposed the so named *push-pull* technique for hitless spectrum defragmentation; the technique performs defragmentation by moving lightpaths only to contiguous and free spectrum frequencies along the same route of the original path. Although such constraint might limit the efficiency of the overall re-optimization, particularly when the optical network is heavily loaded, it showed good performance when applied to capacity elasticity [Pao15-1].

Defragmentation can be also applied in the contexts of multi-domain single operator scenarios, where the visibility of the resources within the domains is available in a centralized element [Zhu15-1].

To the best of our knowledge, no previous works proposed integrating the control algorithms for the hitless push-pull technique in an in-operation planning tool. Moreover, defragmentation in multi-operator scenarios has not yet covered in the literature. In consequence, in this thesis, we study hitless defragmentation using the push-pull technique and experimentally validate its feasibility. In addition, per-domain spectrum defragmentation is carried out in the context of multi-operator multi-AS networks.

Point-To-MultiPoint Provisioning

Although p2mp content requests are usually served in packet switching layers, it makes sense to serve high-capacity connections directly in the optical layer. In that regard, two approaches can be considered to interconnect a set of nodes through the optical layer: using a set of lightpaths or using one single light-tree. Several works have compared the performance of establishing light-trees against lightpaths. Authors in [Rah09-1], [An10-1], [Lin13-1], and [Yu13-1] proposed algorithms for computing the light-trees for multicast provisioning over WDM networks; they concluded that using light-trees improves optical resources

utilization in about one order of magnitude with respect to using a set of lightpaths.

Notwithstanding the improvement from serving large bitrate multicast requests on the optical layer, implementing light-trees depends on the technology of the optical nodes. Authors in [Sam13-1] experimentally demonstrated the feasibility of multicast over flexgrid-based networks and extended their PCE to support p2mp provisioning and to control the data-plane.

Besides, authors in [Ru14-1] compared the performance of creating a single transparent light-tree against creating several transparent sub-trees and showed that, although resource utilization is improved using a single light-tree, its limitation lies in the spectrum continuity constraint and the connections' length, thus producing poor performance in large networks [Ve14-2].

Some works have explored the creation of virtual topologies to groom multicast traffic. Authors in [Ha04-1] and [Mel01-1] created VNTs, where vlinks are supported by lightpaths on an underlying WDM network. Multicast connections are set-up by creating trees on such VNTs. Serving multicast connections on a VNT originates an asymmetric use of resources when bidirectional vlinks are considered. To deal with such asymmetry, authors in [Wo13-1] studied the benefits of creating unidirectional lightpaths to support the virtual topology when traffic is significantly asymmetric. In contrast, light-tree-based VNT on WDM was first introduced in [Sah99-1] and updated in [Ya03-1]; the resulting virtual topology connects the source node to the set of destination nodes by means of direct p2p virtual links.

In this thesis, we study the single-layer and multi-layer p2mp approaches over a flexgrid optical network. After its evaluation, both p2mp approaches are experimentally validated.

Anycast provisioning

The authors in [Gh12-1] consider both, available network resources and IT resource requirements to compute an optimal anycast route for dynamic connections. Routing and spectrum allocation algorithms were proposed in [Wal14-1] to serve anycast connections on elastic optical networks; to select the destination of the anycast connection, a set of candidate paths is computed.

Since the approaches available might entail long computation times, in this thesis, we propose a novel mechanism for provisioning anycast connections in a multi-layer scenario; the mechanism avoids the computation of candidate paths by using an auxiliary graph.

3.4.3 Other Triggering Events

As stated in section 3.4.1, re-optimization can be triggered after new resources are added to the network. One specific case of adding network resources is when a link becomes active after it has been repaired. Performing a network re-optimization in this scenario might improve blocking probability by rerouting part of the traffic through the new link. Authors in [Zo15-1] named that problem as After Failure Repair Optimization (AFRO), proposed a mathematical formulation and carried out extensive numerical simulations for its performance evaluation.

AFRO is thus, mostly associated to the case of link failures. In the event of a link failure, restoration schemes can be applied to recover affected connections. Since connection restorability is related to the availability of continuous and contiguous spectrum resources, the multipath (MP) restoration scheme was proposed in [Cast14-2] to increase restorability. MP-restoration divides original connections into several parallel disaggregated connections, which are routed independently among them. In addition, authors in [Son11-1] proposed to recover only part of connections' original bitrate in case that the lack of resources prevented recovering all the original bitrate (this is named as *bitrate squeezing*).

Although MP-restoration brings benefits, it provides poor resource utilization and it is spectrally inefficient. Hence, re-optimizing the network after a link is repaired becomes beneficial. For this very reason, the AFRO problem needs to be enhanced with the capability of merging parallel connections and increasing the served bitrate to its original value. As far as we know, no available works can be found in the literature exploring the AFRO problem using multiple paths.

Another scenario for re-optimization is the case where a failure affects a VNT; such VNT might become disconnected thus, creating several connected components. The VNT can be designed to cope with network failures, e.g. creating 2-connected topologies where the failure of one link does not disconnected the topology. Nonetheless, simpler network topologies, such as trees, can be considered and *pre-planned reconnection* (similar to protection) and *dynamic reconnection* (similar to restoration) schemes can be considered to *reconnect* the VNT in the event of a failure. In this regard, authors in [Qiu11-1] proposed a fast spanning tree reconnection mechanism to handle single link failures in Ethernet-based networks.

However, to the best of our knowledge, no works are available in the literature proposing similar schemes for optical networks.

3.4.4 Inventory-based In-Operation Planning

When the capacity of the optical backbone network becomes exhausted in some parts, new capacity needs to be installed in the network. To that end, network operators periodically, e.g. yearly, can use a planning tool to decide the equipment

and connectivity to be installed at the minimum cost to cope with the current and forecast traffic.

In consequence, planning a network requires reliable traffic forecasting so the resulting network design can cope with future traffic. Nonetheless, making prediction for such traffic increments is not an easy task especially if, in addition, it comes with spatial changes in the demand (e.g. during business and night hours). In fact, in most cases there are just forecasts for traffic volume growth over time, which results in installing spare capacity that is not even used during the planning cycle.

In this context, some authors have proposed interesting works to plan the backbone network for multiple periods. In [St06-1], the authors assume knowledge of the traffic for every period to provide a globally optimal solution to the planning problem. That knowledge is, however, not easy to be gained and it can lead to large inefficiencies when real and forecast traffic diverge. In view of that, incremental planning was proposed. Authors in [Bon13-1] tackled the problem of reducing power consumption targeting the power-aware logical topology design. The problem of designing translucent optical networks was faced in [Sh07-1] and [Zhu12-1]. Authors in [Gea01-1] proposed techniques for multi-period planning based on routing optimization. Besides, a migration scenario aiming at increasing the capacity of optical networks was studied in [Meu10-1].

In a recent study, the authors in [Ei15-1] proposed using flex-rate transmission taking advantage of its adaptability to changing and unknown conditions. In addition, they propose taking advantage of modular line-card architectures to facilitate implementing *pay-as-you-grow* approaches.

The steps for implementing gradual migration from currently deployed WDM optical networks to flexgrid ones were presented in [Ru14-2]. In their discussion, the problem of reconfiguring the optical network came to light. Such problem has to be solved using an advanced planning tool able to propose solutions not only for reconfiguring the already deployed network resources, but also to propose new resources to be purchased and installed taking into account new services, population, and inventory. In that regard, the network planning problem needs data from operational, as well as from inventory databases.

The problem to upgrade an operator's network when it cannot convey the required amount of traffic was introduced in [Ve14-2]. This problem takes a network topology, a set of resources available for growing up this network, and a traffic matrix representing the traffic patterns to be conveyed in the network, and computes the set of resources that have to be deployed in the original network to fulfill the traffic matrix requirements.

The above inspired us to address the problem of devising network reconfiguration algorithms to perform incremental network capacity planning, so in this thesis we

consider an inventory database in addition to the operational ones already in the in-operation planning tool.

3.5 Monitoring-based Data Analytics

Network operators are looking for efficient architectures able to reduce costs, while providing the required GoS. To that end, VNTs need to be dynamically adapted not only to variations in traffic volume, but also to changes in the direction of the traffic. In addition, traffic monitoring is an essential task for network operators since it allows to evaluate network performance. Specifically for packet networks, the traffic monitoring function allows identifying (classifying) the traffic belonging to a specific service or destination so as to apply specific policies. Monitoring traffic samples can be collected in a repository for further analysis as proposed in [Mas14-1].

Authors in [Mat12-1] suggested a model based on splitting a 24 hours day period into 16 non-overlapping intervals of 90 min. Their on-line change detection algorithm identified relevant changes in link utilization and reported those links to the NMS for further analysis. Finally, authors in [Sou07-1] compared three traffic aggregations formalisms such as ingress routers, input links, and Origin-Destination (OD) pairs and concluded that traffic aggregation level has a significant impact on the number of relevant traffic changes detected; they showed that aggregating traffic by OD pairs is the most appropriate choice.

[RFC7536] presents a list of traffic monitoring use cases. Specifically, one of the use cases is of particular interest for this PhD thesis. It consists in applying machine-learning techniques to create traffic models [Mo16-1]. By using these traffic models, traffic matrices for the near future can be constructed and used as input for VNT reconfiguration. To automate VNT adaptability, traffic needs to be monitored in the packet nodes and counters are accessible by the control plane.

In this PhD thesis, we assume that the disaggregated traffic volume forwarded by each packet node to every other destination node is available. In addition, a notification message can be also triggered when the used vlink capacity reach some configured threshold (e.g., 90%). In fact, threshold triggered VNT reconfiguration was proposed in [Agu15-1]. Nonetheless, VNT adaptation requires from powerful architectures and algorithms to analyze large amounts of monitored traffic data, so as to anticipate, when possible, to traffic changes targeting at optimizing resource utilization. To that end, in this thesis we propose a big data network manager architecture to support VNT adaptability based on traffic prediction from applying data analytics on the monitored traffic data.

3.6 Conclusions

In this chapter, we reviewed the state-of-the-art of some relevant works related to the goals of this thesis. Table 3-1 summarizes the study.

Table 3-1 State-of-the-art summary

Goals		References
G1 – Architectures for in-operation planning		[Az11-1], [Case13-1], [Che14-1], [Floodlight], [Nvidia], [OIF15-1], [ONF-TR509], [OpenIRIS], [Pao13-1], [Pao15-1], [Par14-1], [Per11-1], [POX], [RFC5376], [RFC5441], [RFC6805], [StrongestD34], [Ve14-1], and [Xilinx].
G2 - Applications of in-operation network planning	G2.1 - Spectrum shifting.	[Agr09-1], [Ah11-1], [Bou05-1], [Cast12-1], [Cast13-1], [Che14-1], [Chu07-1], [Cu13-1], [Pao15-1], [Qin13-1], [Wan13-1], [Wu11-1], and [Zhu15-1].
	G2.2 - After failure repair optimization using multiple paths.	[Cast14-2], [Chu07-1], [Son11-1], [Wu11-1], and [Zo15-1].
	G2.3 - Single- and multi-layer point-to-multipoint connectivity.	[An10-1], [Ha04-1], [Mel01-1], [Lin13-1], [Rah09-1], [Ru14-1], [Sah99-1], [Sam13-1], [Ve14-2], [Wo13-1], [Ya03-1], and [Yu13-1].
	G2.4 - Content distribution in the telecom cloud.	[As14-1], [Bu14-1], [Con12-1], [De12-1], [Ericsson12-1], [Gh12-1], [Ka12-1], [Mah11-1], [Qiu11-1], [Ve13-1], [Ve14-3], [Ve14-4], [Ve15-2], and [Wal14-1].
	G2.5 - Inventory-based planning.	[Bon13-1], [Ei15-1], [Gea01-1], [Meu10-1], [Ru14-2], [Sh07-1], [St06-1], [Ve14-2], and [Zhu12-1].
	G2.6 - Multi-domain connectivity.	[Case11-1], [OIF15-1], [ONF-TR509], [Pa11-1], [Pao13-1], [RFC5376], [RFC5441], [RFC6805], [Yo14-1] and [Zhu15-1].
	G2.7 - Monitoring-based data analytics.	[Agu15-1], [Mas14-1], [Mat12-1], [Mo16-1], [RFC7536], and [Sou07-1].

We can conclude that, although some previous works have proposed on-line algorithms and some others have performed simulation and experimental assessments, no previous works have covered in-operation planning in flexgrid optical networks.

Chapter 4

Architectures for In-Operation Planning

Computations performed in the PCE may be intensive when solving the RSA problem for LSP re-optimization. To reduce PCE overload, thus eliminating bottlenecks, the PCE architecture supports load-sharing between two (or more) coordinated PCEs: fPCE / bPCE.

As previously introduced, the PCE overload can be shared between the fPCE and the bPCE. For in-operation planning, we take advantage of that architecture and consider that fPCE steers the RSA computations for single LSP computation, whilst bPCE is dedicated to more complex computation such as restoration/re-optimization, where a number of LSP need to be jointly optimized. This requires synchronizing the local TED and LSP-DB databases of both fPCE and bPCE; in particular, BGP-LS is used for the TED and PCEP for the LSP-DB.

Looking at the technology used in computers architecture for hardware acceleration, we found that GPUs are lately used on intensive computational tasks. State-of-the-art GPUs can perform computations in the range of Tera-FLOPS (FLoating-point Operations Per Second) consuming a fraction of energy and physical space compared to CPU-based computers.

In this chapter, we propose the use of GPUs to reduce computation times in the bPCE. We experimentally compare the performance of the proposed high performance (HP) bPCE on two real backbone networks and we study PCEP and BGP-LS protocols to coordinate fPCE and bPCE.

After the fPCE-bPCE architecture has been validated, we devise a software architecture that provides a set of interfaces, including PCEP and BGP-LS, and that is able to run optimization algorithms.

4.1 Front-end / Back-end PCE Architecture

In the classical centralized PCE architecture (Fig. 4-1a), the PCE executes all tasks to serve connection requests using its internal CPU. Although parallelism is available in current CPU architectures (e.g. multi-core or even multi-processor), the amount of threads that can be executed in parallel is relatively low, e.g. 4 cores per processor. Therefore, in case of an intensive computation needs to be performed the PCE can either dedicate all the available resources to that computation thus, introducing additional delay to any other request arriving during such computation, or dedicate only part of the resources thus, enlarging the computation time of that intensive computation.

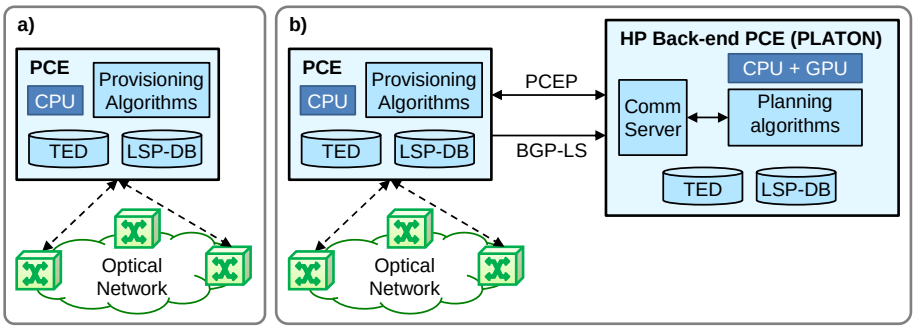


Fig. 4-1 Centralized (a) vs. fPCE / bPCE Architecture (b).

A solution to the above problem is to split the centralized PCE into two parts (Fig. 4-1b). Note the replacement of the centralized PCE by the fPCE and bPCE modules. While the fPCE is responsible for simple computations (e.g. single path computation), intensive computations are delegated to the bPCE.

When an intensive task need be computed, the fPCE forwards the set of path computation requests to the bPCE to solve the task. To this end, PCEP can be used; a single PCReq message can contain the whole set of path computations and a SVEC object is included to group together several path requests so as the computation is performed for all those requests (bulk), attaining thus a global optimum.

It is clear that separating the PCE into two parts introduces some additional delay to intensive path computation. Therefore, an estimation of the minimum size to decide whether a computation is better performed locally at the fPCE or is delegated to the bPCE is needed to minimize bulk path computation running time.

To compare the above PCE architectures, let us consider a simple single-layer optical network bulk path computation problem. Note that this problem can be easily applied to many scenarios, including failure recovery and re-optimization. The bulk path computation problem can be stated as:

Given:

- a network topology represented by a graph G(N, E), where N is a set of nodes and E is a set of fiber links each one connecting two nodes;
- the spectrum characteristics of each link, including the set S of slices; and
- a set (bulk) D of demands to be routed and allocated on the network; each demand d is identified by the tuple <s_d, t_d, n_d>, where s_d and t_d are the source and destination node, respectively, and n_d is the amount of slices required for the bitrate requested.

Output: the route and spectrum allocation for each demand d in the bulk.

Objective: Minimize the amount of spectrum resources (i.e. each slice in each link) used.

The problem need to be solved in stringent times (e.g. ms.); as a consequence, some heuristic algorithm need to be devised. Although several meta-heuristics can be used (see e.g. the algorithms in [Cast12-1], [Cast13-2]), for the sake of clarity, we have developed a simple algorithm that performs a number of iterations, randomly sorting the demands in the bulk at each iteration (Table 4-1). Following the resulting ordering, the RSA problem is solved for each demand and the resources are allocated in the local copy of the TED representing the graph $G(N, E)$ (lines 5-9 in Table 4-1). At the end of each iteration the objective function is computed (line 10), the best solution is updated (lines 11-12), and the solution with the minimum objective value is eventually returned (line 13).

Table 4-1 Bulk Path RSA Heuristic Algorithm

```
INPUT G(N, E), D, maxIter
OUTPUT bestS
 1:  bestS ← Ø
 2:  for i = 1..maxIter do
 3:    S ← Ø; G' ← G
 4:    sort(D, random)
 5:    for each d ∈ D do
 6:      {<k, c_k>} ← getRSA(d.src, d.tgt, d.bw)
 7:      if k = Ø then continue
 8:      allocate(G', d)
 9:      S ← S ∪ {d}
10:    Compute Φ(S)
11:    if Φ(S) > Φ(bestS) then
12:      bestS ← S
13:  return bestS
```

For illustrative purposes, Fig. 4-2 presents an example of a bulk consisting of 3 demands to be computed on the 4-node and 5-link network topology. Two iterations are shown: for the first (second) iteration, Fig. 4-2a (Fig. 4-2d) shows the network topology and the routed demands, Fig. 4-2b (Fig. 4-2d) gives insight of the order and the route of each demand, whereas Fig. 4-2c (Fig. 4-2e) depicts the spectrum

allocation in each link (each slice identifies the demand using it). Note that different bulk orderings result in different solutions and thus, different resource usage. For instance, demand with Id. 3 (B-D) is routed through path B-C-D in the first iteration as a consequence of the lack of free contiguous spectrum slices in the direct link B-D, whilst a direct path through link B-D is used in the second iteration.

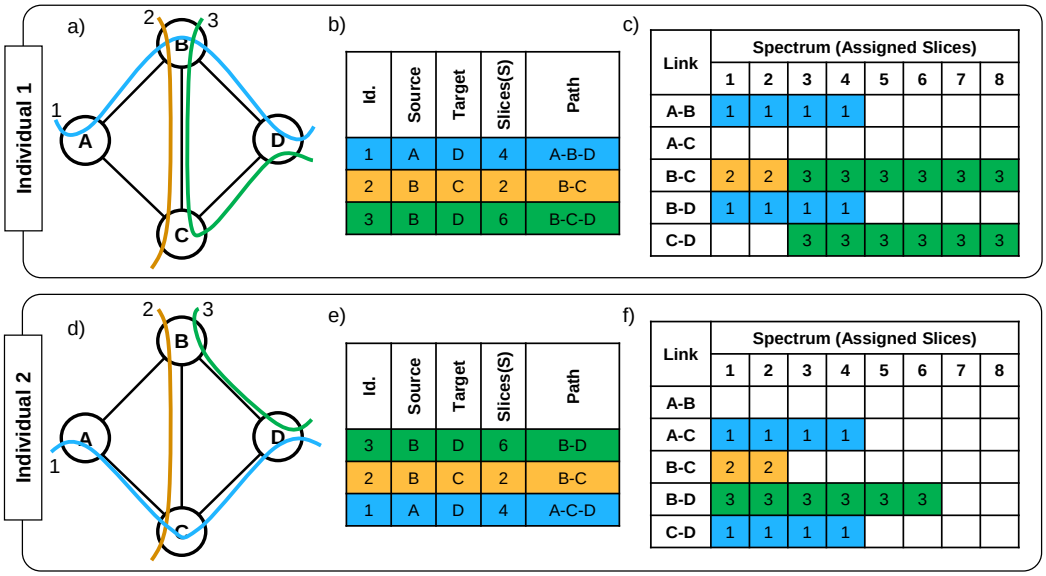


Fig. 4-2 Example of RSA for two heuristic iterations.

Let us now to validate the fPCE/bPCE architecture. In our experiments, we consider two different real backbone network topologies. The 22-node and 35-link British Telecom (BT) network topology (Fig. 4-3a) and the 30-node and 56-link Telefonica (TEL) network topology (Fig. 4-3b). These networks are preloaded using a simulator with a number of optical connections to ensure that, at least, one demand in the bulk is routed through a path different from the shortest one. Therefore, different iterations will produce different solutions depending on the order of the bulk.

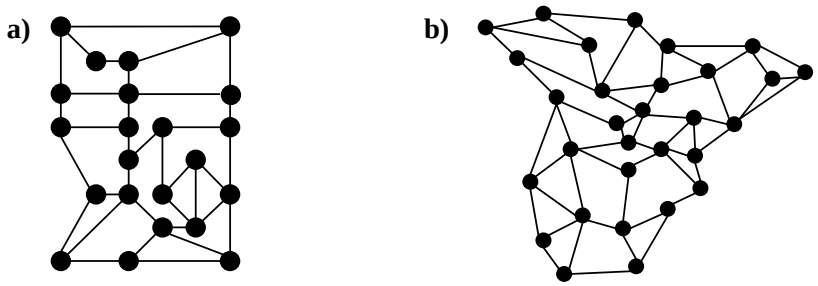


Fig. 4-3 British Telecom (a) and Telefonica (b) network topologies.

The heuristic algorithm has been implemented in C++ for CPUs and GPUs and placed in the fPCE and bPCE modules, respectively. Several bulk path computation requests were created by the fPCE. When the fPCE is in charge of performing the bulk computation, one single CPU thread is used; therefore, iterations of the above heuristic algorithm are performed sequentially in the fPCE.

In contrast, when the fPCE delegates the computation to the bPCE (Fig. 4-4a), the former opens a new PCEP session, creates a PCReq message (Fig. 4-4b) containing every path request and a SVEC object, and forwards the message to the bPCE.

Upon receiving a PCReq message, the HP bPCE uses the GPU device to accelerate the computation; each iteration is executed into an independent thread block in the GPU, and all the threads inside a block collaborate in solving the RSA problem in parallel for each demand in the bulk following the defined order. The bPCE sends a PCRep message (Fig. 4-4c) containing the route and spectrum allocation for each demand or the object NO_PATH when no feasible solution was found for a demand. Due to GPU's architecture, huge amount of blocks must be computed to achieve the peak performance, so we expect better results when the number of iterations is high.

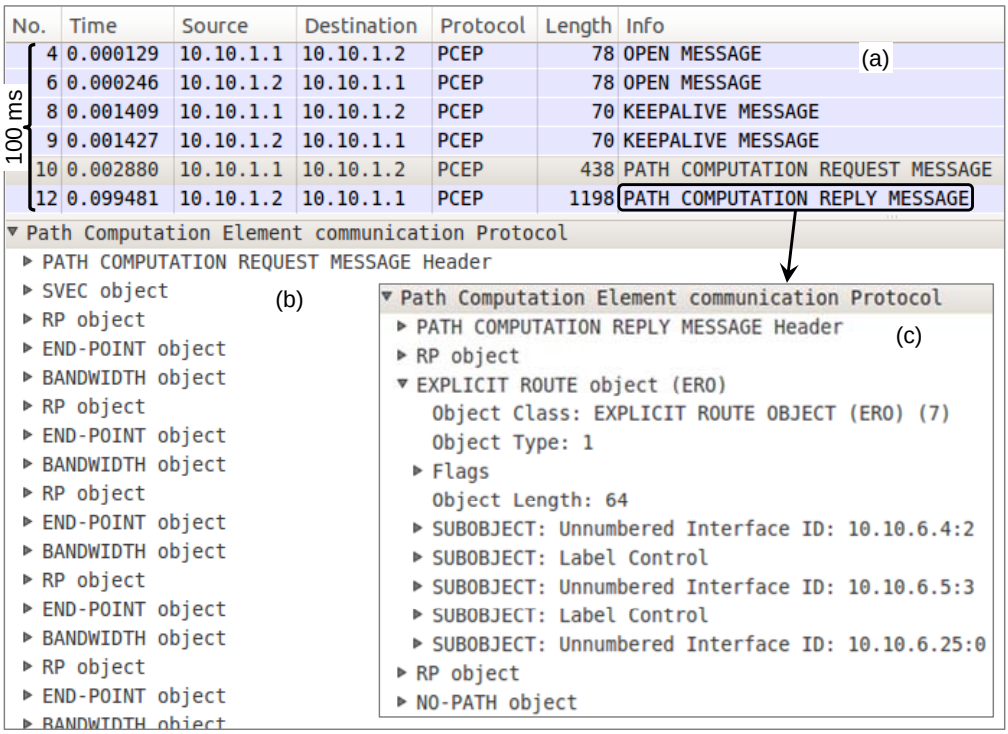


Fig. 4-4 Exchanged messages and details.

We assume that the time available for bulk computation is 100ms. Therefore, that is the maximum time in the case the computation is performed in the fPCE's CPU. Nonetheless, since PCEP time, including PCEP session opening and PCReq and

PCRep transmission, is in the order of 2ms on average, the available computation time in the case the bPCE performs the computation is limited to 98ms.

A large number of bulk path computations were performed by both the fPCE and the bPCE for each of the considered networks. Several bulk sizes and number of iterations were also considered. In addition, each bulk was solved 10 times and so, average figures are provided.

Fig. 4-5a plots the solving times for bulks of 50 demands on the BT network computed on the fPCE's CPU (dotted line) and on the bPCE's CPU+GPU (solid line). As observed, 120 heuristic iterations can be done within 100ms when the computation is performed in the fPCE, whereas 990 iterations are done in the bPCE within 98ms (i.e. 8.4x speedup). Similarly, 6.7x speedups are shown for the TEL network topology (Fig. 4-5b). The jerk-like behavior of GPU plots are as a consequence of the amount of blocks that need to be allocated in the GPU.

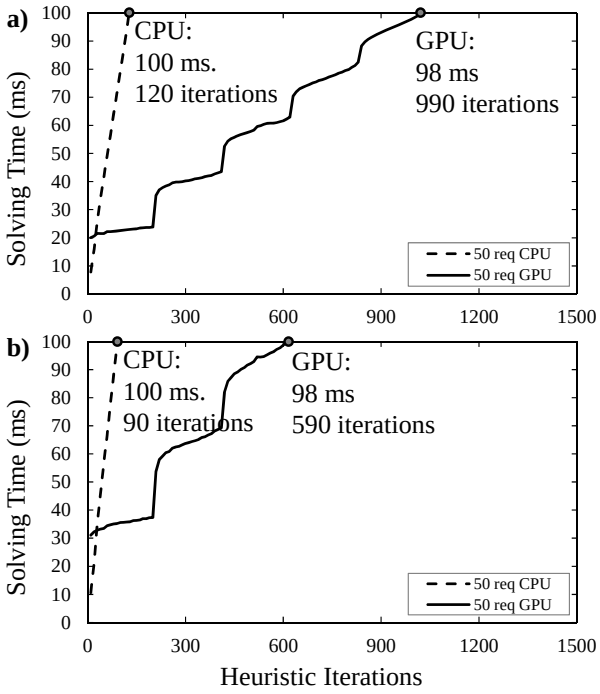


Fig. 4-5 Solving times for BT (a) and TEL (b) networks.

It is clear that the larger the number of iterations done, the better the quality of the solution. Aiming at studying that quality, we have evaluated the probability of obtaining an optimal solution as a function of the number of iterations computed. In the case of the BT network (Fig. 4-6a), the probability of finding the optimal solution with the fPCE is relatively high with only 120 iterations (78.5%), although the bPCE reaches 91.5% of probability as a result of computing 990 iterations. For the TEL network (Fig. 4-6b), however, the obtained probabilities are lower as a

result of its larger size. Notwithstanding that, the bPCE reaches 85.9% of probability in contrast to the poor 55.8% obtained when the fPCE was used.

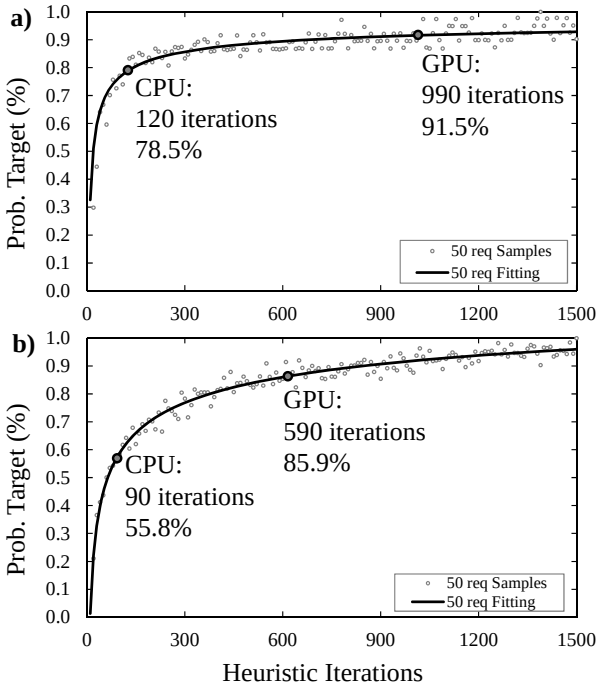


Fig. 4-6 Solution quality for BT (a) and TEL (b) networks.

Table 4-2 details the number of iterations and solving times to obtain 90% of probability of finding an optimal solution for three different bulk sizes. As observed, when both the size of the bulk and the size of the network are low, the front end PCE is able to compute the amount of iterations needed to reach 90% of probability of optimality. However, as soon as either the size of the bulk or the topology increases, the fPCE cannot compute the number of iterations required within 100 ms. Note that bulk sizes in the order of 40-50 demands are common in restoration and re-optimization.

Table 4-2 Solving times to reach 90% probability of optimality

	BT network topology			**TEL network topology**		
Bulk Size	**# Iter.**	**Time (ms)**		**# Iter.**	**Time (ms)**	
		fPCE	**bPCE**		**fPCE**	**bPCE**
30	40	21.6	11.5	390	248.4	38.5
40	120	72.4	16.9	570	466.3	65.4
50	680	514.3	74.6	850	889.3	142.6

4.2 Database Synchronization

As introduced in Chapter 2, the PCE stores the current state of the network resources and LSPs on its local copies of the TED and the LSP-DB, respectively. However, the bPCE needs also access those data since they are input parameters for the in-operation algorithms. This requires deploying a local TED and LSP-DB and using protocols to synchronize databases with the fPCE. A solution consists in placing BGP speakers at each PCE to perform the TED synchronization; BGP Update (*BGPUpd*) messages are used to notify the availability (*BGPUpd* – Reach) of new nodes and links and its unavailability (*BGPUpd* Un-Reach). Additionally, PCEP Report (*PCRpt*) messages are used for the LSP-DB synchronization.

To illustrate the database synchronization between fPCE and bPCE, let us use the example of a link failure and link repair conditions. The fPCE/bPCE protocol exchange for TEDs and LSPDBs synchronization is illustrated in Fig. 4-7.

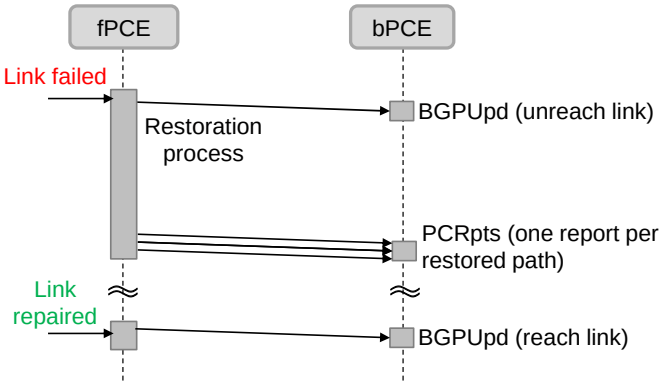


Fig. 4-7 Exchanged PCEP and BGP-LS messages.

When a link failure is received in the fPCE, it sends an unreach message to the bPCE to notify such condition, and the fPCE continues with its restoration process and notifies the bPCE with the new routes for the restored LSPs by means of report messages. Later, once the link is repaired, the fPCE notifies the bPCE using a reach message.

The experimental validation of the database synchronization procedures was carried out in a distributed test-bed connecting CTTC's ADRENALINE testbed (Castelldefels, Spain) with UPC premises by means of an IPSec tunnel (Fig. 4-8). The ADRENALINE testbed is composed of a stateful fPCE co-located with the GMPLS controller. At UPC premises, the PLATON bPCE was in charge of running the re-optimization algorithm.

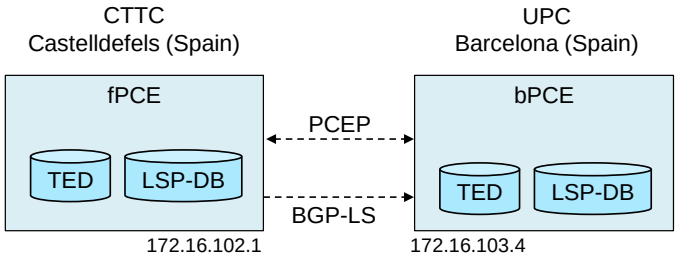


Fig. 4-8 Experimental set-up.

Fig. 4-9 shows relevant BGP-LS and PCEP messages exchanged between the fPCE and the bPCE. When the fPCE is notified about the link failure, it notifies that condition to the bPCE (labelled as 1 in Fig. 4-9) and requests the computation of the new routes, e.g., by issuing a bulk path computation request to the bPCE and waits for the bPCE reply as described in section 4.1 (2). When the bPCE replies, the fPCE sets-up the new LSPs and issues report messages to the bPCE with the new routes (3). Later, when the link is replaired, the fPCE notifies the bPCE the new state of the links (4).

	Source	Destin	Proto	Info
	CTRL	fPCE	PCEP	Path Computation Request (PCReq)
①	fPCE	bPCE	BGP	UPDATE Message
	fPCE	bPCE	BGP	UPDATE Message
②	fPCE	bPCE	PCEP	Path Computation Request (PCReq)
	bPCE	fPCE	PCEP	Path Computation Reply (PCRep)
③	fPCE	bPCE	PCEP	Path Computation LSP State Report (PCRpt)
	fPCE	bPCE	PCEP	Path Computation LSP State Report (PCRpt)
	fPCE	bPCE	PCEP	Path Computation LSP State Report (PCRpt)
	fPCE	CTRL	PCEP	Path Computation Reply (PCRep)
④	fPCE	bPCE	BGP	UPDATE Message
	fPCE	bPCE	BGP	UPDATE Message

Fig. 4-9 Exchanged BGP-LS and PCEP messages.

4.3 iONE Architecture

In view of the previous sections, we decided to develop our own implementation of the ABNO architecture to support our experiments. This section, presents the architecture of the iONE software, a module specifically designed to facilitate workflow definition and deployment starting from the architecture of the bPCE module presented in section 4.1, that consisted of a communications server that included PCEP and BGP-LS protocols, the TED and LSP-DB local databases, and a module for the planning algorithms (see Fig. 4-1).

This architecture need to be extended to improve the control of the several tasks that the bPCE can perform simultaneously and to include new communication interfaces, such as REST API-based protocols. In that way, the resulting module,

named as iONE, can be adapted very easily so as to behave as any component of the ABNO architecture. In fact, to facilitate the adaptation of the iONE module, we based all the operations inside the module on a number of defined workflows.

Therefore, the architecture of the iONE module, depicted in Fig. 4-10, consists of five components: the communication interfaces, the manager, the network databases, the algorithms framework, and the workflow engine.

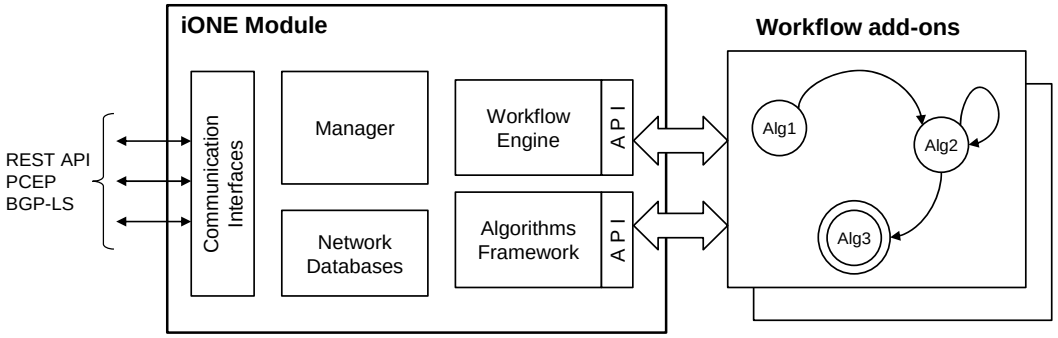


Fig. 4-10 Architecture of the iONE Module.

iONE has been implemented as one single generic software module developed in C/C++, which can be configured using an eXtended Markup Language (XML) file to implement many of the ABNO modules. Furthermore, workflows can be defined through XML files, where the algorithms to be executed are dynamically loaded into the iONE module.

The communication interfaces are the gateway for all incoming and outgoing messages; these messages can be encoded using REST API, PCEP, and BGP-LS protocols. iONE's implementation of PCEP supports both active stateful [draft-stateful] and LSP initiate capabilities [draft-initiate]. Besides, the REST API interfaces support both XML and JavaScript Object Notation (JSON)-encoded messages. An arbitrary number of interfaces for each protocol can be configured using a XML file.

The manager is responsible for configuring and coordinating the rest of components; it collects incoming messages received by any communication interface and either redirects them to the proper component or uses them to update databases.

The network databases component provides in- memory storage for the TED and the LSP-DB. The databases can be pre-loaded by means of XML files and configured to synchronize their content through specific communication interfaces. For instance, a BGP-LS interface can be configured to handle *BGPUpd* messages to synchronize the TED, while a PCEP interface can synchronize the state of the LSP-DB using asynchronous *PCRpt* messages.

The algorithms framework provides common functionalities for agile algorithm development, including routing algorithms, such as the RSA [Ve14-2] algorithm,

optical spectrum handling functions, metaheuristic frameworks, and many more. Besides, protocol helper functions for rapid message handling are provided. All those functions are available through an API.

Workflows are implemented as finite state machines (FSM). Two elements are needed to add a new workflow to an iONE module: a dynamically linkable file containing a number of algorithms and a XML file defining the workflow. Since workflows are implemented as FSMs, a number of states and transitions between states are defined in the XML file; a different algorithm is executed in each state of the workflow.

The workflow engine is responsible for initiating new workflow instances, and coordinating their execution subject to incoming messages. The engine contains a message table storing identifiers of messages pending to be received and the workflow instance awaiting each message. Workflows, and their state algorithms, use the algorithms framework and workflow engine APIs to interact with the iONE module.

Each workflow is configured through an XML file that contains:

i) the triggering message to initiate the workflow,

ii) the set of states,

iii) the set of transitions between states based on incoming message types,

iv) the initial state of the FSM,

v) the path to a dynamic linked library implementing the algorithms to be run on each state of the FSM, and

vi) a workflow state algorithms configuration file to define parameters and constants.

Incoming *RESTReq*, *PCReq*, *PCUpd*, and *PCInitiate* messages trigger workflow instances creation. When RESTReq messages are received, the REST API resource path that user specified to issue the request is used to select the desired workflow. Besides, an `OF` object can be included in PCReq messages to specify the desired workflow to be run; a default workflow must be defined for those PCReq messages without any `OF` object.

When a workflow is started (Fig. 4-11), a workflow instance is created by the workflow engine containing the current state of its FSM and internal workflow data, thus enabling workflow concurrently.

To select the appropriate function in the dynamic library implementing a given workflow state algorithm, its name must be the same as the name of the state. Workflow state algorithms can execute any generic algorithm and send messages to other modules using functions in the APIs. At the end of the state algorithm execution, the list of pending messages to be received is reported to the workflow

engine, which updates the messages table with the expected identifiers in RP and Stateful PCE RP (SRP) objects. If no pending messages are reported, the workflow automatically finishes it execution.

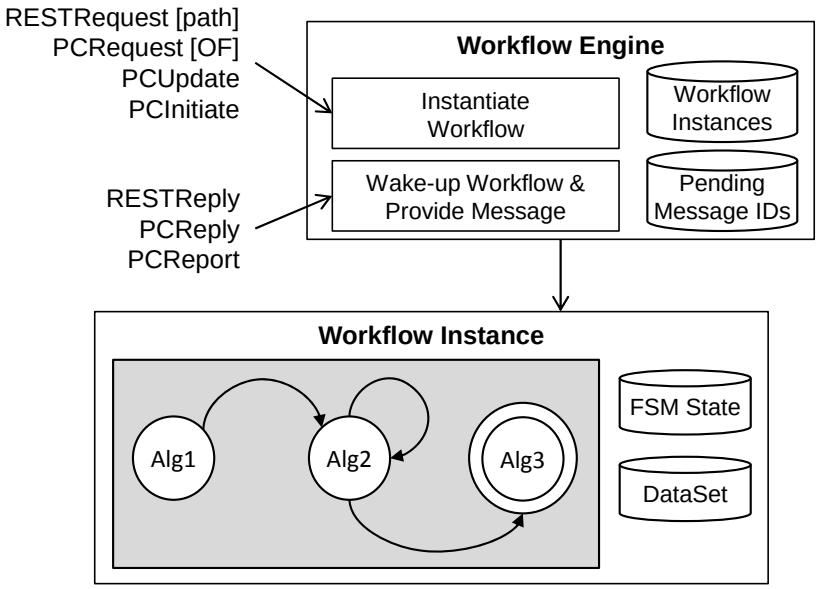


Fig. 4-11 Workflow instantiation.

As previously anticipated, the iONE module can be configured to behave as any component of the ABNO architecture. See Appendix A for a detailed example. In the rest of this thesis, however, the iONE module is, in principle, used as a bPCE and supports our planning tool for optical networks, named as PLATON.

4.4 Conclusions

In this chapter, we proposed an architecture in support of in-operation planning. We first studied the performance of splitting the centralized PCE in the ABNO architecture in a fPCE / bPCE scheme. While the fPCE is in charge of path provisioning and network events handling, the bPCE is specialized in solving re-optimization problems.

We implemented such a specialized HP GPU-based bPCE to compute re-optimization in Flexgrid networks. Experimental results showed 6x speedups compared to fPCE CPU running times, resulting in an increased probability of finding optimal solutions.

Then, we studied the synchronization of TED and LSP-DB databases between the fPCE and the bPCE. We demonstrated the necessity of extending the PCEP protocol to allow carrying the Symbolic Path Name TLV in the RP object for

scalability purposes. Finally, we demonstrated the feasibility of fPCE/bPCE database synchronization.

Once the fPCE/bPCE coordination has been demonstrated, the iONE implementation was presented; our PLATON tool is based on the iONE module configured as a bPCE.

The next chapters focus on the applications of in-operation planning in flexgrid networks.

Chapter 5

Spectrum Shifting

Dynamic operation of flexgrid networks might cause optical spectrum to be divided into fragments, which makes difficult finding contiguous spectrum of the required width for incoming connection requests, leading thus to an increased blocking probability. To alleviate to some extent that spectrum fragmentation, the central frequency of already established connections can be shifted to create wider contiguous spectrum fragments to be allocated to incoming connections; this procedure is called as hitless spectrum defragmentation. In this chapter, we study the spectrum shifting problem for hitless spectrum defragmentation and propose using the ABNO architecture to deal with spectrum defragmentation while the network is in operation. A workflow involving several elements in the ABNO architecture is proposed and experimentally assessed.

5.1 Introduction

The spectrum defragmentation problem can be addressed using *spectrum reallocation*, where the set of LSPs to be moved are first released and then reallocated using a different frequency slot. Note that original and final frequency slots can be overlapped, thus preventing to use a make-before-break procedure. Hence, even when spectrum reallocation brings the maximum flexibility to the spectrum defragmentation process, in such cases where make-before-break cannot be applied, traffic will suffer of disruptions during the re-optimization.

Spectrum defragmentation can be also addressed using *spectrum shifting*, that consists in moving the in-use frequency slot between contiguously free portions of the spectrum. Provided that spectrum shifting eliminates the need of releasing the LSPs during their reconfiguration, traffic disruption problem is mitigated.

5.2 The Spectrum Shifting Problem

As discussed in the previous section, to avoid traffic disruption we propose using spectrum shifting for defragmentation. To that end, in this section we firstly formally state the spectrum shifting (SPRING) problem and then, present an ILP model to solve the defragmentation problem. The model is based on the work in [Cast12-1], but adapted with specific constraints to ensure that connections are shifted and not reallocated, i.e. no other connections are established using slices between current and new slots. Finally, a heuristic algorithm providing much better trade-off between optimality and complexity is proposed.

5.2.1 Problem Statement

The SPRING problem can be formally stated as follows:

Given:

- an optical network, represented by a graph G (N, E), being N the set of nodes and E the set of optical links connecting two nodes,
- a set S of frequency slices available in each link $e \epsilon E$,
- a set P of already established LSPs,
- a new request (*newP*) to be established in the network. A route for the LSP has been already selected but there is no feasible spectrum allocation,

Output:

- for each LSP to be shifted, its new spectrum allocation,
- the spectrum allocation for *newP*.

Objective: Minimize the amount of LSPs to be shifted to fit *newP* in.

A MILP model for the SPRING problem is presented next.

5.2.2 MILP Formulation

The MILP model takes as input the set of established LSPs that share common links with the shortest route for *newP*.

The following sets and parameters are defined:

E	set of optical links, index e.
P	set of already established LSPs, index p.
$E(p)$	subset of E with those links in the route of LSP p.
$P(e)$	subset of P with those LSPs using optical link e.

$P(p)$	subset of P with those LSPs sharing at least one link with LSP p. $P(p) = \cup_{e \in E(p)} P(e)$
Pm	subset of P with the candidate LSPs. $Pm = P(newP)$.
S	set of frequency slices, index s.
C	set of slots, index c. Each slot c contains a subset of contiguous slices.
$C(p)$	subset of C with those slots that can be allocated to p.
δ_{cs}	1 if slot c uses slice s, 0 otherwise.
ω_{pc}	1 if LSP p was using slot c, 0 otherwise.
η_{es}	1 if slice s in optical link e is free, 0 otherwise. Note that to compute η_{es} only LSPs in $P \setminus Pm$ are considered.
$\beta_{pp'}$	0 if originally the index of the first slice allocated to LSP p was lower than that of the first slice allocated to LSP p', provided that p' was sharing at least one link with LSP p, i.e. $c<c'$, $c \in C(p)$, $c' \in C(p')$, $p' \in P(p)$. 1 if $c>c'$. Note that $\beta_{pp'}$ is not defined for those LSPs not sharing any link.

Decision variables:

x_{pc}	binary, 1 if slot c is allocated to LSP p, 0 otherwise.
y_p	binary, 1 if LSP p is shifted, 0 otherwise.

Then, the formulation for the SPRING problem is as follows:

$$\text{(SPRING)} \qquad \min \sum_{p \in Pm} y_p \tag{5.1}$$

subject to:

$$\sum_{c \in C(p)} x_{pc} = 1 \quad \forall p \in Pm \cup \{newP\} \tag{5.2}$$

$$x_{pc} - \omega_{pc} \le y_p \quad \forall p \in Pm, c \in C(p) \tag{5.3}$$

$$\sum_{p \in P(e)} \sum_{c \in C(p)} \delta_{cs} \cdot x_{pc} \le \eta_{es} \quad \forall e \in E, s \in S \tag{5.4}$$

$$\sum_{c' \in C(p')} c' \cdot x_{p'c'} - \sum_{c \in C(p)} c \cdot x_{pc} < \beta_{pp'} \cdot |S| \quad \forall p \in Pm,\ p' \in P(p) \tag{5.5}$$

$$\sum_{c' \in C(p')} c' \cdot x_{p'c'} - \sum_{c \in C(p)} c \cdot x_{pc} > (\beta_{pp'} - 1) \cdot |S| \quad \forall p \in Pm,\ p' \in P(p) \tag{5.6}$$

The objective function (5.1) minimizes the number of LSPs that need to be shifted in the spectrum so *newP* can be served. Constraint (5.2) ensures that every candidate LSP and the received request for *newP* have one slot allocated.

Constraint (5.3) stores whether LSP p needs to be shifted comparing current and assigned slots. Constraint (5.4) guarantees that each frequency slice in every optical link is assigned to one path at most. Constraints (5.5) and (5.6) ensure feasible shifting by taking care of relative spectral position between LSPs pairs sharing any link, i.e. if two LSPs p and p' sharing a common link were originally allocated to slots with indexes $c<c'$ ($\beta_{pp'} = 0$), then that relative spectral position mush be kept in the solution. The same must be ensured when $c>c'$ ($\beta_{pp'} = 1$). Note that when LSPs p and p' do not share any link, these constraints do not apply.

Although the size of the problem is limited –the number of variables is $O(|Pm| \cdot |C|)$ and the number of constraints is $O(|Pm|^2 + |Pm| \cdot |C| + |E| \cdot |S|)$– it must be solved in real time, e.g. tens milliseconds, to minimize set-up delay of *newP*. For this very reason, we propose an heuristic algorithm in next section.

5.2.3 Heuristic Algorithm

In this section, we propose to use the heuristic algorithm described in Table 5-1 to solve the Spectrum Shifting problem. The algorithm iterates on every frequency slice s to find the set of LSPs in the route of *newP* allocated using the closest slice with index lower than s (s^-), the set of LSPs allocated using the closest slice with index greater than s (s^+) and the set of LSPs allocated using s (lines 3-10 in Table 5-1).

Table 5-1 Algorithm for the SPRING problem

INPUT E, $newP$, n_d
OUTPUT $Solution$

```
 1: Solution ← ∅
 2: numLSPs ← INFINITE
 3: for s = 1..|S| do
 4:   P+ ← ∅; P- ← ∅; Ps ← ∅
 5:   for each e ∈ E(newP) do
 6:     if p(e, s-1) ≠ p(e, s) then
 7:       P- ← P- ∪ { p(e, s-) }
 8:       P+ ← P+ ∪ { p(e, s+) }
 9:     else
10:       Ps ← Ps ∪ { p(e, s) }
11:   { shift, conn } ← getMaxShift(P-, Ps, P+, s)
12:   if shift ≥ nd AND conn < numLSPs then
13:     Solution.P- ← P-
14:     Solution.Ps ← Ps
15:     Solution.P+ ← P+
16:     Solution.s ← s
17:     numLSPs ← conn
18: return Solution
```

Procedure *getMaxShift* find the largest continuous slot that can be generated by shifting LSPs (line 11). LSPs in set P^- are left shifted, LSPs in P^+ are right shifted, and LSPs in P^s are shifted left and right and the option generating the widest slot is chosen. If a slot with, at least, n_d contiguous slices by shifting LSPs is found and

the set of LPSs involved is lower than that of the best solution found so far, the set of LSPs is stored as the best solution and the number of LSPs involved is updated (lines 12-17). The best solution found is eventually returned.

The next section describes our proposal to integrate SPRING into the control plane using the ABNO architecture.

5.3 Proposed Workflow and Algorithm

From a control and management perspective, the defragmentation process maps into a set of state changes of active connections. Such state changes are reflected in the change of connections' attributes, in our case spectrum allocation by shifting the nominal central frequency of the slot allocated to a connection.

The optimization process can be triggered either manually by a network operator through the NMS, by an automated process triggered by some threshold or in a periodical fashion. In our case, let us assume that the defragmentation procedure is triggered after the fPCE fails to find a suitable route for a provisioning request.

The request for a new connection is originally issued by the NMS and received by the ABNO controller through the north-bound interface. In such case, the ABNO controller is responsible for coordinating connection set-up, which composes and sends a specific request towards the fPCE, in charge of computing and finally coordinate connection establishment.

The workflow that represents the provisioning-triggered defragmentation use case is detailed in Fig. 5-1. As already introduced, it starts with a network operator requesting a new connection provisioning through the NMS. Establishing a flexible optical connection includes computing and provisioning a continuous slot between two nodes in the data plane. The request is received by the ABNO controller via its north-bound interface (step 1 in Fig. 5-1). When the ABNO controller receives the request it asks the policy agent to check about rights of the received request (2). If access is granted, the ABNO controller requests the fPCE to compute the route and eventually set up the optical connection (3).

Let us assume that as a result of spectrum fragmentation no end-to-end continuous slot is found (4). In that case, the ABNO controller may autonomously decide to perform a defragmentation process and sends a message to the fPCE (5). When the fPCE receives the request for defragmentation, it checks its feasibility and gathers information to create a GCO request that is sent towards the bPCE to solve the optimization problem (6). When the bPCE ends, it sends back the solution found. The fPCE proceeds then to execute the defragmentation that consists in shifting some of the candidate LSPs (7) and finally, when every LSP has been updated, the fPCE proceeds to establish the requested connection (8). Upon its completion, the fPCE notifies the ABNO controller (9), which in turn notifies the NMS (10).

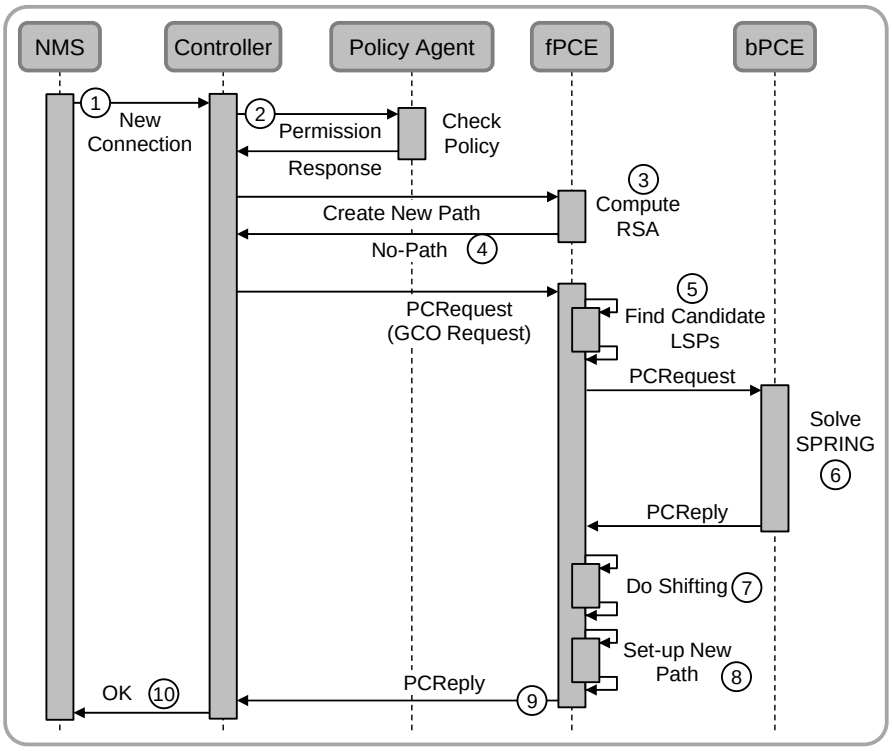


Fig. 5-1 Defragmentation workflow.

Obviously, if the request in (3) finds a feasible route and spectrum allocation, the fPCE proceeds with step (8) to establish the connection.

All interactions between ABNO and PCEs are done by exchanging PCEP messages. In particular, PCReq and PCRep messages are exchanged between fPCE and bPCE (step 6 in Fig. 5-1).

Let us now focus on the steps performed once a request arrives at the fPCE (step 3 in Fig. 5-1). From that point on, the distributed algorithm shown in Fig. 5-2 is executed. To facilitate the explanation, steps in Fig. 5-2 relate to those in Fig. 5-1.

After a request arrives at the fPCE, an RSA algorithm is run to compute the route and find a feasible spectrum allocation. Assuming that no end-to-end continuous slot is found (step 4), the ABNO controller decides to perform a defragmentation process and sends a PCReq message to the PCE (step 5). When the PCE receives the request for defragmentation, it finds the set of already established LSPs candidate to be part of the defragmentation process (Table 5-2). The procedure computes the shortest path between source and destination nodes of the requested connection, and the requested slot width (lines 2-3 in Table 5-2). Next, it verifies that every link in the found route has enough available spectrum resources and creates the set of candidate LSPs (*P*) as the LSPs using any of those links (lines 4-7). Both, the route and the set of LSPs are eventually returned (line 8).

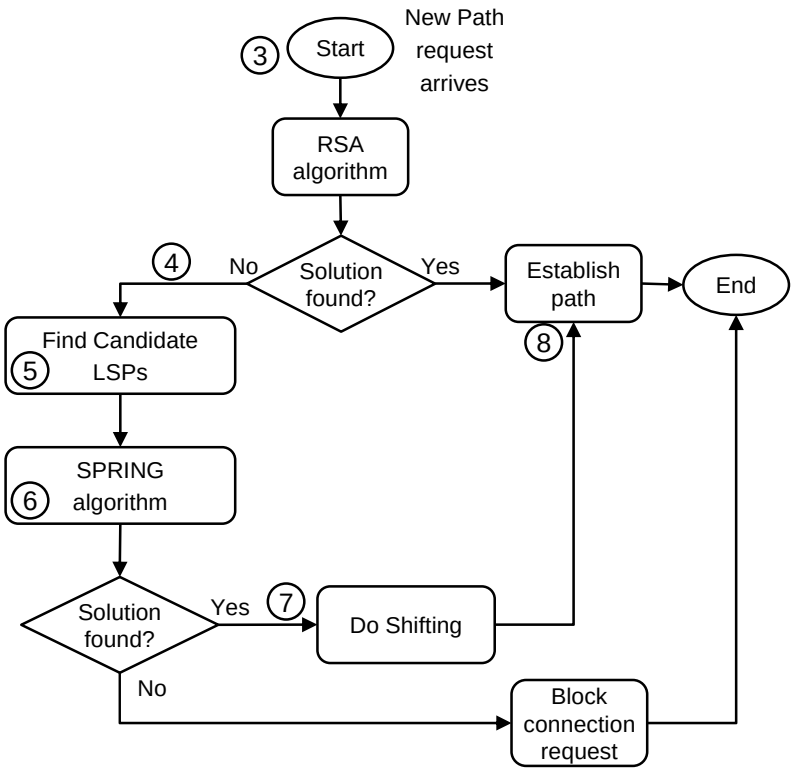


Fig. 5-2 Defragmentation distributed algorithm.

Table 5-2 Procedure Find_Candidate_LSPs

```
INPUT TED, LSP-DB, sliceWidth, source, destination, bitrate
OUTPUT P, R
1:  P ← Ø
2:  newP ← shortestPathByLenght(TED, source, destination)
3:  n_d ← computeSlotSize(R, bitrate, sliceWidth)
4:  for each e ∈ E(newP) do
5:    if freeSlices(e) < n_d then
6:      return { Ø, Ø }
7:    P ← P ∪ getLSPs(LSP-DB, e)
8:  return < P, newP, n_d >
9:
```

Once the set of candidate LSP is found, the PCE sends a PCReq message to the bPCE to solve the optimization problem (step 6). When the bPCE ends, it sends back a PCRep message detailing the solution found. The fPCE proceeds then to execute the defragmentation that consists in shifting some of the candidate LSPs. To that end, a PCUpd message is sent towards each ingress Path Computation Clients (PCC) of a LSP that needs to be shifted (step 7); PCCs report updating completion back using PCRpt messages. Finally, when every LSP has been updated, the PCE proceeds to establish the requested connection by sending a PCInit message to the source PCC (step 8).

5.4 Experimental Assessment

In this section, we first define the specific contents and semantic of PCReq and PCRep messages to deal with defragmentation. As stated in Chapter 4, we assume in this thesis that a bPCE is in charge for solving optimization problems, so the SPRING algorithm is implemented in that bPCE and can be invoked from the fPCE using a specific code in OF objects.

Next, we experimentally validate our proposal for in-operation spectrum defragmentation.

5.4.1 PCEP Issues

Owing to the fact that to solve SPRING the route and the set of candidate LSPs needs to be specified, that input data has to be coded and sent in a PCReq message. Table 5-3 specifies the contents to convey the needed parameters.

Table 5-3 PCReq message contents

```
<PCReq Message>::= <Common Header>
                   <SVEC>
                   <OF>
                   <re_opt request-list>
                   <request>
where:
   <re_opt request-list>::= <re_opt request>[<re_opt request-list>]

   <re_opt request>::= <RP>
                       <END-POINTS>
                       <RRO>

   <request>::= <RP>
                <END-POINTS>
                <BANDWIDTH>
                <IRO>
```

A list named `re_opt request-list` contains the candidate LSPs. The information includes source and destination nodes for the LSP (`END-POINTS` object) and its current route and spectrum allocation in the `RRO` object. With that information, the bPCE can find the associated LSP in its LSP-DB. Information regarding the new connection requested follows, including source and destination nodes, requested bitrate (`BANDWIDTH` object) and computed route (`IRO` object).

Since all the requests have to be jointly considered, they are grouped by a `SVEC` object. Finally, the PCReq message uses the `OF` object to specify the objective function requested; spectrum defragmentation in our case.

Once the SPRING problem is solved, the solution is coded in a PCRep message. Table 5-4 specifies the contents to convey the solution.

Table 5-4 PCRep message contents

```
<PCRep Message> ::= <Common Header>
                    <response-list>
where:
   <response-list>::= <response>[<response-list>]

   <response>::= <RP>
                 <NO-PATH> | <ERO>
```

Since all the received requests in a PCReq message have to be replied, the `response-list` contains an `ERO` object of each of the re-optimization requests. Altough the route in the `ERO` will be invariant, since no rerouting is performed, the spectrum allocation can be either the same or a different one. Hence, an algorithm needs to be used in the fPCE to find the LSPs to be updated; the algorithm is detailed in Table 5-5.

Table 5-5 Procedure Do_Shifting

```
INPUT LSP-DB, P
1: for each p ∈ P do
2:   if p.ERO ≠ getERO(LSP_DB, p) then
3:     update(p)
4: return
```

Regarding the request for the new connection, an `ERO` object can be received provided that a feasible solution for the SPRING problem has beed found. Note that the `ERO` object must follow the route specified in the incoming `IRO` object. A `NO-PATH` object is included if no feasible solution is found.

The proposed algorithm for the SPRING problem and the described workflow have been experimentally validated in a distributed test-bed connecting infrastructures in Madrid and Barcelona (Spain), and Pisa (Italy). The next section describes the scenario and the tests performed.

5.4.2 Experiments

Scenario

The set-up has been deployed in three locations: Madrid, Barcelona and Pisa (Fig. 5-3). The ABNO controller and the NMS are located in Telefonica's premises, while the bPCE runs in UPC. The data plane, the GMPLS nodes and the fPCE are in CNIT's premises. The ABNO controller, the fPCE and the bPCE communicate through PCEP interfaces, where sessions are established on top of IPSec tunnels.

The CNIT fPCE has been extended to enable back-end computation, functionally separated from PCEP sessions established locally with the data plane nodes. The Path Solver has been extended with the *Find_Candidate_LSPs* procedure of Table 5-2.

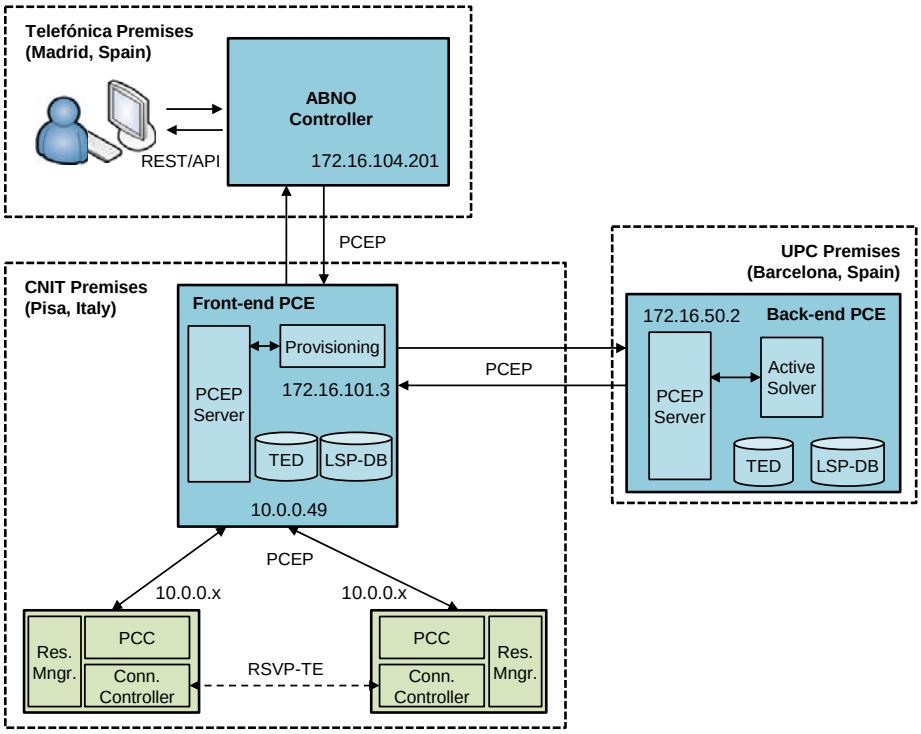


Fig. 5-3 Distributed test-bed set-up.

The data plane includes four programmable spectrum selective switches (SSS); to complete the network topology, four node emulators are additionally deployed. Nodes are handled by co-located GMPLS controllers running RSVP-TE with flexgrid and push-pull operation extensions [Cu13-1]. GMPLS controllers communicate with the fPCE by means of PCEP. The GMPLS controllers connected to a SSS (by means of a USB interface) runs a dedicated programmable configuration tool for automatic filter re-shaping with a resolution of 1 GHz.

Finally, PLATON, based on the iONE module defined in Chapter 4, was used.

5.4.2.1 Results

Aiming at illustrating the spectrum defragmentation problem, Fig. 5-4 shows an example, reproduced from [Cast12-1], on the small network topology depicted in Fig. 5-4a, where each node and link is labeled. The entire spectrum width consists of 16 frequency slices. Fig. 5-4b represents the utilization of each frequency slice in the network, where a number of optical connections are already established.

In this scenario, the lack of consecutive optical spectrum resources prevents serving the connection request between nodes 4 and 7 requiring 4 slices. Notwithstanding, each link in the shortest route of the new optical connection *newP* (through links 4-5-6) has at least 4 free slices and then, the request could be established shifting some of the established connections.

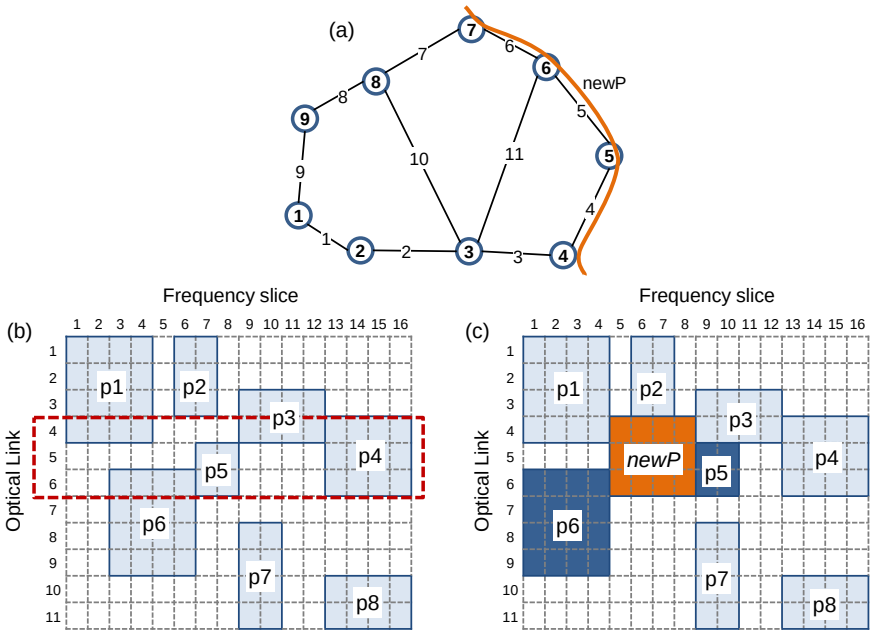


Fig. 5-4 Example of spectrum defragmentation (reproduced from [Cast12-1]).

In the example, connections *p1*, *p3*, *p4*, *p5*, and *p6* are using one or more of the links in the computed shortest route, and thus can be considered as candidates to be part of the defragmentation process. Finally in Fig. 5-4c, connections *p4* and *p5* have been shifted making enough room for *newP*.

Before running the experiments, local TED and LSP-DB databases where populated with the same configuration. In particular, the topology and LSPs shown in Fig. 5-4 were loaded locally.

Next, a request for a new connection between nodes 4 and 7 was issued from the NMS to the ABNO controller, which resulted in the PCEP exchange depicted in the capture in Fig. 5-5. For the sake of easily follow the subsequent discussion messages are correlated to the steps described in the workflow in Fig. 5-1. The capture, done at the fPCE, includes PCEP messages exchanged inside the ABNO architecture (IP sub-network 172.16.x.x) as well as those exchanges between the fPCE and the PCCs (IP sub-network 10.10.x.x).

Fig. 5-6 presents the contents of the PCEP messages exchanged between the ABNO controller and the fPCE. PCReq (message 3) includes the IP address of the end nodes and the requested bitrate (50Gb/s). When no solution is found in the fPCE for the request, the PCRep (message 4) including a `NO-PATH` object is sent back to the ABNO controller.

Next, PCReq (messages 5 in Fig. 5-7) includes an `OF` object with the objective function code “defragmentation”. When the defragmentation process finishes, the fPCE replies with PCRep (message 9) that includes the route and spectrum

allocated for the new connection. In our experiments, we defined as reference frequency that between slices 8 and 9, so the new connection has been established using slices 5-8 (n=-2, m=2).

	No.	Time	Source	Destination	Prot	Le	Info	
3	56	181.0738	172.16.104.201	172.16.101.3	PCEP	110	PATH COMPUTATION REQUEST MESSAGE	
4	58	181.0739	172.16.101.3	172.16.104.201	PCEP	98	PATH COMPUTATION REPLY MESSAGE	
5	60	181.1461	172.16.104.201	172.16.101.3	PCEP	110	PATH COMPUTATION REQUEST MESSAGE	
6	61	181.1471	172.16.101.3	172.16.50.2	PCEP	730	PATH COMPUTATION REQUEST MESSAGE	
	66	181.2823	172.16.50.2	172.16.101.3	PCEP	574	PATH COMPUTATION REPLY MESSAGE	
7	68	181.3027	10.0.0.49	10.0.0.5	PCEP	78	UPDATE MESSAGE	LSP5
	69	181.3057	10.0.0.5	10.0.0.49	PCEP	78	REPORT MESSAGE	LSP5
	72	183.5327	10.0.0.49	10.0.0.6	PCEP	78	UPDATE MESSAGE	LSP6
	73	183.5356	10.0.0.6	10.0.0.49	PCEP	78	REPORT MESSAGE	LSP6
8	82	192.4523	10.0.0.49	10.0.0.4	PCEP	194	INITIATE MESSAGE	newP
	83	192.4555	10.0.0.4	10.0.0.49	PCEP	78	REPORT MESSAGE	newP
9	85	195.4268	172.16.101.3	172.16.104.201	PCEP	190	PATH COMPUTATION REPLY MESSAGE	

Fig. 5-5 PCEP messages exchanged.

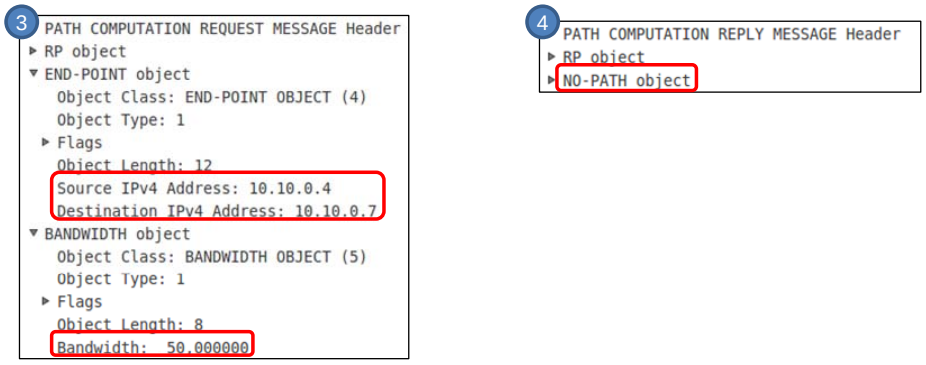


Fig. 5-6 PCEP messages 3,4.

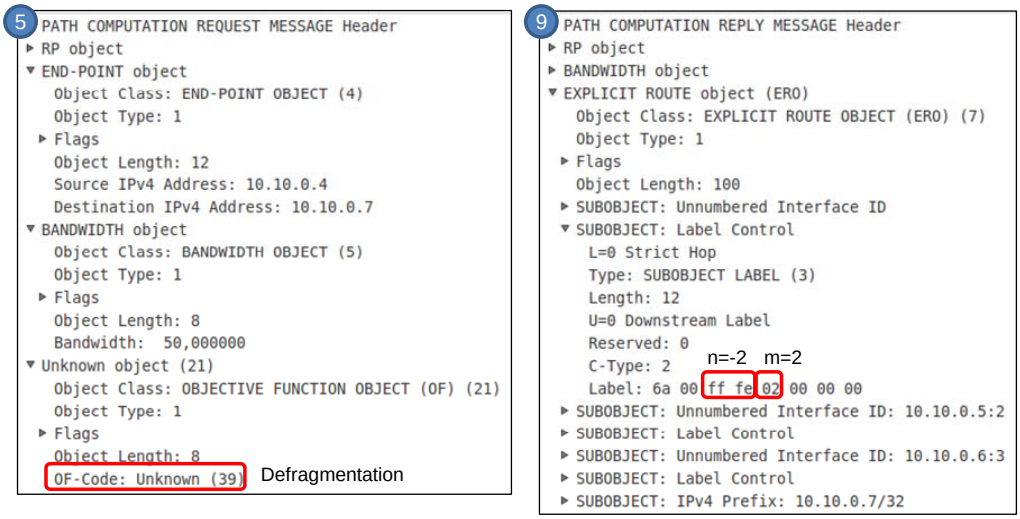


Fig. 5-7 PCEP messages 5,9.

Finally, Fig. 5-8 shows the PCEP messages exchanged between fPCE and bPCE. The fPCE computes the set of candidate LSPs that will participate in the optimization process, and includes them into PCReq (message 6), indicating the current route and spectrum allocation. For instance, the RRO object for LSP 5 is detailed in Fig. 5-8. Together with the candidate LSPs, the request for the new connection is included, constrained to the route specified in the IRO object.

After the optimization problem has been solved, the bPCE sends back to the fPCE the PCRep (message 6) shown in Fig. 5-8, where an ERO object is included in response to every already established LSP. To illustrate the defragmentation, we can observe that the original spectrum allocation for LSP5 was n=-1 and m=1 (slices 7-8) and the computed one was n=+1 and m=1 (slices 9-10), i.e. LSP5 has been shifted 2 slices to make room for the new connection.

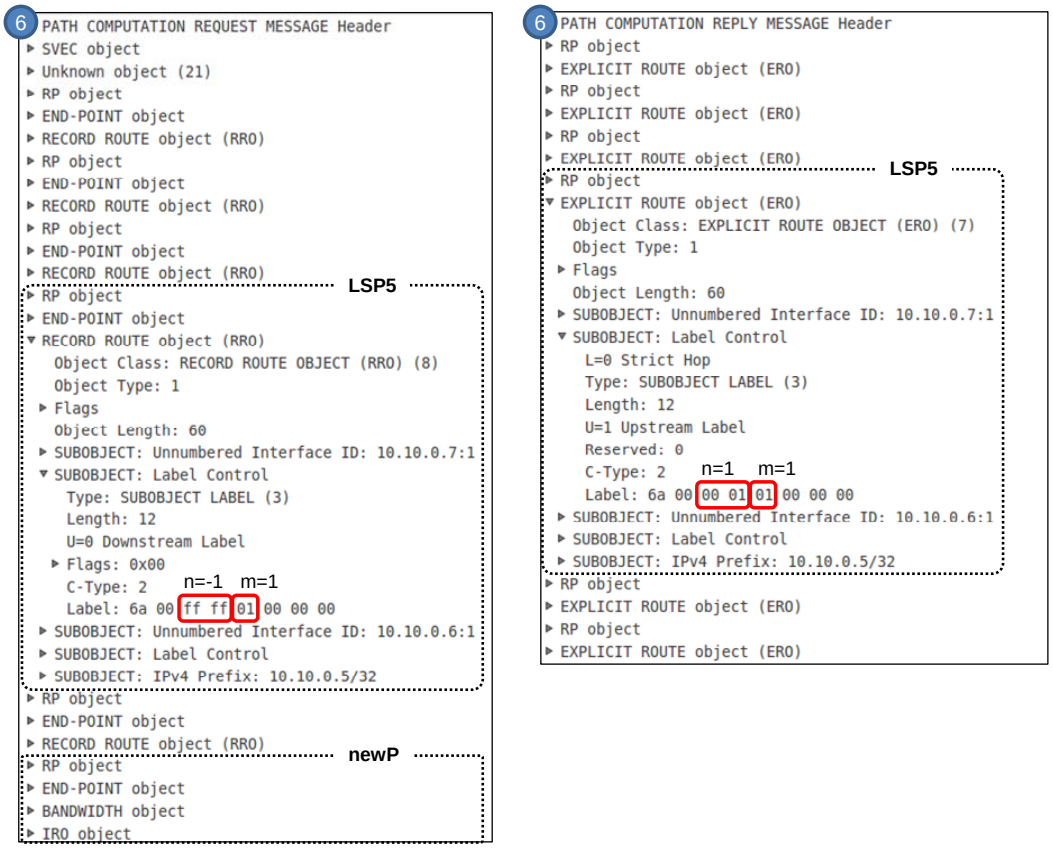


Fig. 5-8 PCEP message 6.

Once the fPCE receives the response with the solution, finds which are the LSPs that need to be updated and starts that updating process sending PCUpd messages to the source PCCs (step 7). Note, in view of messages in Fig. 5-5 that defragmentation is performed sequentially, following the order specified in RP objects. After all the LSP updating process ends, the new connection is set up using

a PCInit message (step 8). Finally, PCRep (message 9) is sent back to the ABNO controller, which in turns notifies connection set up to the NMS.

5.5 Conclusions

In this chapter, the spectrum shifting (SPRING) problem, an adaptation of the provisioning-triggered spectrum defragmentation for hitless re-optimization, has been studied as a first use case of in-operation network planning. The use case starts when a connection request cannot be served because of spectrum fragmentation in the links of the network. The hitless condition was ensured by constraining to shift connections between adjacent free frequency slots to make enough room to serve the new connection requested.

Firstly, the problem was formally stated and an existing MILP formulation for spectrum defragmentation was extended by adding constraints ensuring that connection reallocations are done in a hitless manner. Then, algorithms for the fPCE to select the candidate LSPs to be re-optimized and an exact algorithm to be run within the bPCE to solve the spectrum shifting problem for the set of candidate LSPs was proposed. A workflow defining the interactions between the ABNO components was devised and implemented in our iONE-based testbed to experimentally assess the spectrum shifting use case.

In the next chapter, we continue studying another in-operation planning application.

Chapter 6

Multiple Paths - After Failure Repair Optimization

Traffic affected by link failures can be recovered using path restoration schemes. To increase traffic restorability, multiple paths, named as *sub-connections* in flexgrid networks, can be used to restore every affected connection. However, the multipath restoration scheme might result in a poor resource utilization entailing a lesser GoS. In-operation network planning algorithms can be used to mitigate this problem once the failed link is repaired. In this chapter, we propose solving the so-called multipath after failure repair optimization problem (MP-AFRO) to reduce sub-connections count by aggregating those belonging to the same original connection and re-routing the resulting connection to release spectral resources. The MP-AFRO problem is modelled using a MILP formulation. In view of the complexity of the model and the limited time to solve the problem, we propose a heuristic algorithm that provides a good trade-off between complexity and optimality. The performance on the MP-AFRO heuristic algorithm is firstly validated by simulation and next, it is deployed inside our bPCE.

6.1 Restoration and After Failure Repair Optimization with Multiple Paths

For illustrative purposes, Fig. 6-1 reproduces a small flexgrid network topology where several connections are currently established; in particular, the route of connections *P1* and *P2* is shown. The spectrum usage is also shown in Fig. 6-1, where the spectrum allocation for all five established connections is specified. Three snapshots with the state of the network are shown. The link 6-7 has failed in

Fig. 6-1a; multipath restoration has been applied in Fig. 6-1b and connection *P2* has been split into two parallel sub-connections *P2a* and *P2b* squeezing the total conveyed bitrate to fit into the available spectrum. Failed link 6-7 has been repaired in Fig. 6-1c and re-optimization has been performed by solving MP-AFRO so that sub-connections have been merged back and bitrates have been expanded to its originally requested values.

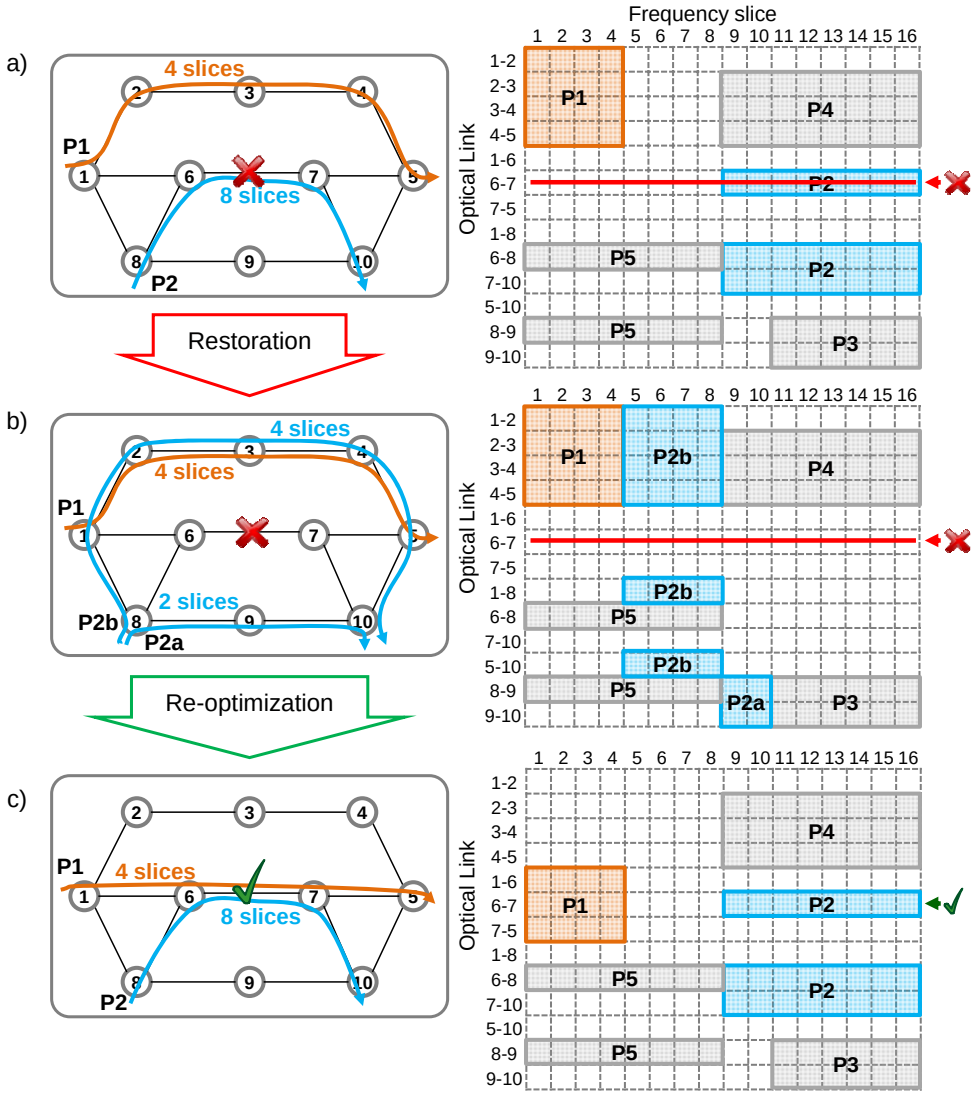


Fig. 6-1 Example of MP restoration and after failure repair optimization.

In view of the example, it is clear that multipath restoration allows increasing restorability, in particular when no enough contiguous spectrum can be found along a single path, as happened when restoring *P2*. Nonetheless, this benefit is at the cost of an increased resource usage, not only as a result of using (not shortest) parallel routes and squeezing the total conveyed connection's bitrate, but also

because the spectral efficiency is degraded when connections are split. For instance, a 400 Gb/s aggregated flow can be conveyed on one single 100 GHz connection or on four parallel 37.5 GHz sub-connections, therefore using 50% more spectral resources even in the case of being routed through the same links.

For this very reason, resource utilization can be improved by applying MP-AFRO, i.e. by re-routing established connections on shorter routes by merging parallel sub-connections to achieve better spectrum efficiency and expanding the conveyed connections' bitrates to its original ones. Fig. 6-1c illustrates an example of such re-optimization, where connection *P1* has been rerouted using a shorter route that includes the repaired link, whilst sub-connections *P2a* and *P2b* have been merged on a single connection conveying the originally requested bandwidth.

In the next section, the MP-AFRO problem is formally stated and a MILP formulation to solve it is presented.

6.2 The MP-AFRO Problem

6.2.1 Problem Statement

The MP-AFRO problem can be formally stated as follows:

Given:

- a network topology $G(N, E)$ defined as a set of optical nodes N and a set of fiber links E;
- an optical spectrum divided into frequency slices of a given width;
- the set of slices used by non-selected connections;
- a set of demands candidate for re-optimization, D.

Output: the route and spectrum allocation for demands in D, merging sub-connections serving each original connection.

Objective: maximize the bitrate served while minimizing the total number of sub-connections used to convey the traffic served.

An MILP-based model is presented in next section.

6.2.2 MILP Formulation

The MILP model is based in a *link-path* formulation for RSA [Ve14-2] where a set of routes is computed for each of the candidate demands to be re-optimized.

The following sets and parameters have been defined:

E Set of network links, index e.

S Set of slices in the spectrum of each link, index s.

D Set of candidate demands, index d. Each demand d represented by tuple $<s_d, t_d, b_d>$, where s_d is the source node, t_d is the target node, and b_d is the requested bitrate.

$P(d)$ Set of sub-connections being used to serve d.

$K(d)$ Set of pre-computed routes for demand d, index k.

$C(d)$ Set of slots for demand d, index c.

R Set of pairs reach-modulation format, index r.

δ_{ke} 1 if route k contains link e; 0 otherwise.

a_{es} 1 if slice s in link e is used by any already established connection not in D; 0 otherwise.

γ_{cs} 1 if slot c contains slice s; 0 otherwise.

β Objective function weight.

$b(c, r)$ Maximum bitrate that can be conveyed using slot c with the modulation format given by pair r.

$b(k, c)$ Maximum bitrate that can be conveyed through route k using slot c.

$len(r)$ Reachability of pair r (in Km).

$len(i)$ Length of route k (in Km).

Decision variables:

x_{dkc} Binary. 1 if route k and slot c are used to serve demand d; 0 otherwise.

y_{dkr} Binary. 1 if pair r is used for demand d in route k; 0 otherwise.

z_{dkc} Positive Real. Served bitrate for demand d through route k and slot c.

w_d Positive Integer. Total number of sub-connections used to serve demand d.

The $K(d)$ set is computed using eq. (6.1) as the union between the already-in-use paths $P(d)$ and the set of all shortest paths between s_d and t_d of the same length in hops as the shortest one, using the repaired link e.

$$K(d) = P(d) \cup \left\{ r \in KSP(s_d, t_d), e \in r \wedge |r| = |SP(s_d, t_d)| \right\} \quad \forall d \in D \tag{6.1}$$

The MILP formulation for the MP-AFRO problem is as follows:

$$\text{(MP-AFRO)} \qquad \max \quad \Phi = \sum_{d \in D} \left(\frac{\beta}{b_d} \sum_{k \in K(d)} \sum_{c \in C(d)} z_{dkc} - \frac{1}{|P(d)|} w_d \right) \tag{6.2}$$

subject to:

$$\sum_{k \in K(d)} \sum_{c \in C(d)} x_{dkc} = w_d \quad \forall d \in D \tag{6.3}$$

$$\sum_{d \in D} \sum_{k \in K(d)} \sum_{c \in C(d)} \delta_{ke} \cdot \gamma_{cs} \cdot x_{dkc} \le (1 - \alpha_{es}) \quad \forall e \in E, s \in S \tag{6.4}$$

$$w_d \le |P(d)| \quad \forall d \in D \tag{6.5}$$

$$\sum_{k \in K(d)} \sum_{c \in C(d)} z_{dkc} \le b_d \quad \forall d \in D \tag{6.6}$$

$$z_{dkc} \le b_d \cdot x_{dkc} \quad \forall d \in D, k \in K(d), c \in C(d) \tag{6.7}$$

$$z_{dkc} \le \sum_{r \in R} b(c,r) \cdot y_{dkr} \quad \forall d \in D, k \in K(d), c \in C(d) \tag{6.8}$$

$$\sum_{c \in C(d)} len(k) \cdot x_{dkc} \le \sum_{r \in R} len(r) \cdot y_{dkr} \quad \forall d \in D, k \in K(d) \tag{6.9}$$

$$\sum_{r \in R} y_{dkr} = 1 \quad \forall d \in D, k \in K(d) \tag{6.10}$$

Objective function in eq. (6.2) maximizes the total served bitrate while minimizing the amount of sub-connections used to serve that bitrate. Constraint (6.3) accounts for the number of sub-connections used to serve each demand. Constraint (6.4) ensures that any single slice is used to convey one sub-connection at maximum, provided that it is not already used by other demand not in *D*. Constraint (6.5) guarantees that sub-connections count to serve any specific demand is not increased. Constraint (6.6) assures that served bitrate does not exceeds demand's requested bitrate. Constraint (6.7) sets to zero bitrate of unused sub-connections. Constraint (6.8) limits the bitrate conveyed by any specific sub-connection to the maximum associated to pair *r*, whereas constraint (6.9) limits the length of each used sub-connection to the reachability associated to pair *r*. Finally, constraint (6.10) selects one pair *r* for each sub-connections.

The MP-AFRO problem is NP-hard since simpler network routing problems have been proved to be NP-hard (see e.g. [Ramas95-1]). Regarding its size, the number of variables is $O(|D| \cdot |K(d)| \cdot (|C(d)| + |R|))$ and the number of constraints is $O(|D| \cdot |K(d)| \cdot |C(d)| + |E| \cdot |S|)$.

Although the MILP model can be solved in a short period of time, e.g. minutes using a commercial solver such as CPLEX or dozens of seconds if a column generation algorithm is used [Ru14-3], during MP-AFRO computation new connections could arrive, which need to be queued and delayed until the MP-AFRO solution is implemented in the network. Aiming at reducing provisioning delay, while providing a good trade-off between complexity and optimality, a heuristic algorithm is presented next to solve MP-AFRO in the stringent required times (e.g. < 1 s.).

6.2.3 Heuristic Algorithm

The MP-AFRO algorithm is shown in Table 6-1. The algorithm maximizes the amount of bitrate that is served (note that only part of the original bitrate could be restored), while minimizing the number of sub-connections per demand being re-optimized. The algorithm receives as input the network's TED, i.e. $G(N, E)$, the candidate list of demands to be re-optimized D, and the maximum number of iterations *maxIter* to be run; it returns the ordered set *bestS* containing the best solution found, where the list of sub-connections to serve each demand is specified.

All the sub-connections of each demand are first de-allocated from of the original TED in G (line 2 in Table 6-1) and then, the algorithm performs a number of iterations (*maxIter*), where the set D is served in a random order (lines 3-5). At each iteration, a copy of the original TED is stored into an auxiliary TED G', so that every subsequent operation is performed over G'. Each iteration consists of two steps performed sequentially. In the first step (lines 6-12), the set of demands is served ensuring the currently served bitrate in the hope of finding a shorter route. The *getMP_RSA* function computes the set of sub-connections consisting of the routes with the highest available capacity, using eq. (6.11), and minimum cost and selects those to serve the required bitrate (line 8). A solution is feasible only if every demand obtain at least the same bitrate than the current allocation so in case the required bitrate cannot be served, the solution is discarded (lines 9-10), otherwise the resources selected are reserved in the auxiliary TED G' (line 12). In the second step, the bitrate of the demands is increased firstly by allocating wider slots along the same routes (lines 17-21) and then by adding more sub-connections (lines 22-27).

$$c_k(d) = \left\{ c^* \;\middle|\; \begin{array}{l} b(k,c^*) \geq b(k,c)\ \forall c \in C(d) \\ \alpha_{es} = 0\ \forall e \in k, s \in S(c) \end{array} \right\} \tag{6.11}$$

After a complete solution is obtained, its fitness value is computed (line 28). Next, the used resources in the solution are released from the auxiliary TED (line 29). The fitness value of the just obtained solution is compared to that of the best solution found so far (line 30) and stored provided that it is feasible and its fitness

value is better than that of the incumbent. The state of G is restored and the best solution is eventually returned (lines 31-32).

Table 6-1 MP-AFRO Heuristic Algorithm

```
INPUT G(N, E), D, maxIter
OUTPUT bestS
 1:  bestS ← D
 2:  for each d ∈ D do deallocate(G, d.SC)
 3:  for i = 1..maxIter do
 4:    S ← Ø; G' ← G; feasible ← true
 5:    sort(D, random)
 6:    for each d ∈ D do
 7:      d.SC' ← Ø; d.reoptBw ← 0
 8:      {<k, c_k>} ← getMP_RSA(d, d.servedBw)
 9:      if b({<k, c_k>}) < d.servedBw then
10:        feasible ← false; break
11:      d.reoptBw ← b({<k, c_k>}); d.SC' ← {<k, c_k>}
12:      allocate(G', d.SC'); S ← S ∪ {d}
13:    if not feasible then
14:      for each d ∈ S do deallocate(G', d.SC')
15:      continue
16:    for each d ∈ D do
17:      if d.reoptBw < d.bw then
18:        {<k, c_k>} ← expandSlots(d, d.bw, G')
19:        if b({<k, c_k>}) > d.reoptBw then
20:          d.reoptBw ← b({<k, c_k>})
21:          d.SC' ← {<k, c_k>}; continue
22:        if d.reoptBw<d.bw and |d.SC'| < |d.SC| then
23:          {<k,c_k>} ← getMP_RSA(d, d.bw - d.reoptBw)
24:          if b({<k, c_k>}) = 0 then break
25:          d.reoptBw ← d.reoptBw + b({<k, c_k>})
26:          d.SC' ← d.SC' ∪ {<k, c_k>}
27:          allocate(G', d.SC')
28:    Compute Φ(S)
29:    for each d ∈ D do deallocate(G', d.SC')
30:    if Φ(S) > Φ(bestS) then bestS ← S
31:  for each d ∈ D do allocate(G, d.SC)
32:  return bestS
```

6.3 Proposed Re-optimization Workflow

In this section, we propose a workflow to implement MP-AFRO. We assume that an operator in the NMS triggers the MP-AFRO workflow after a link has been repaired. To that end, the NMS issues a service request towards the ABNO Controller. Fig. 6-2 reproduces the re-optimization workflow involving ABNO components and its execution flow diagram (Fig. 6-3). When the request from the NMS arrives at the ABNO controller, re-optimization is requested by sending a PCReq message (labeled as 1 in Fig. 6-2) to the fPCE. Upon receiving the request,

the fPCE collects relevant data to be sent to the bPCE in the form of a PCReq message containing a GCO request (2).

In light of the MP-AFRO problem statement and assuming that the network topology and the current state of the resources has been synchronized, the information to be included in the GCO request is the set of connections candidate for re-optimization, *D*. Therefore, an algorithm to find the candidate connections in the LSP-DB is needed. The algorithm, presented in Table 6-2, receives as input the network's TED in *G*, the set *L* with the LSP-DB, and the repaired link *e*. The connections whose shortest path traverses the repaired link are selected and the candidate list of demands *D* to be re-optimized is eventually returned.

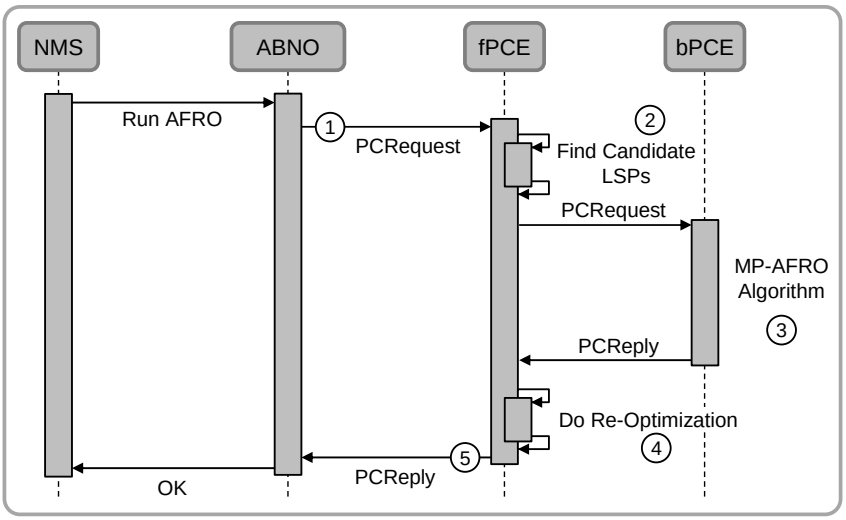


Fig. 6-2 Proposed workflow.

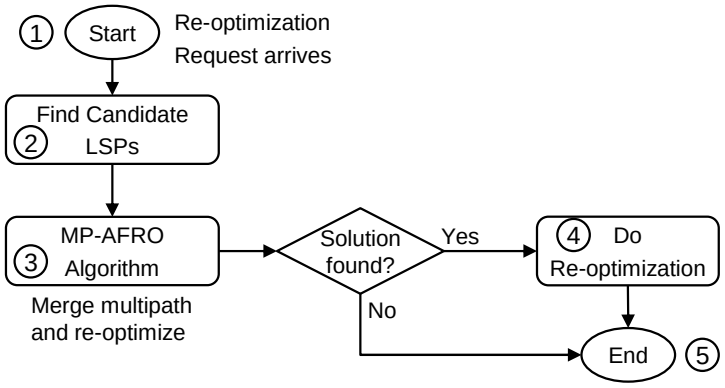


Fig. 6-3 Re-optimization flow diagram.

Information regarding the candidate connections is sent in a PCReq message containing the GCO request. Each candidate connection is sent as an individual request, identified by a `RP` object; the end points and original bandwidth are also included using the `END-POINTS` and the `BANDWIDTH` objects, respectively. In

addition, for each individual sub-connection related to the original connection, its current route and spectrum allocation are specified using a RRO object. The RRO object interleaves the route's nodes with its spectrum allocation. The sequence of nodes is encoded using the Unnumbered Interface sub-objects encoding each node and the corresponding outgoing port identifiers to reach the next node; and an IPv4 Prefix sub-object to identify the destination node.

Table 6-2 Find Candidate Demands Algorithm

```
INPUT G(N, E), L, e
OUTPUT D
1:  D ← Ø
2:  for each d ∈ L do
3:    {SP} ← KSP(d.s, d.t, G)
4:    for each SP ∈ {SP} do
5:      if e ∈ SP then D ← D ∪ {d}
6:  return D
```

The spectrum allocation is encoded using the Label Control sub-object that contains the tuple <n, m> that unambiguously defines any frequency slot, where n is the central frequency index (positive, negative or 0) of the selected frequency slot from a reference frequency (193.1 THz), whereas m is the slot width in number of slices at each side of the central frequency. Aiming at finding an optimal solution for the entire problem, individual requests are grouped together using a SVEC object.

The desired network-wide GCO related criterion, such as "MP-AFRO", is specified by means of the OF object. Finally, the repaired link must be used for re-optimization. To that end, we extended current standards adding an IRO object that identifies those repaired link that should be included into the new routes in a symmetric way with respect to the XRO (Exclude Route Object) object that specifies the links that should be excluded from route being computed.

Upon receiving the PCReq, the bPCE runs the specified heuristic algorithm. The solution of the MP-AFRO problem is encoded in a PCRep (labeled as 3 in Fig. 6-2); each individual request is replied specifying the bitrate that could be served and the route and spectrum allocation of the connections related to each request in a list of ERO objects. Note that the solution might entail merging several existing sub-connections to create one or more new connections. The Order TLV needs to be included in RP objects to indicate the order in which the solution needs to be implemented in the data plane.

Upon receiving the PCRep message with the solution from the bPCE, the fPCE runs the *Do-Re-Optimization* algorithm (labeled as 4 in Fig. 6-2), listed in Table 6-3, to update the connections in the network's data plane. The algorithm sorts D in the same order than solution S (line 1 in Table 6-3). Next, each demand d in D is sequentially processed. The possibly-updated demand ds in S corresponding to the

current demand *d* in *D* is taken (line 3) by matching them using the identifier stored in its related RP object. If *d* differs from *ds*, the bPCE has found a better set of sub-connections for the demand, so the fPCE should send the corresponding PCUpd/PCInit messages towards source GMPLS controllers (line 4) to update/implement it [draft-stateful], [draft-initiate].

Table 6-3 Do-Re-Optimization Algorithm

```
INPUT D, S
1:  sort(D, S.order)
2:  for each d ∈ D do
3:    ds ← getDemand(S, d)
4:    if d ≠ ds then update(d, ds)
```

When the solution has been completely implemented, i.e. when the corresponding PCRpt messages are received, the fPCE replies the completion of the requested operation to the ABNO controller using a PCRep message (labeled as 5 in Fig. 6-2), which eventually informs the NMS.

6.4 MP-AFRO Validation

6.4.1 MP-AFRO Performance Evaluation

To evaluate the performance of MP-AFRO, we used two representative core network topologies: the 30-node Spanish Telefonica (TEL) and the 28-node European (EON) networks. We consider the fiber links with the spectrum width equal to 2 THz, divided into frequency slices of 12.5 GHz and granularity 6.25 GHz.

The MP-AFRO heuristic was developed in C++ and integrated in an ad-hoc event driven simulator based on OMNET++. Connection requests are generated following a Poisson process and are torn down after an exponentially distributed holding time with mean equal to 6 months. The source and destination nodes are randomly chosen using the uniform distribution.

As in [Pao14-1], the bitrate of each connection request was randomly selected considering 80% of the connections being 100 Gb/s and the other 20% of 400 Gb/s. To convert bitrate into spectrum width, we use the correspondence in Table 6-4. Note that every point in the results is the average of 10 runs with 100,000 connection requests each. Finally, we assume a link failure rate of $2.72\ 10^{-3}$ failures per km per year [Gr04-1] and consider that the link is repaired immediately after the restoration process ends.

Table 6-4 Bitrate-Spectrum Width

Bitrate (Gb/s)	Bandwidth (GHz)	#Slices
100	37.5	6
200	62.5	10

400	100	16

Aiming at comparing the performance when the MP-AFRO is used, Fig. 6-4 plots the blocking probability (Pb) against the offered load with and without applying MP-AFRO. Pb is weighted using connections' bitrate and the load is normalized to the largest load unleashing Pb < 4%. Although the actual load gain depends, among other factors, on the characteristics of network topology considered, gains above 13% can be obtained using MP-AFRO.

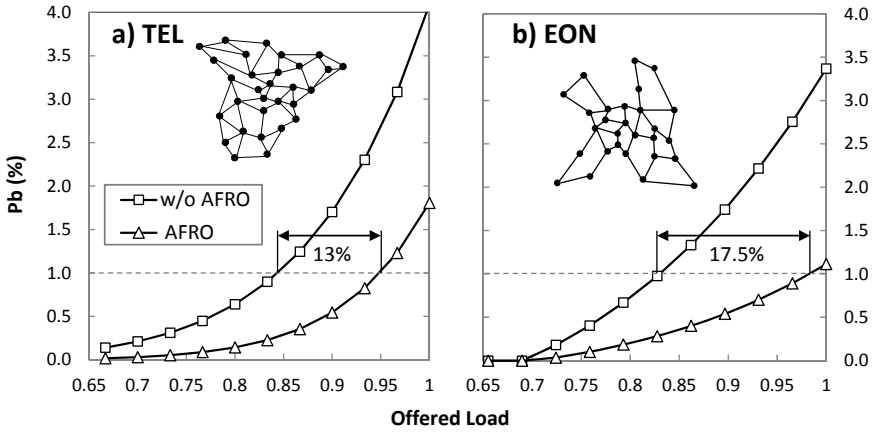


Fig. 6-4 Blocking probability against offered load.

Once the performance of MP-AFRO has been evaluated, the proposed workflow is experimentally validated next.

6.4.2 Experimental Assessment

For the experiments, we consider the network topology illustrated in Fig. 6-1b. The data plane includes programmable SSS and node emulators are additionally deployed to complete the topology. Nodes are handled by co-located GMPLS controllers running RSVP-TE with flexgrid extensions. The controllers run a proprietary configuration tool for automatic filter re-shaping with a resolution of 1 GHz. Controllers communicate with the fPCE by means of PCEP through Gigabit Ethernet interfaces.

Fig. 6-5 depicts the distributed field trial set-up connecting premises in Telefonica (Madrid, Spain), CNIT (Pisa, Italy) and UPC through IPSec tunnels where experiments have been carried out.

A link failure has already triggered a multipath restoration to reroute the failed connections splitting them into multiple sub-connections per demand. After the failed link is repaired, the NMS operator decided to trigger the MP-AFRO algorithm to improve network efficiency, so the NMS requests the computation to the ABNO controller. The ABNO controller in turn sends the PCReq message (message 1 in Fig. 6-6) to the fPCE containing the repaired link identification by

means of an IRO object, and an OF object to specify the algorithm to be executed, in this experiment the MP-AFRO.

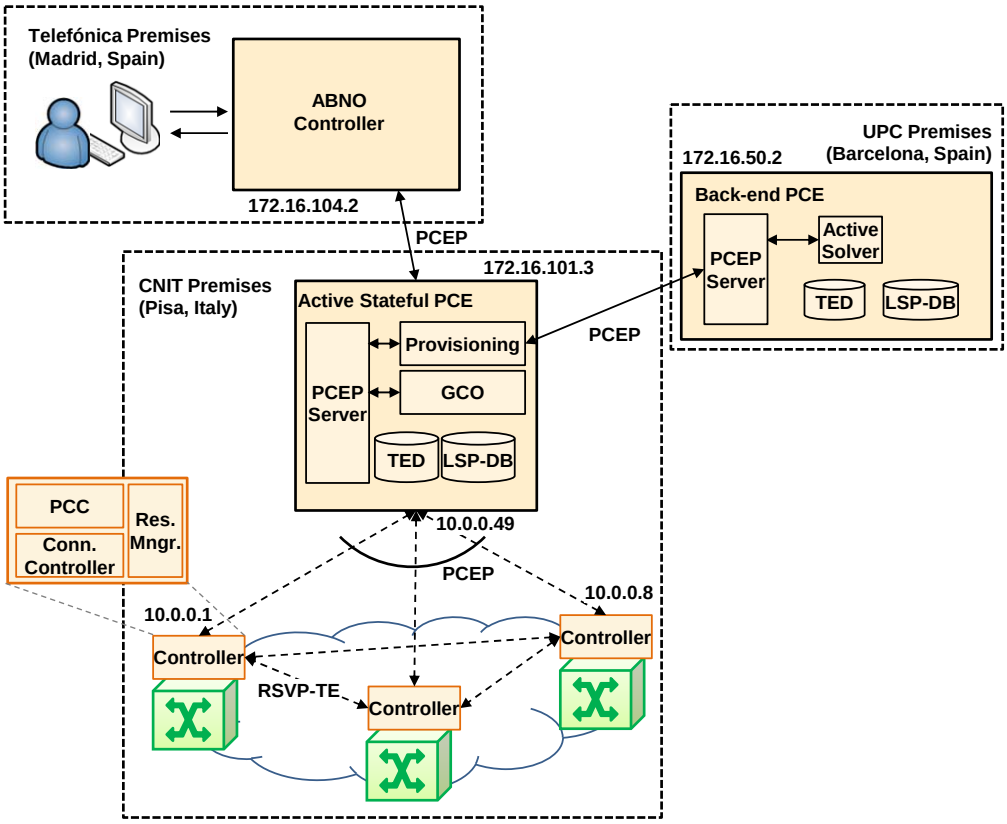


Fig. 6-5 Distributed field trial set-up.

No.	Time	Source	Destination	Info
1 17	8.097681	172.16.104.2	172.16.101.3	PATH COMPUTATION REQUEST
2 18	8.097845	172.16.101.3	172.16.50.2	PATH COMPUTATION REQUEST
3 19	8.199351	172.16.50.2	172.16.101.3	PATH COMPUTATION REPLY
4 20	8.220174	10.0.0.49	10.0.0.1	PATH COMPUTATION UPDATE
21	8.223918	10.0.0.1	10.0.0.49	PATH COMPUTATION REPORT
5 22	11.281405	10.0.0.49	10.0.0.8	PATH COMPUTATION UPDATE
23	11.285044	10.0.0.8	10.0.0.49	PATH COMPUTATION REPORT
6 24	16.639926	172.16.101.3	172.16.104.2	PATH COMPUTATION REPLY

Fig. 6-6 PCEP messages exchanged.

When the fPCE receives the ABNO request, it runs the *Find Candidate Demands* algorithm using as repaired link the one encoded in the received IRO object, to obtain the candidate list of demands to be re-optimized, and a PCReq to be sent to the bPCE is composed.

This PCReq message encodes the candidate demands using RP objects to identify them, an END-POINTS object defining its source and target nodes, a BANDWIDTH object to specify the requested bandwidth, and one RRO object for each sub-connection being used by the demand. Included candidate demands are grouped by means of a SVEC object so that they are re-optimized jointly. The received OF and IRO objects are also included in the PCReq message to inform to the bPCE on which algorithm should be executed and which link has been repaired respectively (message 2).

Fig. 6-7 details the PCReq message (2) received by the bPCE highlighting the RRO objects encoding the original sub-connections. The first RRO object and one of the Label Control sub-objects has been expanded to identify its original sub-connection's route and frequency slot. Objects in Fig. 6-7 correspond to demand *P2a* and *P2b* in Fig. 6-1b, composed by two sub-connections. *P2a* traverses nodes 8-9-10 using frequency slot <*n*=1, *m*=1>, and *P2b* traverses nodes 8-1-2-3-4-5-10 using frequency slot <*n*=-2, *m*=2>.

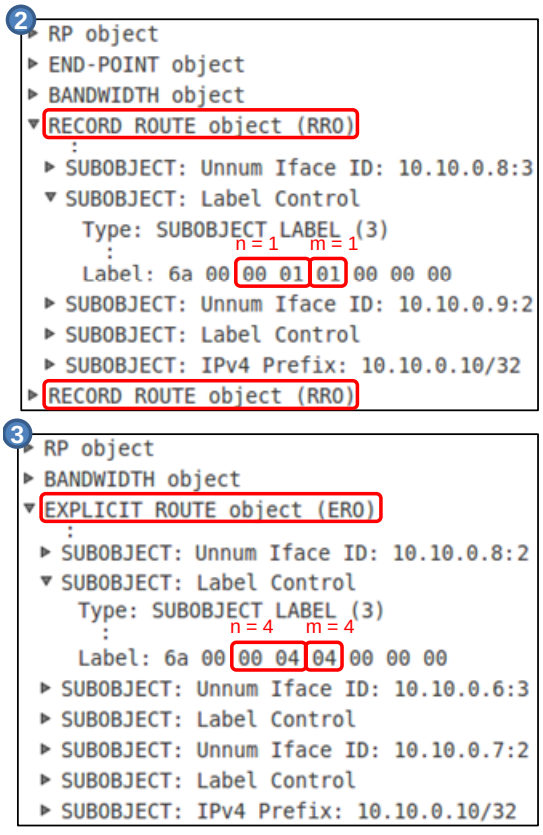


Fig. 6-7 Detail of PCEP messages (2) and (3).

After receiving the PCReq message, the bPCE computes the MP-AFRO algorithm and responds using a PCRep message (message 3) containing the solution found.

For each demand, the received RP object is duplicated and included into the PCRep message, adding also a BANDWIDTH object specifying the served bandwidth, and one ERO object for each sub-connection to be established to serve the demand. The ERO objects have the same format than the RRO objects.

Fig. 6-7 details the PCRep message (3) replied by the bPCE highlighting the ERO objects encoding the re-optimized sub-connections. In contrast to message (2), objects in message (3) correspond to demand *P2*, the result of merging *P2a* and *P2b*, in Fig. 6-1b. The new computed connection traverses nodes 8-6-7-10 using frequency slot <*n*=4, *m*=4>. The resulting connection has restored the initially failed connection using the repaired link.

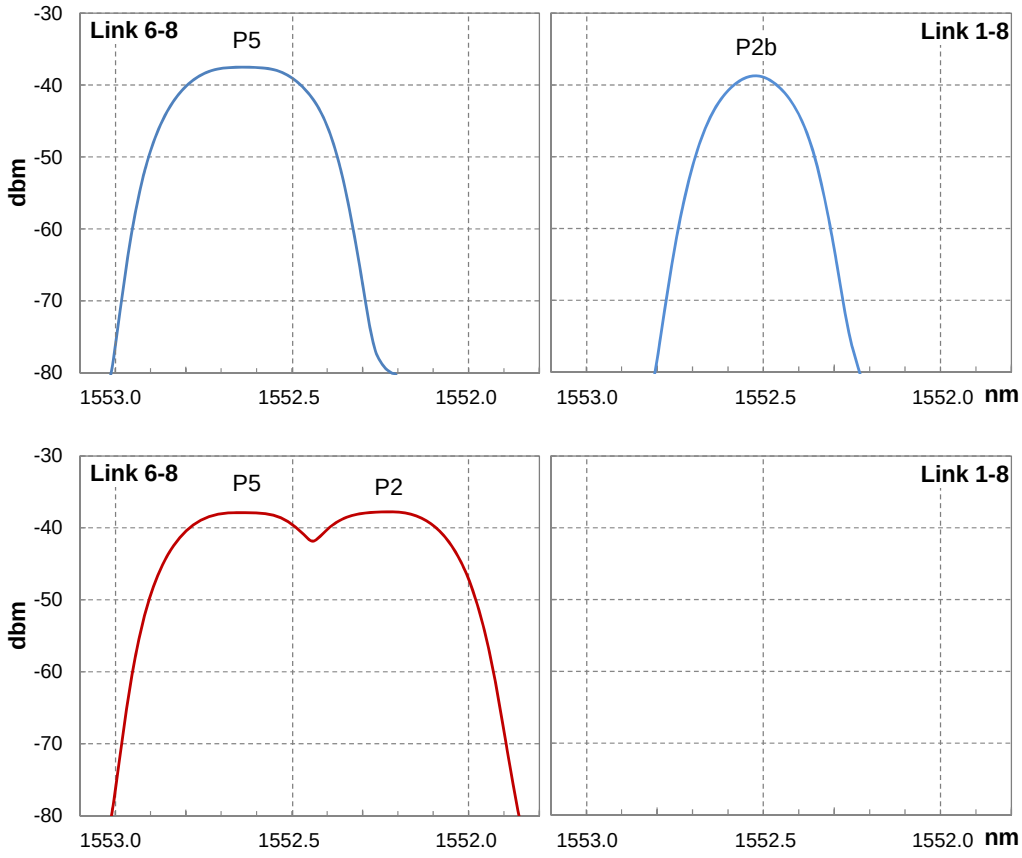


Fig. 6-8 Optical spectrum before (top) and after MP-AFRO (bottom).

Upon PCRep message arrival, the fPCE starts updating connections by sending PCUpd messages towards the involved GMPLS controllers (IPs: 10.0.0.X). Data plane nodes are located in network 10.10.0.X. The controllers operate on a flexgrid test-bed derived from [Cu12-1], including 100 Gb/s polarization multiplexed quadrature phase shift keying (PM-QPSK) optical signals and bandwidth variable cross-connects. Fig. 6-8 shows the optical spectrum on links 6-8 and 1-8 before and after the re-optimization is performed. PCRpt responses are generated when the

requested actions have been performed (messages 4-5 in Fig. 6-6). When the solution computed by the bPCE has been completely deployed, the fPCE confirms the ABNO controller the end of the requested re-optimization by sending a PCRep message (6).

The overall re-optimization was successfully completed in around eight seconds, including circa 150 ms. of pure control plane contribution (i.e. messages exchange and path computation algorithms at fPCE and bPCE). Note that around 1 s is required to tune each SSS.

6.5 Conclusions

In this chapter, we faced another use case of in-operation network planning. When a link fails, multipath restoration can be used to increase restorability of affected connections at the cost of worse resource utilization and spectral efficiency. After the link is repaired, the MP-AFRO problem can be solved to aggregate multiple sub-connections serving a single demand using shorter routes, thus releasing spectrum resources that now can be used to convey new connection requests. The MP-AFRO was modelled as a MILP formulation and a heuristic algorithm was devised to find good feasible solutions in practical computation times.

The performance of MP-AFRO was evaluated on an ad-hoc network simulator and experimentally demonstrated on a distributed test-bed. The relevant PCEP messages were shown. Note that since network dynamicity frequently derives into not optimal use of resources, this use case can easily be extended to be triggered by any other event.

Chapter 7

Point-to-MultiPoint Connectivity

In this chapter, we present another application of in-operation planning that is motivated by the increasing demand of internet services. This increment is pushing service providers to increase the capacity of their DCs as well as the DC interconnection networks. As a result of the huge capacity required for the inter-DC network, the flexgrid optical technology can be used. In this scenario, applications can run in DCs placed in geographically distant locations and hence, multicast-based communication services among their components might be required.

Typically, one or more core Ethernet programmable switches within each DC are connected to the flexgrid optical core network providing the necessary interconnection among the DCs. We study two different approaches to provide multicast services in multi-layer scenarios assuming that the optical network is based on the flexgrid technology: *i*) establishing a point-to-multipoint optical connection (light-tree) for each multicast request, and *ii*) using a multi-purpose VNT to serve both unicast and multicast connectivity requests. When the VNT is not able to serve an incoming request as a result of lack of capacity, it can be reconfigured to add more capacity.

To deal with both approaches, heuristic algorithms are proposed and next, the proposed algorithms are experimentally assessed.

7.1 Approaches to Serve Multicast Connectivity Services

To implement multicast connectivity services in a multi-layer network, a VNT is needed to connect every switch. As a result of the large bitrate required for the

multicast connectivity, the VNT can be created ad-hoc for each multicast request and removed when the multicast connection is torn-down.

We assume that the ad-hoc VNT is based on one single light-tree connecting the source switch to each of the destination switches. An example of the light-tree-based approach is depicted in Fig. 7-1, where a 100 Gb/s light-tree is set-up in Fig. 7-1a connecting the switch in DC-A to the switches in DCs B, C, and D. It is clear that specific p2mp RSA algorithms are needed to compute the route from the source to every leaf in the connection and allocate a contiguous and continuous spectrum slot. The resulting VNT consists in a single p2mp virtual link connecting the source switch to all the leaves (Fig. 7-1b). The multicast connection request can be served on the just created ad-hoc p2mp VNT (Fig. 7-1c). The advantage of this approach include the reduction of switching capacity in the switches, since multicast is performed by the optical layer.

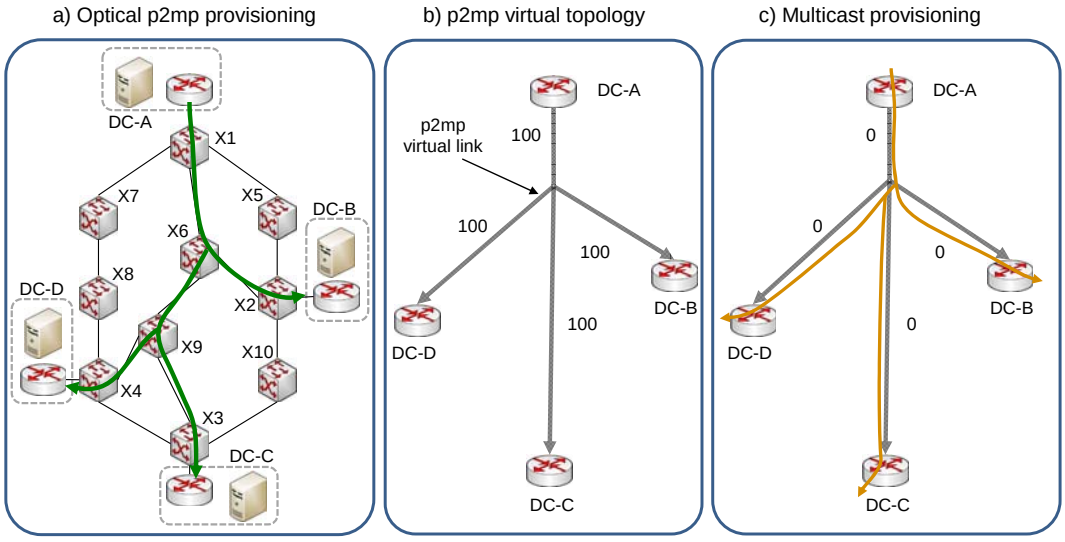


Fig. 7-1 Light-tree-based VNT to serve an incoming multicast request.

Although the p2mp virtual link created for the multicast connection request could be shared, the probability of another multicast request with the same source and set of destination switches as that virtual link arrive is clearly scarce. Thus, in case of some capacity would remain unused after serving the multicast request, as a result of a mismatch between the requested bitrate and the optical layer granularity, it would be unlikely reused.

To foster capacity sharing, a single VNT can be created to serve any kind of connection request, including unicast and multicast connections. In this approach, connection requests are served on the VNT, which is updated to add extra capacity, e.g. new virtual links, in case an incoming connection cannot be served, and releasing virtual links when any connection is using them. Fig. 7-2 shows an example of the multi-purpose VNT approach. In Fig. 7-2a the current multi-

purpose VNT has not enough capacity to serve an incoming 100 Gb/s multicast request from the switch in DC-A to the switches in DCs B, C, and D. Therefore, the VNT is updated adding a new virtual link. Taking advantage of traffic grooming to improve resource utilization, the VNT can be supported on large capacity, e.g. 400 Gb/s, lightpaths; then a new virtual link connecting switch DC-C to switch DC-D with 400Gb/s of remaining capacity is now available in the VNT, as shown in Fig. 7-2b. Finally, the requested multicast connection can be served on the multi-purpose VNT, as illustrated in Fig. 7-2c.

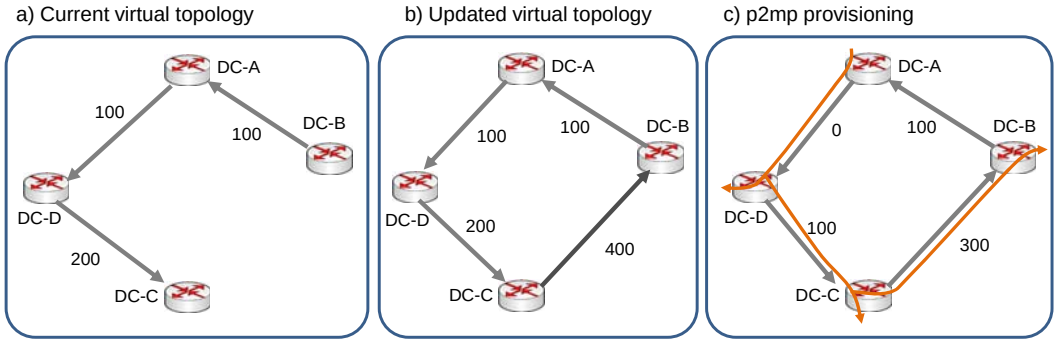


Fig. 7-2 Multi-purpose VNT updating to serve an incoming multicast request.

In the next section, we provide algorithms to route multicast connections under the single layer approach, serving traffic directly on the optical layer, and under the multi-layer approach, serving traffic on a VNT created on top of the optical network.

7.2 Heuristic Algorithms

7.2.1 Multicast Provisioning Single Layer tree (McP-SL-tree)

Table 7-1 shows the main algorithm for solving the McP-SL-tree problem, which consists in a routing phase that builds a tree followed by a spectrum allocation phase. Although routing and spectrum allocation are independent, the routing phase tries to guarantee the availability of at least one end-to-end slot for the request.

The algorithm receives as input:

- the tuple with the parameters for the multicast connection request including its source node o, the set of target nodes T and the desired bitrate b,
- the set S with the spectrum slices,
- the set R with the pre-computed routes, in particular the set $R(o, t)$ contains the routes between nodes o and t, and

- the cost function weights *ch*, *ct* and *cs*, where *ch* is the cost per additional hop, *ct* is the cost per additional intermediate destination node traversed and *cs* the cost per additional available slot.

Before starting the routing phase, the set of all possible slots with enough bandwidth to allow serving the requested bitrate is pre-computed (line 2 in Table 7-1). Next, an iterative procedure that finds and merges routes from source to destinations into a tree under construction is executed (lines 3-12). The procedure runs until all the destination nodes are added to the tree (line 3). At each iteration, all remaining destinations are evaluated as candidates to be inserted into the tree. Once some parameters are initialized (line 4), the route with the minimum cost for each remaining destination is found (lines 5-6). Among all destinations, the one with the highest minimum cost is selected (lines 7-8). The solution is infeasible if no route is available for any destination (line 9). On the contrary, the tree is updated adding the new links in the selected route (line 10). The set of slots is updated eliminating those slots that cannot be selected after adding the new route (line 11). Finally, since a route can cover more than one remaining destination, the set T must be also updated at the end of each iteration (line 12). Once the tree has been built, an available slot is chosen from the set of end-to-end available slots along every tree edge (line 13), and the light-tree is eventually returned.

Table 7-1 McP-SL-tree Algorithm

```
INPUT <o,T,b>, S, R, <ch,ct,cs>
OUTPUT solution
 1:  solution ← Ø
 2:  slots ← computeSlots(S,b)
 3:  while T ≠ Ø do
 4:    maxCost ← -∞; r_sel ← Ø
 5:    for each t ∈ T do
 6:      <route,cost> ← R-SL(R^O(o, t), S, slots, T, solution, <ch,ct,cs>)
 7:      if cost > maxCost then
 8:        maxCost ← cost; r_sel ← route
 9:    if r_sel = Ø return infeasible
10:    solution ← updateTree(solution, r_sel)
11:    slots ← updateSlots(slots, solution)
12:    T ← updateT(T, solution)
13:  solution ← SA(slots)
14:  return solution
```

The success of the algorithm in Table 7-1 strongly depends on the algorithm R-SL that finds the candidate routes to be added to the tree. Table 7-2 shows the details of such algorithm, which receives, in addition to other data and parameters previously described, a set R^O of pre-computed routes from the source to the destination being evaluated over the optical layer. After initializing the output variables (line 1), the cost of each route is evaluated aiming at finding that with min cost, taking into account the already built tree and slot availability.

Before computing the cost of a candidate route, the number of slots in common with the current tree is computed (lines 3-4). If some slot in common is found, the number of hops in the route not used in the current tree is computed (line 5). Next, the number of intermediate destinations not included in the current tree that are intermediate nodes in the route to the end destination is obtained (line 6). These variables are used to compute the cost of the route (line 7). If cost coefficients are non-negative, the cost of the route is minimized when: *i*) the route adds few new links to the existing tree; *ii*) the number of intermediate destinations increases; and *iii*) the number of available slots increases. It is worth mentioning that the weight of each cost coefficient will affect the selected route and, consequently, the quality of the obtained light-tree.

Table 7-2 R-SL Algorithm

```
INPUT R^O, S, slots, T, solution, <ch,ct,cs>
OUTPUT route, cost
 1:  route ← ∅; cost ← ∞
 2:  for each r ∈ R^O do
 3:    nslots ← compAvailableSlots(r, slots, S, solution)
 4:    if nslots = 0 then continue
 5:    nhops ← compHopsNotInTree(r, solution)
 6:    ndests ← compIntermDestinations(r, T)
 7:    costIte ← ch * nhops – ct * ndests – cs * nslots
 8:    if cost < costIte then
 9:      route ← r
10:      cost ← costIte
11:  return {route, cost}
```

7.2.2 Multicast Provisioning Multi-layer tree (McP-ML-tree)

Table 7-3 presents the McP-ML-tree main procedure. The algorithm receives the following input parameters that differ from the McP-SL-tree algorithm: the graph $G^O(N, L)$ representing the optical topology, where N is the set of optical nodes and L the set of optical links, the graph $G^V(N^R, E)$ for the current virtual topology and the cost weight cv per additional vlink added. In contrast to the set of pre-computed routes R^O used in the McP-ML-tree algorithm, the set R^V stores pre-computed routes between o and T over the virtual topology. To compute R^V, two kind of virtual links are considered: *i*) current virtual links with enough residual capacity to serve the request (E^{on}), and *ii*) new virtual links that can be supported by feasible lightpaths, taking into account spectrum and transponders availability (E^{off}). All routes that do not fulfill those conditions are pruned (lines 1-3).

After pre-processing routes, the McP-ML-tree algorithm is called to find a tree on the virtual topology and the set of virtual links in E^{off*} to be set up for the multicast request (line 4). A lightpath with the desired bitrate needs to be found for each of the links in E^{off*} (line 7). To minimize resource utilization when routing several lightpaths, a bulkRSA algorithm can be used (e.g. see [Ve14-2]). Note that in the

pre-processing phase we ensured that each single link in E^{off} can be established. However, when E^{off*} contains several links, not all could be set-up simultaneously with the available capacity, so the request could be rejected (line 8). If every virtual link can be created, G^V is updated and the solution finally returned (lines 9-10).

Table 7-3 McP-ML-tree Main Procedure

```
INPUT G^O(N,L), G^V(N^R,E), <o,T,b>, S, R^V, <ch,ct,cv>
OUTPUT solution
 1:  E^on ← getAvailableVlinks(G^V, b)
 2:  E^off ← getNewVlinks(G^O, G^V, S)
 3:  R^V ← pruneR(R^V, E^on, E^off)
 4:  <solution, E^off*> ← McP-ML-tree(<o,T,b>, R^V, E^on, E^off, <ch,ct,cv>)
 5:  if solution = ∅ then return infeasible
 6:  if E^off* ≠ ∅ then
 7:     status ← bulkRSA(G^O, S, E^off*)
 8:     if status = infeasible then return infeasible
 9:     G^V ← addVLinks(G^V, E^off*)
10:  return solution
```

The McP-ML-tree algorithm is similar to that of the McP-SL-tree in Table 7-1. Just note that pre-computing and updating available slots is not done in the McP-ML-tree algorithm. Much the same, the R-ML algorithm to compute the cost of adding a route to the tree for the multi-layer approach in Table 7-4 is similar to the R-SL algorithm in Table 7-2. In this case, the number of new virtual links to be established needs to be obtained (line 3). As for the single layer case, the cost is computed as the sum of the three components weighted by their cost coefficients. Note that the higher is the amount of new links to install, the higher is the cost of the route.

Table 7-4 R-ML Algorithm

```
INPUT R^V, E^off, T, solution, <ch,ct,cv>
OUTPUT route, cost
 1:  route ← ∅; cost ← ∞
 2:  for each r ∈ R^V do
 3:    nvlinks ← compNewVlinks(r, E^off)
 4:    nhops ← compHopsNotInTree(r, solution)
 5:    ndests ← compIntermDestinations(r, T)
 6:    costIte ← cv * nvlinks + ch * nhops – ct * ndests
 7:    if cost < costIte then
 8:      route ← r
 9:      cost ← costIte
10:  return {route, cost}
```

Next section proposes a workflow for each of the approaches and studies the support of PCEP.

7.3 Proposed Workflows and Protocol Issues

We assume that an ASO module is responsible for managing multiple DC networks as a single entity. Thus, it generates the connection requests and forwards them to the ABNO controller for unicast and multicast services between Ethernet switches. Fig. 7-3 for the light-tree-based VNT approach and Fig. 7-4 for the multi-purpose VNT approach present the proposed workflow to serve multicast connection requests.

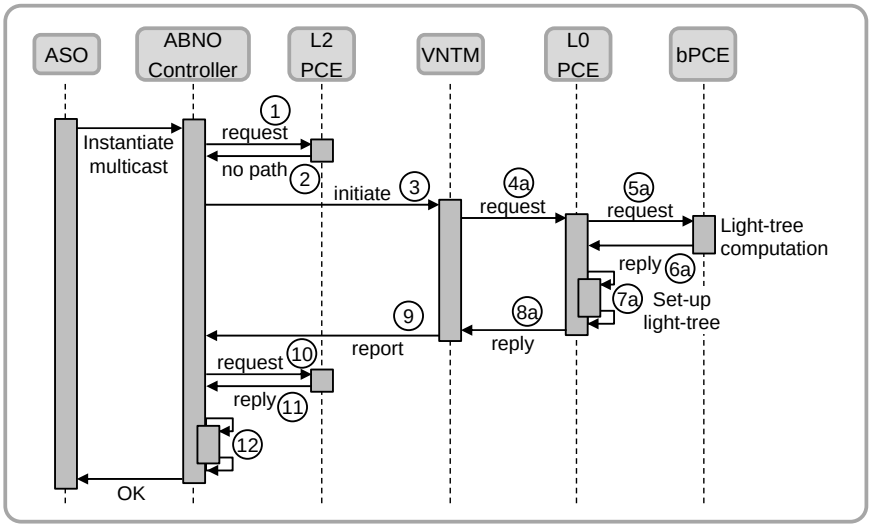


Fig. 7-3 Workflow for the light-tree-based VNT approach.

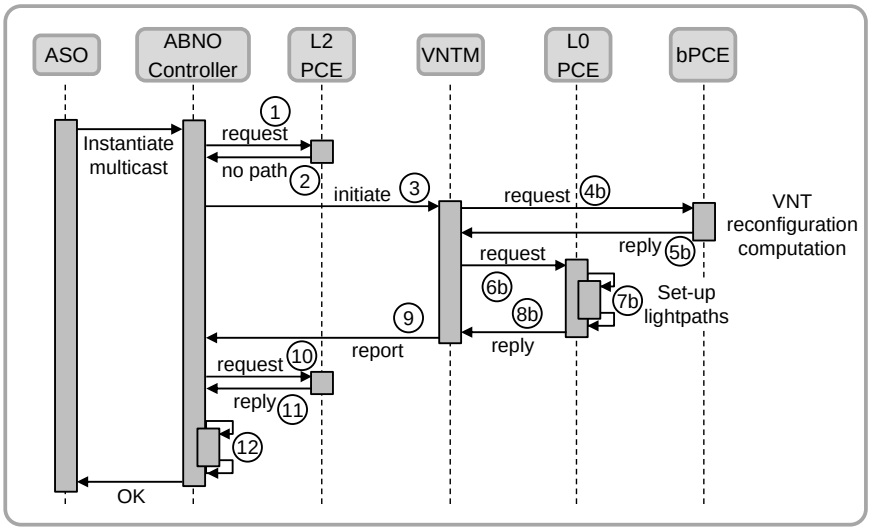


Fig. 7-4 Workflow for the multi-purpose VNT approach.

Upon a multicast connection request arrives in the ABNO controller, it requests a L2 p2mp path computation to the L2 PCE (PCReq message 1 in Fig. 7-3 and Fig. 7-4). In case of a VNT with the same source and destination switches and with enough capacity is available, the L2 PCE would reply with the route for that request. However, this is unlikely to happen so the L2 PCE returns a PCRep message (2) with a NO-PATH object. The controller delegates to the VNTM module updating the VNT, possibly adding more resources to serve the L2 p2mp request. To that end, a PCInit message (3) is sent containing the end points of the requested p2mp connection.

At this point, the VNTM will implement either the light-tree-based or the multi-purpose VNT approach. Regardless of the approach followed, the VNTM sends back a PCRpt message (9) reporting the results. In case of success, the ABNO controller requests a L2 p2mp path computation to the L2 PCE (10), which now finds a feasible route and returns it to the ABNO controller (11). Assuming that the L2 PCE is not active, the ABNO controller module delegates connection set-up to the PM (12) that configures the appropriate rules in each switch.

In the case the light-tree-based VNT approach is followed, upon reception of message (3) the VNTM sends a PCReq message (message 4a in Fig. 7-3) to the L0 PCE to create optical connectivity among the specified end points; in this particular approach, a p2mp optical connection needs to be created.

Because L0 p2mp path computation might take long time, L0 PCE delegates it to the specialized bPCE (5a). When the computation ends, the bPCE sends back the solution to the L0 PCE (6a). When the PCE receives the solution, it sends the appropriate commands to the underlying data plane (7a). When the L0 PCE receives the confirmation from the data plane, a PCRep message (8a) is sent back to the VNTM.

On the contrary, if the Multi-purpose VNT approach is adopted, upon reception of message (3) the VNTM needs to reconfigure the VNT, e.g. by adding some new virtual links. Similarly as in the previous case with the p2mp RSA problem, since computing a solution for VNT reconfiguration might take long time, the VNTM delegates it to the specialized bPCE, sending a PCReq message (message 4b in Fig. 7-4). When the computation ends, the bPCE sends back the solution to the VNTM (5b) specifying the lightpaths to be set-up including the route and spectrum allocation for each of them.

When the VNTM receives the solution, it delegates its setting-up to the L0 PCE (6b), which in turn sends the appropriate commands to the PM module to establish the computed lightpaths in the underlying data plane (7b). When the L0 PCE receives the confirmation from the PM module, a PCRep message (8b) is sent back to the VNTM.

Let us now focus on analyzing the differences between PCReq and PCRep messages (5a) and (6a) in the light-tree-based VNT approach and (4b) and (5b) in the multi-purpose VNT approach.

In the light-tree-based VNT approach, PCEP messages follow PCEP standards for p2mp LSPs [RFC6006]. PCReq message (5a) requests a single L0 p2mp LSP. The end points can be specified using an ENDPOINTS class 3 (p2mp IPv4) object, which includes the IP addresses of the source and the leaves.

PCRep message (6a) specifies the computed RSA for the p2mp LSP using ERO and secondary ERO (SERO) objects; one single ERO object is used to define the route and spectrum allocation from the source node to one of the leaves while additional SERO objects define the route and spectrum allocation for each of the rest of leaves; the starting node in each SERO object can be whatever node already in the ERO or SERO objects.

In contrast, standardization under the multi-purpose VNT approach is currently scarce and incomplete. Note that in this approach the VNTM needs to reconfigure the VNT to serve the incoming multicast request. Nonetheless, the exact L0 LSPs to be created to increase the capacity of the VNT are the result of the optimization algorithm running in the bPCE. Hence, the PCReq message (4b) includes the multicast request with some additional objects adapted from [draft-intlayer].

The INTER-LAYER object indicates whether inter-layer path computation is allowed, whereas the SWITCH-LAYER object specifies the layers to be considered, in our case L0 as well as L2. Additionally, two objective function (OF) objects are included; the first OF object specifies the objective function for L2, whilst the second one refers to L0 and includes a TLV with the bandwidth of the L0 LSPs to be created. Table 7-5 depicts the contents of the PCReq message (4b).

Table 7-5 PCReq Message (4b) Contents

```
<PCReq Message>::= <Common Header>
                   <request-list>

where:
   <request-list>::= <request>
                     [<request-list>]

   <request>::= <RP>
                <END-POINTS>
                <BANDWIDTH>
                <of-list>
                <INTER-LAYER>
                <SWITCH-LAYER>
where:
   <of-list>::= <OF>[<of-list>]
```

Table 7-6 PCRep Message (5b) Contents

```
<PCRep Message> ::= <Common Header>
                    <response-list>
where:
   <response-list>::= <response>
                      [<response-list>]

   <response>::= <RP>
                 [<NO-PATH>]
                 [<ERO>]
                 [<server-layer path-list>]

   <server-layer path-list>::= <server-layer path>
                               [<server-layer path-list>]

   <server-layer path>::= <ERO>
                          <BANDWIDTH>
                          <SERVER-INDICATION>
```

Regarding the PCRep message (5b) (Table 7-6), it is worth highlighting that `ERO` objects can be used to point out paths computed on the requested layer and in server layers. In the latter case, a `SERVER-INDICATION` object is used to specify the layer in which a path has been computed among those allowed in the incoming `SWITCH-LAYER` object.

7.4 Validation

In this section, we first evaluate the performance of path and tree schemes under the two approaches and then, experimentally validate the workflows and PCEP considerations in a distributed test-bed facility.

7.4.1 Performance Evaluation

The performance of the two approaches is compared on three realistic national network topologies: the 12-node Deustche Telekom (DT), the 22-node British Telecom (BT), and the 30-node Telefonica (TEL). For these experiments, the optical spectrum width is fixed to 2 THz in each link and all connections use the QPSK modulation format. Over such optical topologies, 6 DCs have been placed in areas with high population density; we consider that connections are requested among those DCs.

A dynamic network environment was simulated where incoming multicast requests arrive to the system following a Poisson process and are sequentially served without prior knowledge of future incoming requests. The number of nodes in each

multicast request was selected in the range [3, 6], and the DCs are chosen using a uniform distribution among all DCs. The bitrate was fixed for every request to 100 Gb/s. The time between two consecutive request arrivals and the holding time of an accepted request were randomly generated following an exponential distribution. Different values of offered network load, computed as the total bitrate of the connections being served, were considered by changing the arrival rate.

The algorithms described in Section 7.2 were used to solve the McP problem for the tree scheme. For comparison purposes, the path scheme was also considered in the experiments. A multicast request involving n destinations is divided into n unicast requests from the source to each single destination. Note that, for the sake of a fair comparison, a multicast request is rejected if one or more unicast requests cannot be served. To solve the McP-SL-path problem, a dynamic RSA algorithm based on the bulk RSA proposed in [Ve14-2] was implemented. Note that this algorithm was used for finding lightpaths for new virtual links in the McP-ML-tree procedure in Table 7-3. As for the McP-ML-path problem, a reduced variation of the bulk RSA algorithm without spectrum assignment was adapted to find minimum cost paths over the virtual topology. In the SL approach, we assumed that SBVTs with 4 flows (f=4) and a switching capacity of 400 Gb/s (B=400 Gb/s) are equipped in the DC routers, whereas in the ML approach, to take advantage of grooming, the virtual topology is supported by 400 Gb/s lightpaths and therefore, 400 Gb/s BVTs are equipped.

Fig. 7-5 plots the blocking probability (Pb) as a function of the offered load for the schemes and approaches considered on the three national topologies. Loads are normalized with respect to the load that causes a Pb=1% under the McP-ML- tree scheme. For this set of experiments, we assumed that enough transponders are equipped at each node to ensure that blocking is as a result of the lack of spectrum resources.

The McP-ML-tree scheme provides the best performance on every topology as a result of its efficient resource usage that combines low resource consumption of the tree scheme and high efficient capacity utilization of the ML approach. In contrast, the McP-SL-tree scheme does not achieve good performance in every topology; whilst it performs similarly than the McP-ML-tree on the DT topology, it shows a noticeable worse performance in the TEL network, even worse than that of the McP-ML-path scheme. The rationale behind this is related to the spectrum continuity constraint and the size of the network. In fact, finding a light-tree under the spectrum continuity constraint poses a challenge in large networks, where the number of hops increases.

As for the paths schemes, the McP-SL-path performs the worst in every network topology, and although the path scheme performs better under the ML approach, it is far from that of the McP-ML-tree.

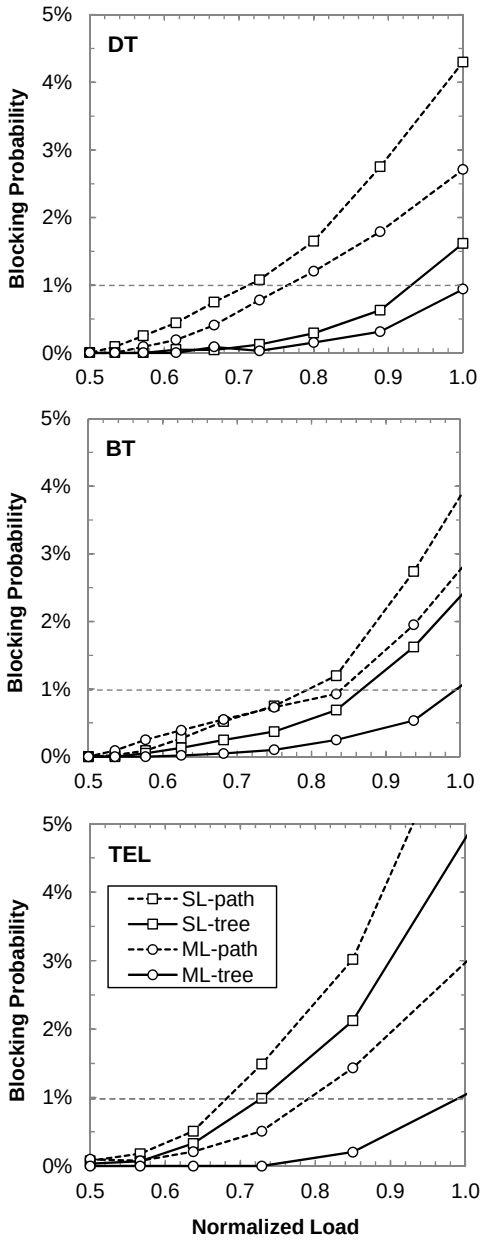


Fig. 7-5 Blocking Probability vs normalized load.

Let us now study the cost of the schemes related to transponders. Recall that SBVTs are installed in DC routers under the SL approach, whereas BVTs are equipped under the ML approach. Aiming at comparing transponder cost for a given multicast traffic load under the different schemes, we accounted for the number of transponders to ensure $Pb \leq 1\%$. Fig. 7-6 shows total transponders cost for several SBVT/BVT cost ratios for the highest normalized loads unleashing

Pb≤1% for each considered network topology. Note that since schemes under the ML approach use BVTs its cost does not depend on SBVT/BVT ratio.

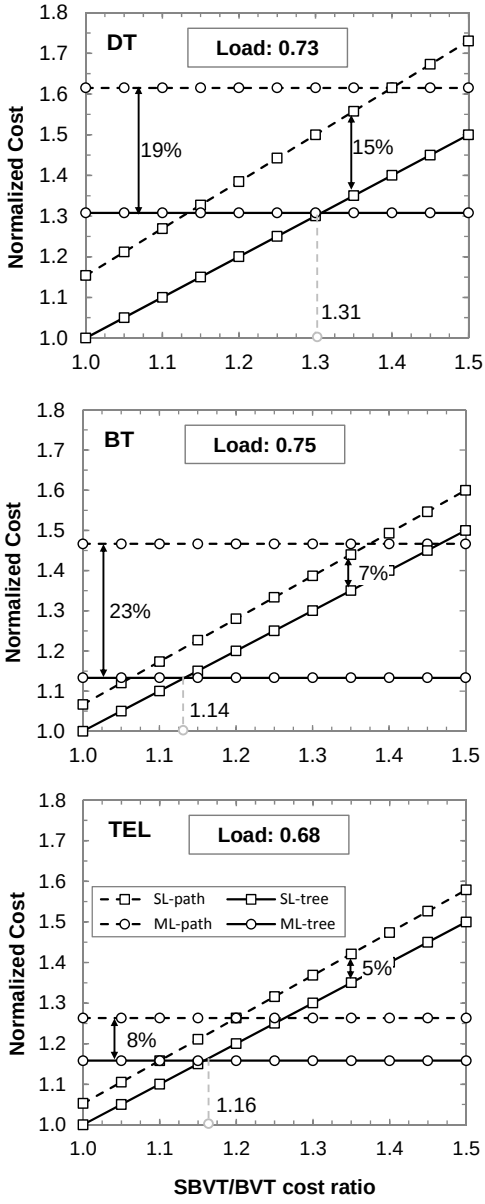


Fig. 7-6 Transponder cost analysis.

It is clear, in view of Fig. 7-6, that the tree scheme brings cost savings compared to the path one since the number of transponders need to be installed in the DC routers is lower. Cost savings of the tree scheme range between 5 and 15% and between 8 and 23% under the SL and the ML approach, respectively. Comparing the tree scheme under the both approaches, the SL approach present cost savings

as long as the cost of a SBVT is not higher than 30% that of an BVT for the DT topology and not higher than 14-16% for the BT and TEL networks. Above those ratios, the ML approach provides costs savings with respect to the SL approach.

7.4.2 Experimental Assessment

The experimental assessment was carried out on a distributed field trial set-up connecting premises in Telefonica (Madrid, Spain), CNIT (Pisa, Italy), and UPC (Barcelona, Spain) (see Fig. 7-7). All three locations are connected through IPSec tunnels and PCEP sessions are established on top of them.

The Cloud computing manager is OpenStack Grizzly [OpenStack]. Neutron plugin is in charge of providing the local overlay networks. As Neutron is a technology dependent plugin, a customized Neutron plugin has been developed to manage the inter-data center connectivity using an SDN approach; the plug-in interacts with the local SDN controller and the ASO. ASO and each of the OpenStack instances exchange information via the Neutron plugin, so the ASO is aware of which VMs in each DC belong to the same network.

OpenStack and Neutron plugin, most of the ABNO modules –the controller, the VNTM, the PM, and the L2 PCE– and the L2 data plane are located in Telefonica's premises. The L2 data plane consists of HP 5406zl Ethernet switches; the PM module configures the switches using OpenFlow via a Floodlight [Floodlight] controller.

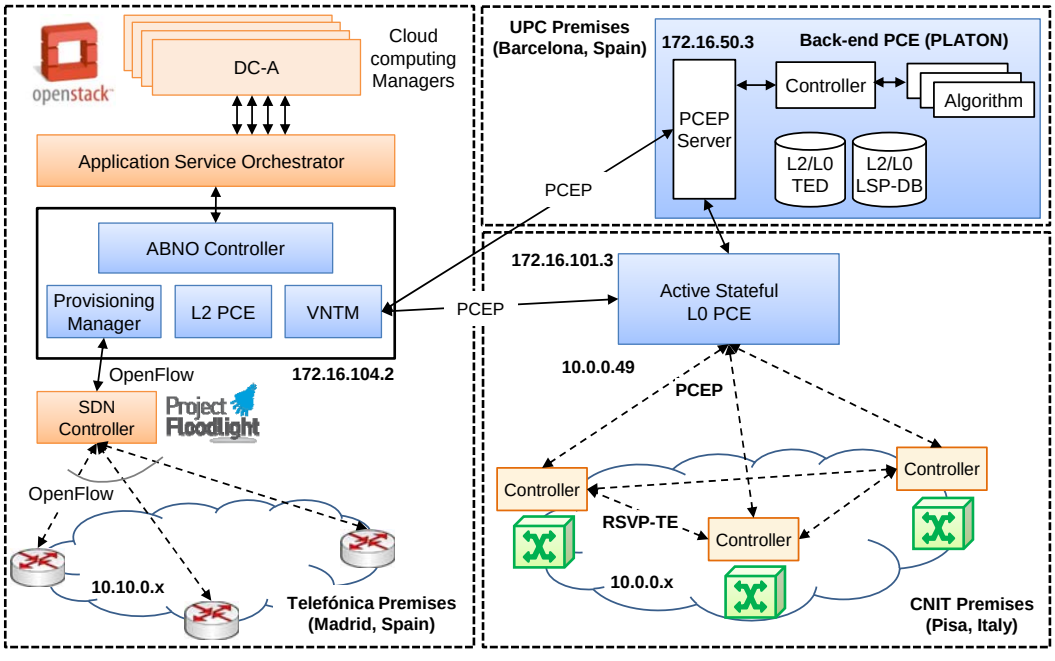


Fig. 7-7 Distributed test-bed set-up. IP addresses are shown.

The L0 data plane includes four programmable SSS; to complete the network topology, six node emulators are additionally deployed. Nodes are handled by co-located GMPLS controllers running RSVP-TE with flexgrid extensions [Cu13-1]. GMPLS controllers communicate with the L0 PCE by means of PCEP.

Finally, the algorithms presented in section 7.2 are implemented in C++ and deployed in our PLATON bPCE based on the iONE module presented in Chapter 4.

7.4.2.1 Light-tree-based VNT approach

Fig. 7-8 shows the relevant PCEP messages for the light-tree-based VNT workflow. Each message is identified with the same sequence used to describe the workflow. The details of messages (5a) and (6a) are given in Fig. 7-9 and Fig. 7-10, respectively. PCReq message (5a) requests a single L0 p2mp LSP. The end points are specified using an `ENDPOINTS` object, which includes the IP addresses of source (X1) and leaves (X2, X3, and X4).

PCRep message (6a) contains the computed RSA for the L0 LSP. In line with [RFC6006], aiming at describing p2mp routes in an efficient way, we use one single `ERO` object and additional `SERO` objects to define the route and spectrum allocation. To illustrate that, the route defined by the `ERO` in Fig. 7-10 is for leaf X2 and includes X6 as intermediate node. The first `SERO` object is for leaf X3 and starts in X6. Since we computed transparent light-trees, the spectrum allocation must be continuous alongside the whole light-tree.

	Source	Destination	Info
1	172.16.104.2	172.16.104.2	PATH COMPUTATION REQUEST MESSAGE
2	172.16.104.2	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
3	172.16.104.2	172.16.104.2	INITIATE MESSAGE
4a	172.16.104.2	172.16.101.3	PATH COMPUTATION REQUEST MESSAGE
5a	172.16.101.3	172.16.50.3	PATH COMPUTATION REQUEST MESSAGE
6a	172.16.50.3	172.16.101.3	PATH COMPUTATION REPLY MESSAGE
7a	10.0.0.49	10.0.0.1	INITIATE MESSAGE
	10.0.0.1	10.0.0.49	REPORT MESSAGE
8a	172.16.101.3	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
9	172.16.104.2	172.16.104.2	REPORT MESSAGE
10	172.16.104.2	172.16.104.2	PATH COMPUTATION REQUEST MESSAGE
11	172.16.104.2	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
12	172.16.104.2	172.16.104.2	INITIATE MESSAGE
	172.16.104.2	172.16.104.2	REPORT MESSAGE

Fig. 7-8 Relevant PCEP Messages for the light-tree-based VNT approach.

```
5a END-POINT object
     Object Class: END-POINT OBJECT (4)
     Object Type: P2MP IPv4 (3)
     Leaf Type: New leaves to add (1)
     Source IPv4 address: 10.0.0.1
     Destination IPv4 address 1: 10.0.0.2
     Destination IPv4 address 2: 10.0.0.3
     Destination IPv4 address 3: 10.0.0.4
```

Fig. 7-9 Details of p2mp object in PCEP Message (5a).

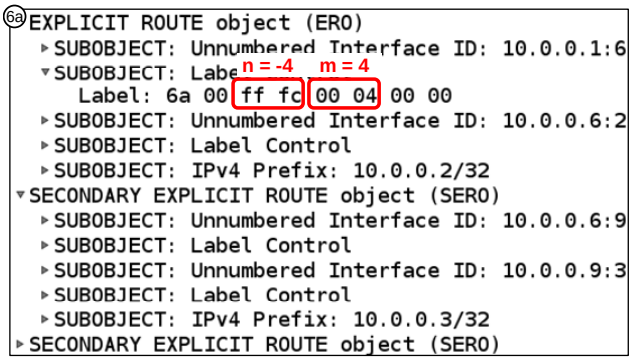


Fig. 7-10 Details of p2mp objects in PCEP Message (6a).

7.4.2.2 Multi-purpose VNT approach

Fig. 7-11 shows the relevant PCEP messages for the multi-purpose VNT workflow. The details of messages (4b) and (5b) are depicted in Fig. 7-12 and Fig. 7-13, respectively.

PCReq message (4b) includes an ENDPOINTS object specifying the L2 end-points, i.e. source (DC-A) and leaves (DC-B, DC-C, and DC-D) L2 switches. In addition, the BANDWIDTH object specifies the requested bandwidth for the multicast service (100 Gb/s).

	Source	Destination	Info
1	172.16.104.2	172.16.104.2	PATH COMPUTATION REQUEST MESSAGE
2	172.16.104.2	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
3	172.16.104.2	172.16.104.2	INITIATE MESSAGE
4b	172.16.104.2	172.16.50.3	PATH COMPUTATION REQUEST MESSAGE
5b	172.16.50.3	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
6b	172.16.104.2	172.16.101.3	PATH COMPUTATION REQUEST MESSAGE
7b	10.0.0.49	10.0.0.3	INITIATE MESSAGE
	10.0.0.3	10.0.0.49	REPORT MESSAGE
8b	172.16.101.3	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
9	172.16.104.2	172.16.104.2	REPORT MESSAGE
10	172.16.104.2	172.16.104.2	PATH COMPUTATION REQUEST MESSAGE
11	172.16.104.2	172.16.104.2	PATH COMPUTATION REPLY MESSAGE
12	172.16.104.2	172.16.104.2	INITIATE MESSAGE
	172.16.104.2	172.16.104.2	REPORT MESSAGE

Fig. 7-11 Relevant PCEP Messages for the multi-purpose VNT approach.

Two OF objects are included in message (4b). The first one refers to the same layer than the request (L2) and specifies that a VNT reconfiguration is requested. The second OF object targets L0 and contains in an embedded TLV the bandwidth for the path computation at that layer; in our experiments, 400 Gb/s L0 LSPs are requested to support the VNT thus, favoring resource utilization. The INTER-LAYER object with the flag I set forces inter-layer path computation, and the SWITCH-LAYER object specifies L2 as well as L0 layers to be considered in the computation. For each layer to be traversed the encoding and switching type has to

be defined: Ethernet encoding and Layer-2 Switch Capable, respectively for L2, and Lambda encoding and Lambda-Switch Capable, respectively for L0.

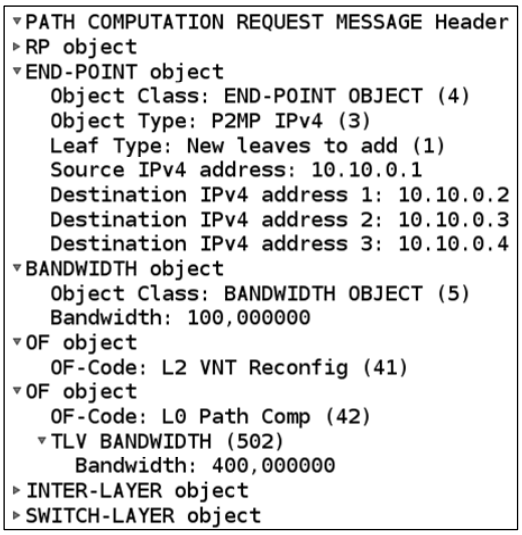


Fig. 7-12 Details of PCReq message (4b).

PCRep message (5b) includes one ERO object for the L2 multicast request. We decided to use loose hops here, since the results of the L2 path computation will be discarded; a subsequent path computation will be requested by the controller to the L2 PCE (messages 10 and 11 in the workflows). After this first ERO object for L2, the list of paths in L0 are included. In the example, one single L0 LSP needs to be created to reconfigure the VNT as shown in the capture in Fig. 7-13. The path includes a SERVER-INDICATION object specifying L0 and an ERO object with the route (X3-X10-X2) and the spectrum allocation for the 400 Gb/s lightpath to be established.

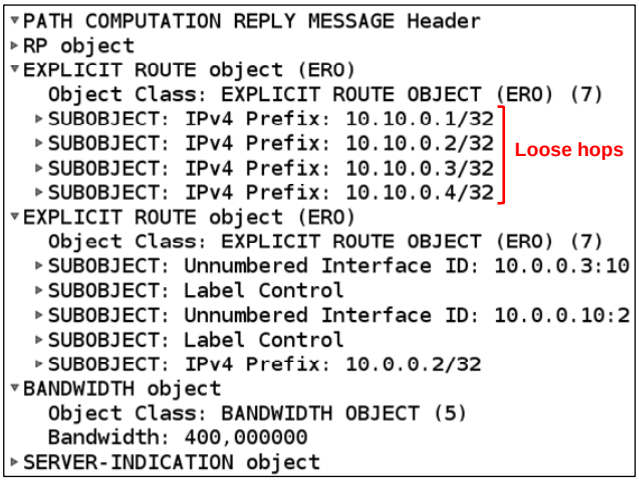


Fig. 7-13 Details of PCRep message (5b).

Finally, the ABNO contribution to the provisioning process under both approaches (i.e. messages exchange and path computation algorithms at bPCE) lasted around 150ms.

7.5 Conclusions

Two approaches have been proposed to serve large capacity (e.g. 100 Gb/s) multicast connectivity services in a multi-layer scenario, where a set of DCs is connected through a flexgrid optical network. In the first approach, a VNT was created for each multicast request by establishing light-trees in the flexgrid network. In the second approach, multicast services are served on a multi-purpose VNT supported by 400 Gb/s lightpaths, thus favoring resource utilization.

Results obtained by simulation validated the ML-tree scheme as the best option to serve a mix of multicast and unicast traffic. Next, the feasibility of implementing both approaches using standardized control plane protocols was studied. Workflows were proposed for the considered approaches. Multicast connections (including light-trees) are supported in PCEP using p2mp extensions. However, VNT reconfiguration, entailing multi-layer computation is not currently supported; an IETF draft was used as a guide.

The next chapter continues with another application of optical networks for datacenter interconnection, where in-operation planning algorithms are, as well required.

Chapter 8

Content Distribution in the Telecom Cloud

The telecom Cloud infrastructure has been proposed to deal with expected traffic growth coming from services like video distribution. Since those services entail large capacity connections from users to DCs, the Telecom Cloud infrastructure can place contents closer to the users by deploying DCs in metro areas, thus reducing the impact of the traffic between users and DCs.

In such scenario, we propose a hierarchical content distribution architecture for the telecom cloud where core DCs, placed in geographically distributed locations, are interconnected through permanent "per content provider" (CP) VNTs (CP-VNT). Additionally, metro DCs need to be interconnected with the core DCs. CP's data is replicated in the core DCs through the CP-VNTs, while metro-to-core (M2C) *anycast* connections are established periodically for content synchronization. Since network failures might disconnect the CP-VNTs, recovery mechanisms are proposed to reconnect both topologies and anycast connections.

8.1 Hierarchical Content Distribution Architecture

The content distribution architecture under study is hierarchical: content replicas are first distributed over a set of core DCs and then, to reduce the traffic in the core, a set of metro DCs are placed closer to the users. Fig. 8-1 shows the architecture for a single CP. A CP-VNT is created by establishing *soft-permanent* [G.8080] Core-to-Core (C2C) virtual links connecting core DCs; this CP-VNT allows core DCs to be permanently synchronized, i.e. any modification performed in the

contents in one core DC is propagated to the rest of the core DCs through the CP-VNT. In contrast, contents in metro DCs are synchronized periodically setting-up Metro-to-Core (M2C) connections to any of the DCs belonging to the CP-VNT (anycast connections).

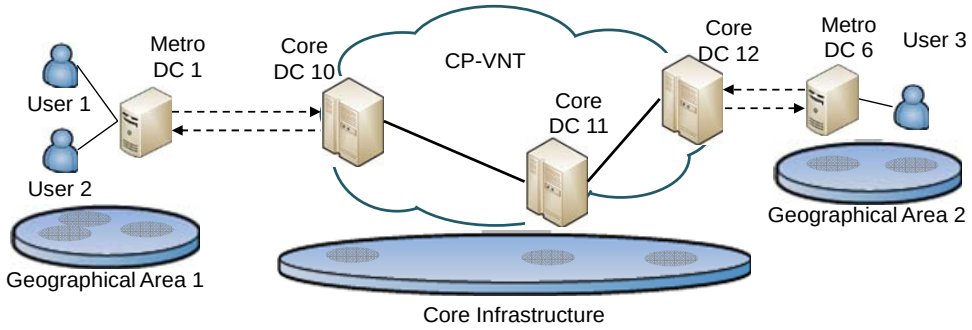


Fig. 8-1 Hierarchical Content Distribution Architecture.

To illustrate how the architecture operates, let us assume a scenario where a distributed database is deployed for research purposes and users (e.g., a research center, a hospital or a university) interact with contents. In the example, user 1, user 2 and metro DC-1 are placed in geographical area 1, while user 3 and metro DC-6 are in geographical area 2. User 1 interacts with the contents in metro DC-1 and uploads new content; user 2 will see the updated content immediately. Conversely, user 3 will have access to the updated content only when contents are synchronized to the core DCs and metro DC-6 synchronizes its contents with the core. To that end, after a synchronization event is triggered, a M2C connection is set-up between metro DC-1 and any of the core DCs in the CP-VNT to synchronize contents. Likewise, a M2C connection needs to be set-up from metro DC-6, so that its contents are synchronized and user 3 will have access to the new uploaded content.

The proposed architecture brings benefits, including the reduction of traffic in the core as well as its inherently high availability against DC failures. However, as a result of considering several core DCs, a CP-VNT topology needs to be created among them. A failure affecting the CP-VNT might disconnect it creating several subsets of connected core DCs (connected components) whose contents become outdated. Therefore, the CP-VNT topology should be designed to cope with network failures. We assume the CP-VNTs are created as a tree topology and dynamic reconnection (similar to connection restoration) is used. Similarly, restoration will be used to recover M2C anycast connections from failures.

Fig. 8-2 presents an example with three core DCs (labelled as 10, 11 and 12) and two metro DCs (labelled as 1 and 6); we assume that a Layer 2 (L2) switch connects each DC to the optical transport network. Core DCs are connected among them by a set of C2C virtual links supported by lightpaths set-up over the optical transport network (blue lines in Fig. 8-2a) forming a CP-VNT (Fig. 8-2b). When a metro DC

needs to synchronize its local contents with those in the CP-VNT, it establishes a dynamic M2C connection to any of the core DCs belonging to the CP-VNT. For example, in Fig. 8-2a, two M2C connections (orange lines) are established: 1-10 and 6-11.

In the event of a failure, C2C as well as M2C connections can be affected; in Fig. 8-2c a link failure has torn-down connections 6-11 and 11-12, thus breaking the original virtual network into two connected graph components ({10, 11} and {12}). Therefore, the CP-VNT needs to be urgently reconnected into a single connected topology; only after the CP-VNT is reconnected keeping contents updated in the core, M2C connections will be recovered. In Fig. 8-2c the CP-VNT has been reconnected by creating the new C2C virtual link 10-12. Fig. 8-2d shows the new CP-VNT topology after the reconnection. Now, affected M2C connections can be restored; in Fig. 8-2c the failed anycast connection 6-11 has been restored by creating the connection 6-12, thus *reconnecting* metro DC6 to the CP-VNT.

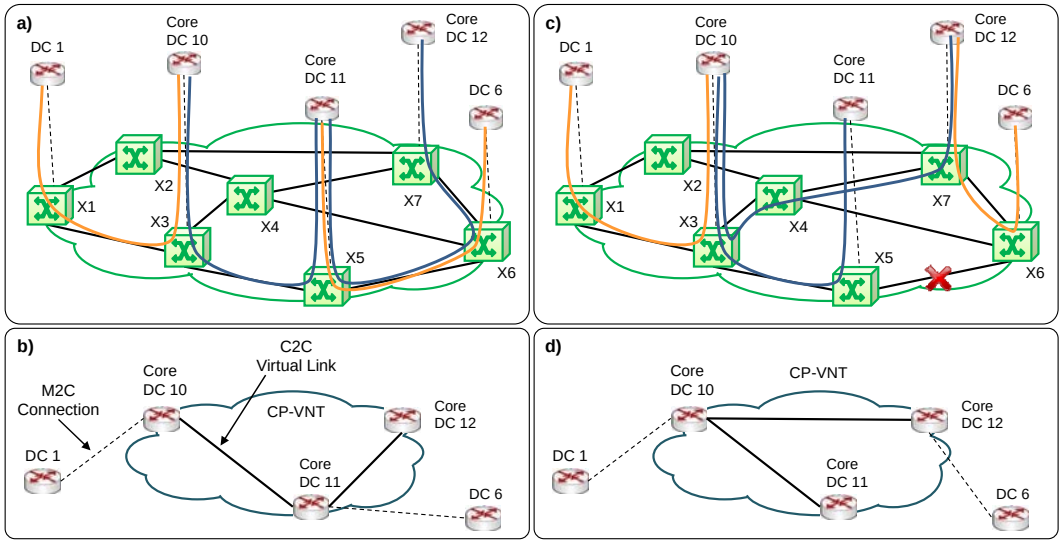


Fig. 8-2 CP-VNT with two M2C anycast connections.

Hence, three problems have been identified: CP-VNT creation, M2C anycast provisioning, and CP-VNT and M2C reconnection. The next section formally defines each problem and proposes mathematical models and algorithms for solving them.

8.2 Mathematical Formulations and Algorithms

8.2.1 CP-VNT_CREATE Problem

This problem consists in finding a tree connecting all core DCs selected by a particular CP. Our approach consists in creating an *auxiliary full-mesh* graph

connecting all involved core DCs. Then, a Minimum Spanning Tree (MST) [Pet02-1] is computed over the auxiliary graph. The links in the resulting tree are the lightpaths that are set-up using a node-link formulation [Ve14-2]; we assume an opaque optical core network, so contiguity and continuity constraints are relaxed. To improve the availability of the CP-VNT, each optical link is used just once on each CP-VNT; therefore, a single link failure would disconnect CP-VNTs into two connected graph components at the most.

The CP-VNT_CREATE problem can be stated as follows:

Given:

- the optical network topology represented by a graph G(N_O, L), where N_O is the set of optical nodes and L the set of optical links.
- the number of available slices η_l in each optical link $l \in L$,
- the CP c that requests the CP-VNT,
- the number of slices χ_l for the virtual links of each VNT.

Output: the route for each bidirectional lightpath.

Objective: minimize the number of optical resources required.

The sets and parameters for this problem (many of them are common for the other problems) are:

Topology:

N set of all nodes in the network, e.g., optical nodes in the network and Ethernet switches in the DCs, index n.

N_O subset of N containing the optical nodes in the network.

N_C subset of N containing the switches in the core DCs.

L set of optical links connecting two nodes, index l.

L_O subset of L containing the optical links connecting two optical nodes.

Content Providers:

C set of CPs, index c.

$N_C(c)$ subset of N_C containing the core DCs for CP c.

E set of candidate virtual links connecting two switches in $N_C(c)$, index e.

$OD(e)$ the subset of $N_C(c)$ containing the endpoints of candidate virtual link e.

Other parameters:

δ_{nl} 1 if link l is incident to node n; 0 otherwise.

χ_1 number of slices required for each C2C connection.

χ_2 number of slices required for each M2C connection.

η_l number of available spectrum slices in link l.

Decision variables:

x_{el} binary, 1 if optical link l is used to support virtual link e in the CP-VNT; 0 otherwise.

y_e binary, 1 if virtual link e is selected; 0 otherwise.

Then, the formulation for the CP-VNT_CREATE problem is as follows:

(CP-VNT_CREATE) $$\min \quad \sum_{e\in E}\sum_{l\in L} x_{el} \tag{8.1}$$

subject to:

$$\sum_{e\in E} y_e = \left|N_C(c)\right| - 1 \tag{8.2}$$

$$\sum_{e\in(S,\bar{S})} y_e \geq 1 \quad \forall S \subset N_C(c) \tag{8.3}$$

$$\sum_{l\in L} \delta_{nl} \cdot x_{el} = y_e \quad \forall e \in E, n \in OD(e) \tag{8.4}$$

$$\sum_{l\in L} \delta_{nl} \cdot x_{el} \leq 2 \quad \forall e \in E, n \in N \setminus OD(e) \tag{8.5}$$

$$\sum_{l'\in L, l'\neq l} \delta_{nl'} \cdot x_{el'} \geq \delta_{nl} \cdot x_{el} \quad \forall e \in E, n \in N \setminus OD(e), l \in L \tag{8.6}$$

$$\sum_{e\in E} \chi_1 \cdot x_{el} \leq \eta_l \quad \forall l \in L \tag{8.7}$$

$$\sum_{e\in E} x_{el} \leq 1 \quad \forall l \in L_O \tag{8.8}$$

The objective function (8.1) minimizes the total number of optical links used to deploy the CP-VNT. Constraints (8.2) and (8.3) define the tree topology by restricting the CP-VNT to have $|N_C(c)|$-1 virtual links and ensuring connectivity of the nodes belonging to the CP-VNT, respectively. Note that constraint (8.3) is applied to any subset S of $N_C(c)$. Constraints (8.4)-(8.6) route the virtual links through the optical network. Constraint (8.4) ensures that one lightpath for each demand is created with end nodes equal to the source and destination of demand. Constraint (8.5) and (8.6) guarantee that the route of each lightpath is connected and it does not contain any loop. Constraint (8.7) ensures that selected optical links

have enough optical resources, while constraint (8.8) guarantees that each optical link is used once at the most.

The CP-VNT_CREATE problem requires a quadratic number of candidate virtual links to be considered with respect to the number of core DCs, i.e. $|E| \sim |N_C(c)|^2$. Thus, the number of variables is $O(|N_C(c)|^2 \cdot |L|)$ and the number of constraints is $O(2^{|NC(c)|}+|N_C(c)|^2 \cdot |N| \cdot |L|)$. For instance, the number of variables and constrains for the 24-node 43-link US topology used in Section 8.4.1, is approximately 10^3 and 10^5, respectively. The size of the problem can be reduced and solved to optimality using large scale optimization techniques, such as column generation [Ru13-1]. In this work, however, a heuristic algorithm is proposed.

The algorithm in Table 8-1 will be reused for the CP-VNT_RECONNECT problem, so the *mode* parameter allows to specify the desired behavior; for the CP-VNT_CREATE algorithm a value of *CREATE* is expected. The algorithm starts creating a *full-mesh* graph $G(N_C(c), E)$ and assigning a metric to each link in E as a function of the estimated number of optical hops for the lightpath supporting that virtual link (lines 1-3). Then, a MST is computed over $G(N_C(c), E)$ (line 6). The links in the tree are iteratively routed over the optical network (lines 9-16) and optical links already used (those in set L_R) are not considered for routing new lightpaths (line 10). If no path is found for a virtual link, it is removed from the *full-mesh* graph and the routing is restarted (lines 11-13); otherwise, the path is added to the *LSP* set and sets are updated (lines 14-16). The solution in *LSP* is eventually returned (line 17).

Table 8-1 CP-VNT_CREATE Heuristic Algorithm

INPUT G(N, L), c, N_C, χ_1, *mode*
OUTPUT *LSP*

```
 1: G(N_C(c), E) ← Create_Full_Mesh_Undirected(N_C(c))
 2: for each e=(a, b) in E do
 3:    e.metric ← |Shortest_Path(G(N_C(c), E), a, b)|
 4: while TRUE do
 5:    L_R ← ∅; LSP ← ∅
 6:    E_T ← MST(G(N_C(c), E));
 7:    if E_T = ∅ then return ∅
 8:    while E_T ≠ ∅ do
 9:       e=(a, b) ← argmax_e.metric(E_T)
10:       p ← Shortest_Path(G(N, L \ L_R), a, b, χ_1)
11:       if p = ∅ then
12:          E ← E \ {e}
13:          break
14:       LSP ← LSP ∪ {p}
15:       if mode=CREATE then L_R ← L_R ∪ p
16:       E_T ← E_T \ {e}
17:    if E_T = ∅ then return LSP
```

8.2.2 M2C-PROV Problem

The M2C-PROV problem aims at finding an anycast connection between a selected metro DC and one of the core DCs in the target CP-VNT. The problem can be stated as follows:

Given:

- the network topology represented by a graph G(*N*, *L*),
- the number of available slices in each link $l \in L$,
- the source metro DC $s \in N$,
- the CP *c* that requests the CP-VNT,
- the number of slices χ_2 required for the anycast connection.

Output: the route for the bidirectional lightpath between *s* and one core DC in $N_C(c)$.

Objective: minimize the number of optical resources required.

The approach we devised to solve this problem consists in building an auxiliary graph with links connecting the metro DC to all the core DCs in the CP-VNT. As before, the metric of each link is a function of number of optical hops for the lightpath supporting that link. The graph is augmented adding one *dummy* node and links connecting each core DC to the dummy node; see Fig. 8-3a, where the dummy node is labelled as *D*. Then, a shortest path is computed between the source metro DC and the dummy node. The resulting path contains the optimal route for the M2C anycast connection to be established (Fig. 8-3b).

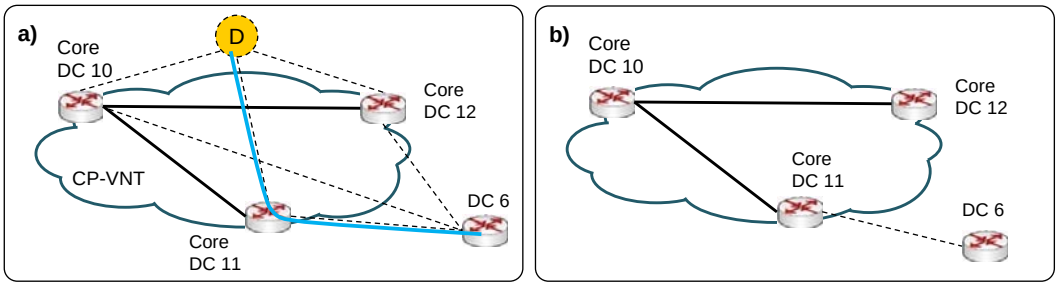


Fig. 8-3 M2C provisioning.

8.2.3 RECONNECT Problem

The RECONNECT problem aims at reconnecting the CP-VNT topologies and the M2C anycast connections affected by a network failure that might involve several nodes and/or links. Recall that reconnecting the CP-VNTs is of first priority to reduce contents in the core becoming outdated.

As for the M2C-PROV problem, our approach for solving this problem consists in creating an augmented graph where *dummy* nodes and links are added. For the CP-VNT reconnection problem, one dummy node for each connected component is added and connected to every core DC in that connected component. Fig. 8-4a illustrates a CP-VNT topology where virtual links 10-11 and 11-12 have failed. Three dummy nodes have been added (labelled as *D1*, *D2*, and *D3* in Fig. 8-4b). Then, finding a tree topology connecting these dummy nodes will reconnect the connected components into a connected CP-VNT topology. We adapt the approach proposed for the CP-VNT creation problem to compute the set of links for the tree and to find feasible lightpaths to support the virtual links (Fig. 8-4c), eventually reconnecting the CP-VNT (Fig. 8-4d).

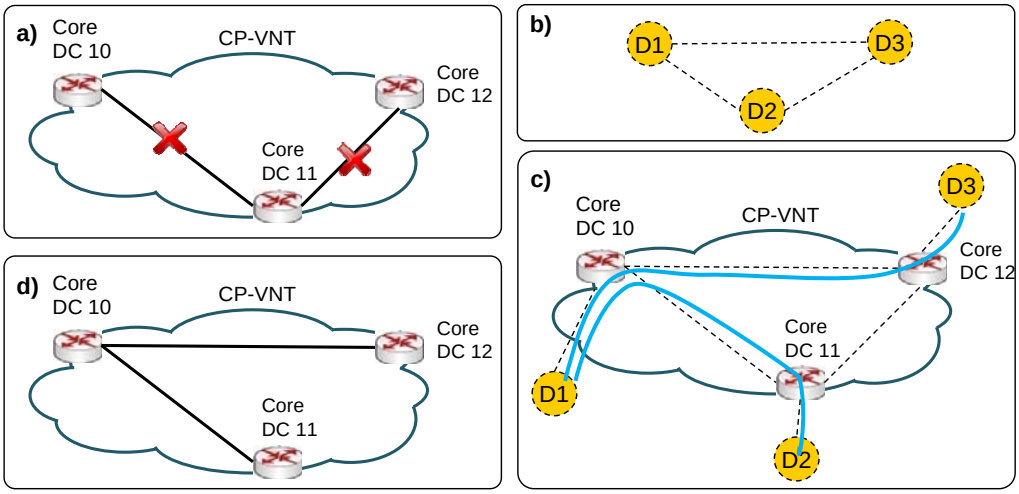


Fig. 8-4 CP-VNT reconnection.

Affected M2C anycast connections are only recovered provided that the associated CP-VNT was either not affected by any failure or already reconnected. Apart from that, the approach is similar to the M2C-PROV problem. Then, the RECONNECT problem can be stated as follows:

Given:

- the network topology represented by a graph G(N, L) where the failed elements (optical links or nodes) have been removed,
- the number of available slices η_l in each link l,
- the set of disconnected CP-VNTs, each of them represented by a graph $G_c(N_C(c), E(c))$, where $E(c)$ is the set of operational virtual links after removing those affected by the failure. It is worth noting that even in the case of a single optical link failure, several virtual links might be affected,
- the set of CP-VNTs not affected by any failure,
- the set D with the failed M2C anycast connections,

- the number χ_1, χ_2 of slices required for each connection type,

Output: the route for the bidirectional LSPs used to reconnect the CP-VNTs and to recover the failed M2C connections.

Objective: maximize the number of connected CP-VNTs and (secondarily) the recovered M2C demands.

In addition to the sets and parameters previously defined, new ones are defined:

CP-VNT Reconnection:

$N_D(c)$ set of dummy nodes corresponding to the connected components in VNT of CP c.

A includes the subset A_C of candidate C2C virtual links connecting pairs of dummy nodes in $N_D(c)$ for CP-VNTs and the subset A_M of M2C virtual links connecting the source of affected anycast connections to the dummy node for its target CP-VNT, index a.

$A_C(c)$ subset of A_C containing the candidate C2C virtual links belonging to the VNT of CP c.

$OD(a)$ subset of N containing the endpoints of virtual link a.

α weight of disconnected CP-VNTs in the objective function.

M2C Recovery:

D set of M2C anycast connections to be reconnected, index d.

$D(c)$ subset of D containing connections belonging to CP c.

$A_M(d)$ the specific M2C virtual link in A_M for anycast connection d.

δ_c 1 if VNT for CP c is already connected; 0 otherwise.

Decision variables:

x_{al} binary. 1 if virtual link a is supported by optical link l; 0 otherwise.

y_a binary. 1 if virtual link a is routed; 0 otherwise.

z_c binary. 1 if VNT of CP c is reconnected; 0 otherwise.

Then, the ILP formulation is as follows:

(RECONNECT) $$\max \ \alpha \cdot \sum_{c \in C} z_c + \sum_{d \in D} \sum_{a \in A_M(d)} y_a \tag{8.9}$$

subject to:

$$\sum_{a \in A_C(c)} y_a \leq |N_D(c)| - 1 \quad \forall c \in C \tag{8.10}$$

$$\sum_{a \in (S,\bar{S})} y_a \geq 1 \quad \forall c \in C, S \subset N_D(c) \tag{8.11}$$

$$\sum_{a \in A_C(c)} y_a \geq \left(\left|N_D(c)\right| - 1\right) \cdot z_c \quad \forall c \in C \tag{8.12}$$

$$\sum_{l \in L} \delta_{nl} \cdot x_{al} = y_a \quad \forall c \in C, a \in A_C(c), n \in OD(a) \tag{8.13}$$

$$\sum_{l \in L} \delta_{nl} \cdot x_{al} \leq 2 \quad \forall c \in C, a \in A_C(c), n \in N \setminus OD(a) \tag{8.14}$$

$$\sum_{l' \in L, l' \neq l} \delta_{nl'} \cdot x_{al'} \geq \delta_{nl} \cdot x_{al} \quad \forall c \in C, a \in A_C(c), n \in N \setminus OD(a), l \in L \tag{8.15}$$

$$y_a \leq z_c + \delta_c \quad \forall c \in C, d \in D(c), a \in A_M(d) \tag{8.16}$$

$$\sum_{l \in L} \delta_{nl} \cdot x_{al} = y_a \quad \forall d \in D, a \in A_M(d), n \in OD(a) \tag{8.17}$$

$$\sum_{l \in L} \delta_{nl} \cdot x_{al} \leq 2 \quad \forall d \in D, a \in A_M(d), n \in N \setminus OD(a) \tag{8.18}$$

$$\sum_{l' \in L, l' \neq l} \delta_{nl'} \cdot x_{al'} \geq \delta_{nl} \cdot x_{al} \quad \forall d \in D, a \in A_M(d), n \in N \setminus OD(a), l \in L \tag{8.19}$$

$$\sum_{c \in C} \sum_{a \in A_C(c)} \chi_1 \cdot x_{al} + \sum_{d \in D} \sum_{a \in A_M(d)} \chi_2 \cdot x_{al} \leq \eta_l \quad \forall l \in L \tag{8.20}$$

The ILP combines a MST formulation to reconnect the failed CP-VNTs with a node-link formulation to set-up the underlying lightpaths. The multi-objective function (8.9) maximizes both, the number of connected CP-VNTs and recovered M2C anycast connections, where the α parameter gives priority to the former. Constraints (8.10)-(8.15) deal with CP-VNT reconnection and are simple extensions to constraints (8.2)-(8.6) presented in section 8.2.1. Constraints (8.16)-(8.20) are responsible for recovering the M2C anycast demands. Constraint (8.16) forces to activate a virtual link related to a M2C connection provided that the corresponding CP-VNT was not affected by failures or it was reconnected. Constraints (8.17)-(8.19) routes the activated links through the optical network. Finally, constraint (8.20) ensures that optical resources used in a link does not exceed its capacity.

The size of the RECONNECT problem is bigger than that of the CP-VNT_CREATE one and it needs to be solved in real time, e.g., less than a second. In view of that, a heuristic algorithm is proposed. Our approach consists in splitting the problem into two sub-problems: CP-VNT_RECONNECT and M2C-RECONNECT. The algorithms are solved sequentially, firstly CP-VNTs are reconnected and then M2C anycast connections are restored.

Table 8-2 presents the pseudo-code of the CP-VNT_RECONNECT heuristic algorithm. It makes use of the CP-VNT_CREATE algorithm to reconnect CP-VNTs by using one dummy node for each connected component of the CP-VNT. The algorithm begins finding, for each disconnected VNT, the set of connected components (line 3), and pre-computing the set of dummy nodes and the links connecting them to the core DCs (lines 4-11). Next, solutions for failed VNTs are generated by randomly sorting the list of CP-VNTs and the best solution is stored (lines 12-19). The best solution found is eventually returned (line 20).

Table 8-2 CP-VNT_RECONNECT Heuristic Algorithm

INPUT $G(N, L)$, C, $G(N_C, E)$, $iter$, χ_1
OUTPUT LSP

```
 1: N_D ← Ø; E_D ← Ø
 2: for each c in C do
 3:    P ← Find_Connected_Components(G(N_C(c), E(c)))
 4:    N_D,c ← Ø; E_D,c ← Ø
 5:    for each p(N_P, E_P) in P do
 6:       v ← Create_Dummy_Node()
 7:       N_D,c ← N_D,c ∪ {v}
 8:       for each n in N_P do
 9:          E_D,c ← E_D,c ∪ {(n, v)}
10:    N_D ← N_D ∪ {N_D,c}
11:    E_D ← E_D ∪ {E_D,c}
12: bestSolution ← Ø
13: for it = 1..iter do
14:    solution ← Ø; C' ← Random_Sort(C)
15:    for each c in C' do
16:       LSP_c ← CPVNT_CREATE(G(N ∪ N_D,c, L ∪ E_D,c), N_D,c, χ1, RECONNECT)
17:       if LSP_c ≠ Ø then solution.paths ← solution.paths ∪ LSP_c
18:       cost ← cost + Cost_Function(LSP_c, |N_D,c|)
19:    bestSolution ← Select_Best(bestSolution, solution)
20: return bestSolution.paths
```

Table 8-3 M2C-RECONNECT Heuristic Algorithm

INPUT $G(N, L)$, $iter$, D, N_C, χ_2
OUTPUT LSP

```
 1: bestSolution ← Ø
 2: for it = 1..iter do
 3:    solution ← Ø
 4:    D' ← Random_Sort(D)
 5:    for each d in D' do
 6:       p ← M2C_PROV(G(N, L), d.src, N_C(d.c), χ2)
 7:       if p ≠ Ø then solution.paths ← solution.paths ∪ {p}
 8:       cost ← cost + Cost_Function(p, |L|)
 9:    bestSolution ← Select_Best(bestSolution, solution)
10: return bestSolution.paths
```

Table 8-3 lists the pseudo-code of the heuristic algorithm for the M2C-RECONNECT sub-problem. It uses the M2C-PROV algorithm to reconnect M2C connections. Solutions are generated by randomly sorting the M2C connection list and keeping the best solution, which is eventually returned.

8.3 Workflows and Protocol Extensions

In this section, we focus on implementing the hierarchical content distribution architecture, i.e. deploying the algorithms for the identified problems on the control and management planes, based on the ABNO architecture. We assume that a bPCE is responsible for dealing with computationally intensive algorithms, such as CP-VNT creation and reconnection algorithms.

Next, workflows for the identified problems as well as related PCEP issues are analyzed and extensions are proposed.

8.3.1 CP-VNT_CREATE Workflow

The CP-VNT_CREATE workflow (Fig. 8-5) is triggered by the operator through the NMS. Its goal is two-fold, first to add the set of core DCs belonging to the CP-VNT and the metro DCs authorized to connect to the CP-VNT to the ASO's database and second to set-up the CP-VNT itself. For the latter, the ASO issues a CP-VNT creation request message (labelled as 1 in Fig. 8-5) to the controller with the set of core DCs to be connected. The controller issues a PCReq message to the fPCE (2), which delegates the CP-VNT topology computation to the bPCE (3). The bPCE runs the CP-VNT_CREATE algorithm and replies the solution in a PCRep message (4). The fPCE issues a PCInit message to the PM so as to implement the related LSPs and waits for the PCRpt messages confirming the implementation of the lightpaths (5). When the CP-VNT topology has been created, the fPCE sends back a PCRep message to the controller (6) with the lightpaths implemented, which replies to the ASO (7) that updates its services database and informs the NMS.

PCReq messages (2) and (3) include one `RP` object for each candidate virtual link connecting pairs of core DCs (specified by the ASO) together with its corresponding P2P `END-POINT` and `BANDWIDTH` objects. To group all the candidate virtual links belonging to the CP-VNT, a `SVEC` object is also included. The topology type, a tree in our case, is defined by adding an `OF` object. `ERO` and `BANDWIDTH` objects are used for the candidate virtual links selected to be part of the CP-VNT. `NO-PATH` objects are used for those virtual links not in the CP-VNT. If no feasible solution was found, all requests are replied with `NO-PATH` objects.

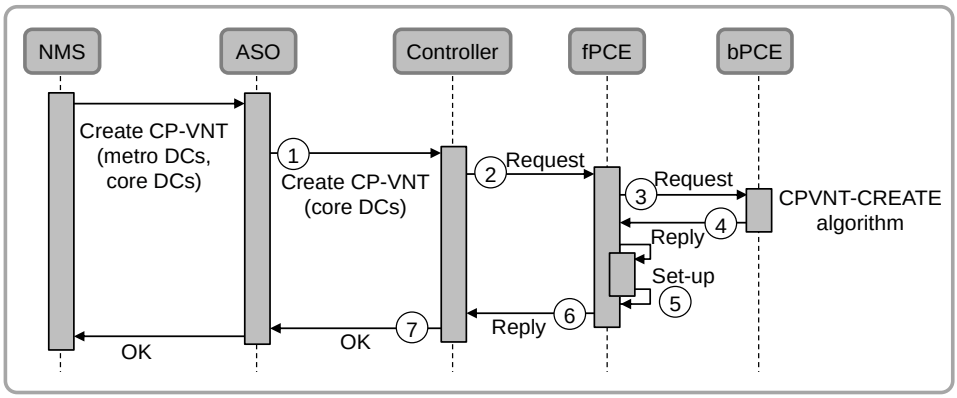


Fig. 8-5 CP-VNT_CREATE workflow.

8.3.2 M2C-PROV Workflow

When a metro DC needs to synchronize contents with the core DCs, the local cloud resource manager issues a M2C connection request to the ASO (Fig. 8-6).

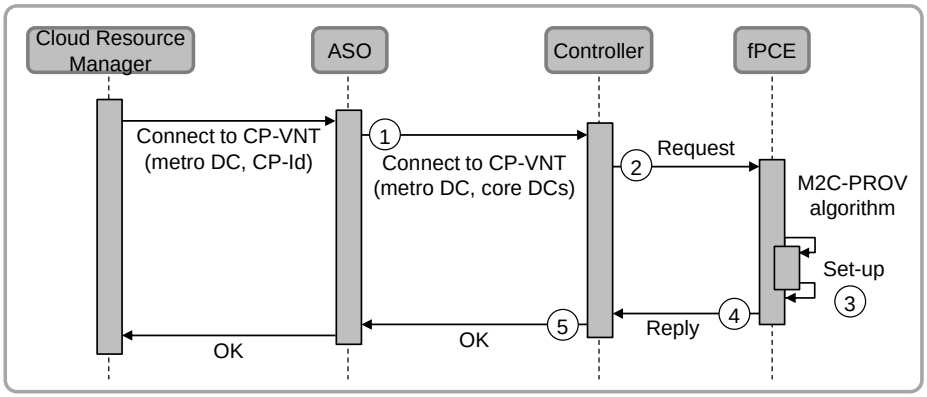


Fig. 8-6 M2C-PROV workflow.

The ASO checks whether the metro DC belongs to the CP and issues a message to the controller with the request (message 1 in Fig. 8-6). The controller creates a PCReq message and forwards it to the fPCE (2), which computes the route for the connection using the M2C-PROV algorithm and issues an initiation request for the LSP to the PM. When the LSP has been set-up, the fPCE replies to the controller (4) that in turn replies to the ASO (5). Finally, the ASO updates its database and replies to the cloud resource manager confirming the availability of the requested M2C connection.

Note that for the anycast M2C path computation the set of candidate core DCs belonging to the target CP-VNT has to be specified. To this end, we propose extending the P2MP `END-POINT` object defined in [RFC6006] with a new leaf type to select one-of-many destinations.

8.3.3 RECONNECT Workflow

The RECONNECT workflow, presented in Fig. 8-7, is initiated by the cloud resource managers, issuing CP-VNT and M2C reconnection requests to the ASO, when their services are affected by a failure. As a result, the ASO might receive multiple reconnection requests.

To accelerate the reconnection process, after receiving a number of reconnection requests the ASO can infer the source of the failure and find the affected services [Cast14-1].

Owing to the fact that the CP-VNTs need to be reconnected before the M2C connections, the ASO groups services and issues separate requests to the controller for CP-VNT and for M2C reconnections. It first issues a CP-VNT reconnection request to the controller (labelled as 1 in Fig. 8-7), which sends a PCReq message to the fPCE (2). CP-VNT reconnection is delegated to the bPCE, so the fPCE forwards the message to the bPCE (3) that runs the CP-VNT-RECONNECT algorithm. Once a solution has been found the bPCE sends a PCRep message (4) to the fPCE. Next, the fPCE issues a request to the PM to implement the lightpaths needed for recovery (5). The PM sends a PCRpt message after each lightpath has been set-up. When all the PCRpt messages related to that CP-VNT have been received, the fPCE informs the controller (6), which in turn informs the ASO (7). The ASO updates its service database, notifies the cloud resource managers, and requests the controller to recover those M2C anycast connections whose target CP-VNT is connected (8); the rest of the workflow is similar as before.

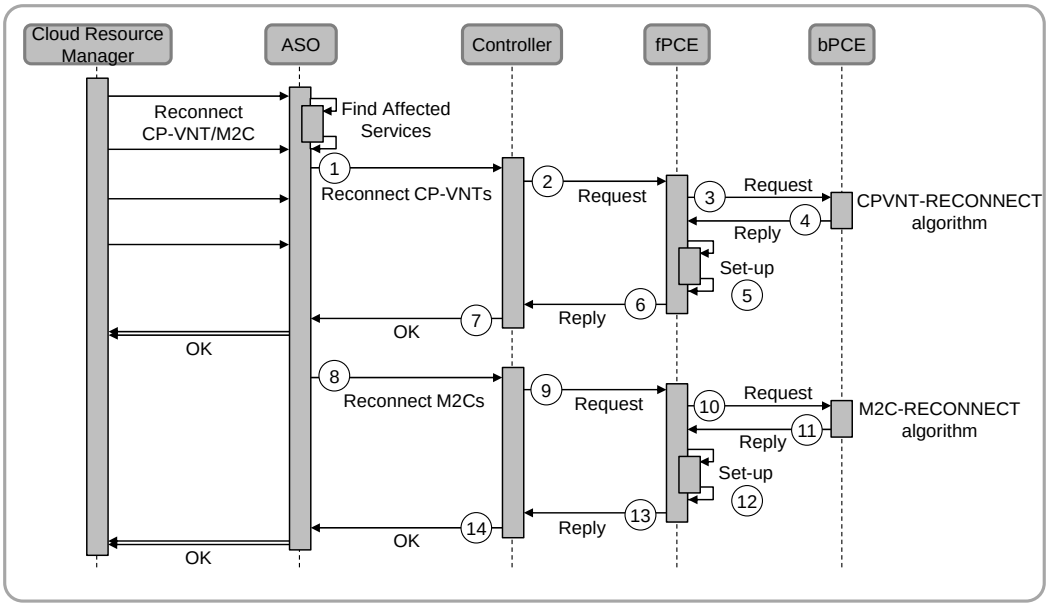


Fig. 8-7 RECONNECT workflow.

Since multiple CP-VNTs could need to be jointly reconnected (see algorithm in Table 8-2), we propose using a hierarchy of `SVEC` objects in PCReq message (2) and (3): the outer pair of `SVEC` and `OF` objects forces a joint computation of all CP-VNT reconnections, while one `SVEC` object per CP-VNT groups the virtual links belonging to that CP-VNT.

In addition, since some CP-VNT's virtual links could remain operational, both, candidate and operational virtual link have to be included in the PCReq messages. To that end, a `Symbolic-Path-Name` TLV is included in each `RP` object for the bPCE to know whether the related LSP already exists.

8.4 Validation

In this section, illustrative simulation results evaluating the proposed algorithms over a realistic network topology are first presented and next, the feasibility of the proposed architecture is experimentally assessed.

8.4.1 Simulation Results

The hierarchical architecture and the algorithms presented in the previous sections have been evaluated on an ad-hoc event-driven simulator based on OMNeT++. To that end, the 24-node 43-link Large Carrier US topology is used (Fig. 8-8), where metro DCs were placed in every node location, while 6 core DCs were strategically placed in New York, Florida, Illinois, Texas, Washington, and California. Each DC is connected to the optical transport network through a L2 switch equipped with 100Gb/s transponders.

A dynamic network environment was simulated, where CP-VNTs are initially created and then 100Gb/s M2C anycast connection requests arrive following a Poisson process. The holding time of the M2C connection requests is exponentially distributed with a mean value equal to 1 h. For each CP-VNT, a metro DC is randomly chosen with equal probability (uniform distribution) among all metro DCs. Different loads are created by changing the arrival rate while keeping the mean holding time constant.

Let us first analyze the M2C provisioning performance. Fig. 8-9a shows the blocking probability as function of the offered load for each CP. Each plot in the graph is for a fixed number of core DCs, ranging from 1 to 5. As observed, the effects of increasing the number of core DCs has a direct effect on the blocking probability; assuming a target 1% blocking probability, 66% more traffic can be served when CP-VNTs consist in 2 core DCs compared to one single DC. The rationale behind this is related to the length of the M2C connections (Fig. 8-9b); the number of hops to reach the core DCs decreases when more core DCs are considered, especially when the selected core DCs are geographically distributed.

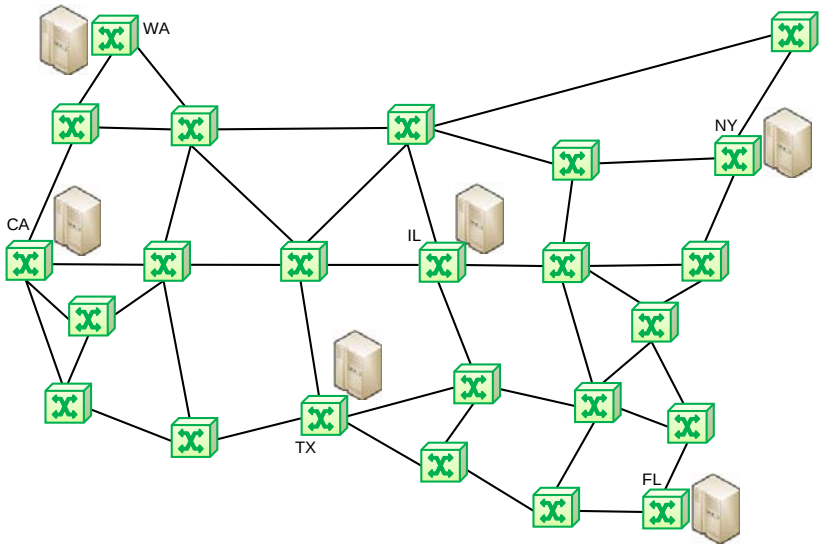


Fig. 8-8 Large Carrier US Network topology, reproduced from [Hu10-1].

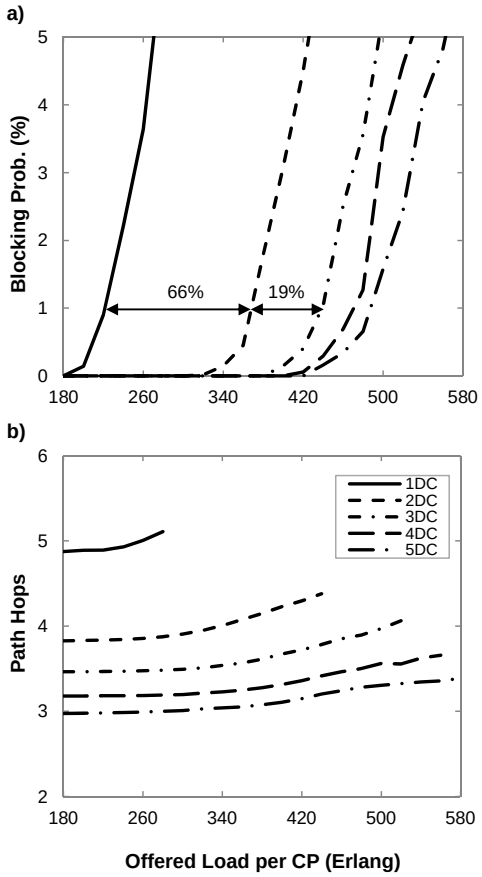


Fig. 8-9 M2C connection provisioning results for 5 CP-VNTs.

Next, let us analyze the effect of increasing the number of CPs, while keeping the total load constant. We selected the load (440 Er) for which having 3 core DCs returns around 1% blocking (see Fig. 8-9a). As shown in Fig. 8-10, increasing the number of CPs has an almost linear effect on blocking, proportional to the number of CP-VNTs to be created. Since more CPs for the same total load entails reducing the load per CP, a network provider should verify the forecast load for that CP to advice about the right size of the CP-VNT.

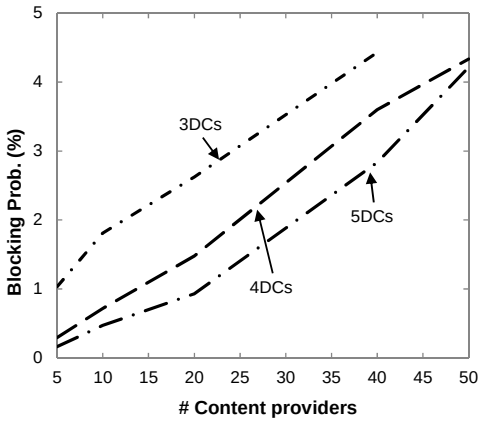


Fig. 8-10 Blocking prob. when increasing the number of CPs.

The restorability of the proposed hierarchical content distribution architecture was evaluated considering single optical link failures, single optical node failures, and double adjacent optical node failures. As illustrated in Fig. 8-11, increasing the number of DCs in the CP-VNT significantly increases restorability; e.g., for an offered load of 300 Erlang, a 99% restorability can be obtained using 2 core DCs under single link failure scenarios, while for single node failures 3 core DCs have to be deployed per CP-VNT to obtain an equivalent restorability. In the 2 adjacent node failures scenario under the same offered load, 95% of restorability can be obtained only when 5 core DCs are deployed per CP-VNT.

Table 8-4 summarizes the increments of traffic load for a 90% of restorability with respect to 1 core DC per CP-VNT, e.g., by deploying 3 core DC CP-VNTs, the traffic load can be doubled under the 2 adjacent node failures scenario while maintaining the rate of restorability of affected demands.

Table 8-4 Load increment (%) w.r.t. 1 DC for 90% restorability

	Failure Type		
	1 Link	**1 Node**	**2 Adj. Nodes**
2 DCs	41.8	38.9	64.3
3 DCs	59.0	65.7	107.8
4 DCs	68.4	74.5	128.6
5 DCs	70.9	81.0	137.7

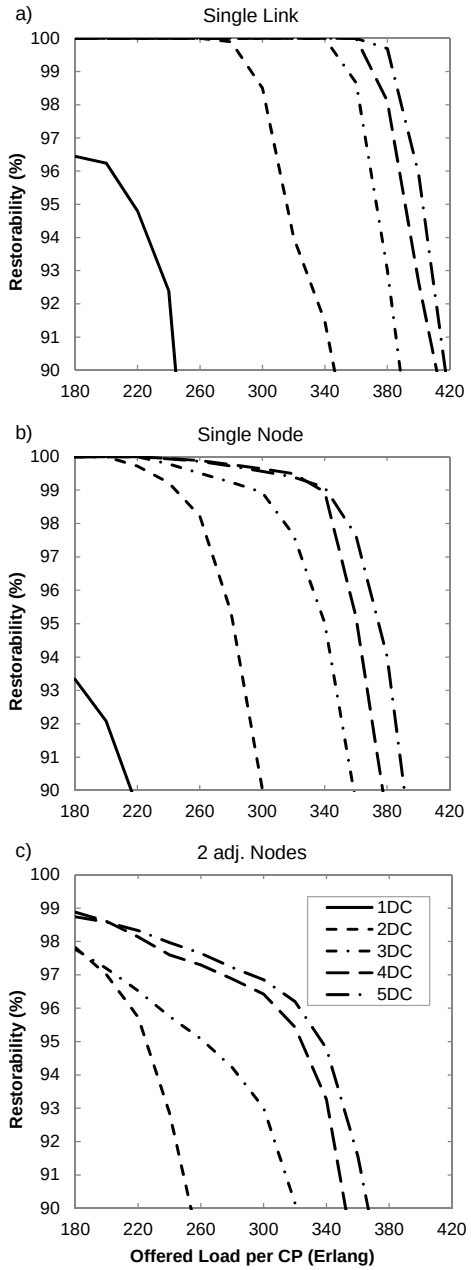


Fig. 8-11 Restorability.

8.4.2 Experimental Assessment

Once the performance of the proposed architecture has been studied, let us now focus on its experimental validation. Experiments were carried out in the UPC's SYNERGY test-bed running the ABNO-based iONE architecture. ASO issues XML-encoded messages to the controller through its HTTP REST API. The

network shown in Fig. 8-2 was assumed, where the optical nodes are in the IP subnetwork 10.0.0.X, while L2 switches for the DCs are in 10.0.1.X. Control and management modules run in the IP subnetwork 172.16.103.X. Specifically, ASO runs in .1, the controller in .2, fPCE in .3, bPCE in .4, and PM in .5. An emulated data plane was deployed for the experiments.

8.4.2.1 CP-VNT_CREATE Workflow

Fig. 8-12 shows the relevant messages for the CP-VNT_CREATE workflow. The messages are identified with the same sequence numbers used in the workflow description. Fig. 8-13 shows PCReq message (3) and PCRep message (4) details. The PCReq message contains a set of `RP`, `END-POINT`, and `BANDWIDTH` objects for each candidate link; an `SVEC` object groups the requests for a joint computation. The PCRep message indicates that 2 virtual links were selected since `ERO` and `BANDWIDTH` objects are included, whereas the other virtual link is discarded. Note in Fig. 8-12 that one single PCInit message was used for the two LSPs to be created, whereas two individual PCRpt reported each individual set-up.

The control plane contribution to the CP-VNT_CREATE workflow processing time was less than 9 ms., including messages exchange and CP-VNT_CREATE algorithm in the bPCE.

	No.	Source	Destination	Protocol	Length	Info
①	509	172.16.103.1	172.16.103.2	HTTP/XML	507	POST /abno/CPVNT_CREATE HTTP/1.1
②	511	172.16.103.2	172.16.103.3	PCEP	220	Path Computation Request
③	513	172.16.103.3	172.16.103.4	PCEP	196	Path Computation Request
④	514	172.16.103.4	172.16.103.3	PCEP	408	Path Computation Reply
	516	172.16.103.3	172.16.103.5	PCEP	444	Initiate
⑤	518	172.16.103.5	172.16.103.3	PCEP	240	Path Computation LSP State Report (PCRpt)
	520	172.16.103.5	172.16.103.3	PCEP	276	Path Computation LSP State Report (PCRpt)
⑥	522	172.16.103.3	172.16.103.2	PCEP	408	Path Computation Reply
⑦	523	172.16.103.2	172.16.103.1	HTTP/XML	1277	HTTP/1.1 200 OK

Fig. 8-12 CP-VNT_CREATE message list.

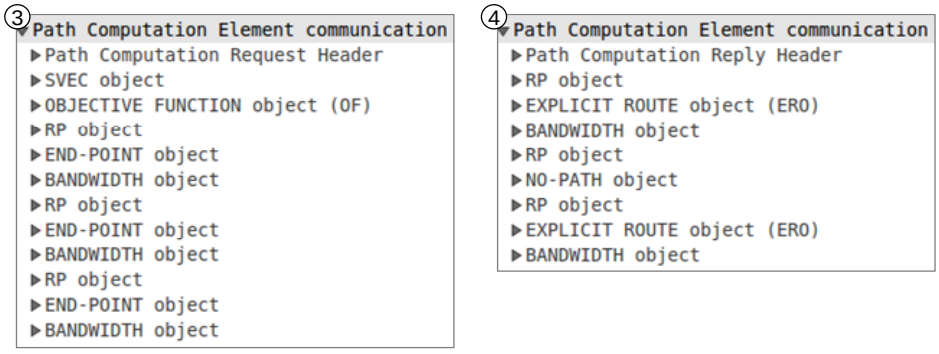


Fig. 8-13 CP-VNT_CREATE PCReq and PCRep messages.

8.4.2.2 M2C-PROV Workflow

Fig. 8-14 shows the relevant messages for the M2C-PROV workflow. Fig. 8-15 shows the P2MP `END-POINT` object included in PCReq message (2); we used the proposed leaf type 5 labelled as *New anycast leave to add* to request an anycast connection from metro DC 1 (10.0.1.1) to any of the core DCs belonging to the target CP-VNT: 10, 11, and 12. In the PCRep message (4), an `ERO` object specifies the route from metro DC 1 to core DC 10 through the optical layer. Note that since the request was for an anycast connection, the route to one single destination is provided.

	No.	Source	Destination	Protocol	Length	Info
①	904	172.16.103.1	172.16.103.2	HTTP/XML	232	POST /abno/M2C_PROV HTTP/1.1
②	906	172.16.103.2	172.16.103.3	PCEP	124	Path Computation Request
③	908	172.16.103.3	172.16.103.5	PCEP	252	Initiate
	910	172.16.103.5	172.16.103.3	PCEP	252	Path Computation LSP State Report (PCRpt)
④	912	172.16.103.3	172.16.103.2	PCEP	212	Path Computation Reply
⑤	914	172.16.103.2	172.16.103.1	HTTP/XML	639	HTTP/1.1 200 OK

Fig. 8-14 M2C-PROV message list.

The control plane contribution to the M2C-PROV workflow processing time was less than 3 ms., including messages exchange and M2C-PROV algorithm in the fPCE.

②
```
Path Computation Element communication Protocol
  Path Computation Request Header
  RP object
  END-POINT object
    Object Class: END-POINT OBJECT (4)
    0011 .... = Object Type: Point-to-MultiPoint IPv4 (3)
    Flags
    Object Length: 24
    Leaf Type: New anycast leave to add (5)
    Source IPv4 Address: 10.0.1.1 (10.0.1.1)
    Destination IPv4 Address: 10.0.1.10 (10.0.1.10)
    Destination IPv4 Address: 10.0.1.11 (10.0.1.11)
    Destination IPv4 Address: 10.0.1.12 (10.0.1.12)
  BANDWIDTH object
```

④
```
Path Computation Element communication Protocol
  Path Computation Reply Header
  RP object
  EXPLICIT ROUTE object (ERO)
    Object Class: EXPLICIT ROUTE OBJECT (ERO) (7)
    0001 .... = Object Type: 1
    Flags
    Object Length: 120
    SUBOBJECT: Unnumbered Interface ID: 10.0.1.1:1
    SUBOBJECT: Label Control
    SUBOBJECT: Label Control
    SUBOBJECT: Unnumbered Interface ID: 10.0.0.1:3
    SUBOBJECT: Label Control
    SUBOBJECT: Label Control
    SUBOBJECT: Unnumbered Interface ID: 10.0.0.3:110
    SUBOBJECT: Label Control
    SUBOBJECT: Label Control
    SUBOBJECT: IPv4 Prefix: 10.0.1.10/32
  BANDWIDTH object
```

Fig. 8-15 M2C-PROV PCReq and PCRep messages.

8.4.2.3 RECONNECT Workflow

Fig. 8-16 shows the relevant messages for the RECONNECT workflow, where messages (1)-(7) are related to CP-VNT reconnection and messages (8)-(14) to M2C anycast connection recovery. Fig. 8-17 shows the details of PCReq message (2). The first `SVEC` object is used to group all the CP-VNTs to be reconnected so as to do a joint computation, as specified in the associated `OF` object. One `SVEC` object is used to group the virtual links of each individual CP-VNT.

	No.	Source	Destination	Protocol	Length	Info
①	1358	172.16.103.1	172.16.103.2	HTTP/XML	533	POST /abno/CPVNT_RECONNECT HTTP/1.1
②	1360	172.16.103.2	172.16.103.3	PCEP	228	Path Computation Request
③	1362	172.16.103.3	172.16.103.4	PCEP	228	Path Computation Request
④	1363	172.16.103.4	172.16.103.3	PCEP	288	Path Computation Reply
⑤	1365	172.16.103.3	172.16.103.5	PCEP	276	Initiate
	1366	172.16.103.5	172.16.103.3	PCEP	276	Path Computation LSP State Report (PCRpt)
⑥	1368	172.16.103.3	172.16.103.2	PCEP	288	Path Computation Reply
⑦	1370	172.16.103.2	172.16.103.1	HTTP/XML	938	HTTP/1.1 200 OK
⑧	1636	172.16.103.1	172.16.103.2	HTTP/XML	236	POST /abno/M2C_RECONNECT HTTP/1.1
⑨	1638	172.16.103.2	172.16.103.3	PCEP	136	Path Computation Request
⑩	1639	172.16.103.3	172.16.103.4	PCEP	128	Path Computation Request
⑪	1640	172.16.103.4	172.16.103.3	PCEP	212	Path Computation Reply
⑫	1642	172.16.103.3	172.16.103.5	PCEP	252	Initiate
	1644	172.16.103.5	172.16.103.3	PCEP	252	Path Computation LSP State Report (PCRpt)
⑬	1646	172.16.103.3	172.16.103.2	PCEP	212	Path Computation Reply
⑭	1648	172.16.103.2	172.16.103.1	HTTP/XML	643	HTTP/1.1 200 OK

Fig. 8-16 RECONNECT message list.

The control plane contribution to the RECONNECT workflow processing time was less than 12 ms., including messages exchange and algorithms. The CP-VNT reconnection sub-workflow took 9 ms., while the M2C recovery took 3 ms.

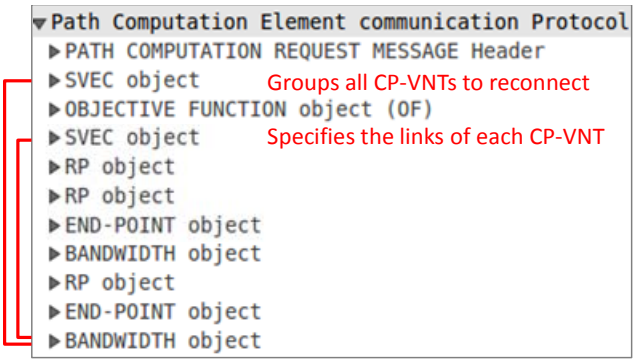


Fig. 8-17 CPVNT-RECONNECT PCReq message (2).

8.5 Conclusions

An architecture for hierarchical content distribution on the telecom cloud was proposed, where core and metro DCs were considered. Core DCs were geographically distributed and interconnected to create a specific CP-VNT topology for each CP. Contents in the core DCs are automatically synchronized using the connectivity provided by the CP-VNT, whereas metro DCs synchronize their contents with core DCs periodically. To that end, M2C anycast connections are dynamically requested.

Three main problems were identified for operating this architecture: the CP-VNT creation, the M2C connection provisioning, and the reconnection of both CP-VNTs and M2C connections after a network failure. These problems were formally stated,

mathematically formulated, and then heuristic algorithms were eventually proposed for solving them in real time. In particular, an exact algorithm for finding anycast shortest paths and efficient heuristic algorithms for creating CP-VNTs and reconnecting connected components of disconnected CP-VNTs were proposed.

Exhaustive simulations were carried out to evaluate the performance of the architecture. The results showed that high efficiency can be achieved when increasing the number of core DCs in the VNT; more than 66% load increment was observed. Furthermore, significant improvement in the network restorability for different types of failures was shown when the number of DCs in the CP-VNTs was increased.

Aiming at experimentally assessing the proposed architecture, an ABNO-based control plane architecture was considered. The ASO module was responsible for service management and issues requests to the ABNO. As a result of the complexity of problems to be solved and the stringent time in which they need to be solved, a bPCE was used. Workflows were developed for the identified problems and the PCEP feasibility for those workflows was studied. PCEP extensions for M2C anycast connection provisioning and joint CP-VNTs reconnection and M2C connection recovery were proposed.

In this and previous chapters, we studied a number of applications of in-operation planning. The next chapter broad our focus and considers the use of inventory databases, as well as operation databases, so as to increment the capacity of in-operation networks.

Chapter 9

On-Demand Incremental Capacity Planning

In this chapter, we propose a pay-as-you-grow approach, where new capacity is installed in accordance with traffic growth, aiming at reducing network expenditures. The approach assumes that a periodical planning cycle is in charge of the design of the network; as a result, new OXCs can be installed and a reduced number of spare line-cards can be purchased and made available in some warehouses distributed over the geography. J*ust-in-time* (JIT) techniques can be used to keep enough spare cards always available. Spare equipment availability is often stored in an *inventory database*, together with information about optical cables, optical amplifiers, fiber usage, etc.

When the capacity of the optical backbone network becomes exhausted in some parts, new capacity needs to be installed in the network to serve the forecast traffic during the next planning period. Based on reliable traffic prediction, new equipment is installed and its capacity is made ready to be used. Nonetheless, due among others to the introduction of new services, exact prediction is not usually available, which leads to installing more capacity than that required thus, increasing network expenditures. To that end, network operators periodically, e.g., monthly, can use a planning tool to run a planning algorithm to decide the equipment and connectivity to be installed at the minimum cost. Hence, the planning tool needs access to both, the inventory database and the current state of the network stored in *operation databases*, i.e. the TED and the LSP-DB.

To reduce expenses, in this section, we propose to increment the capacity of the network as soon as it is required to meet the target performance.

9.1 Incremental Capacity Planning

Fig. 9-1 presents the proposed augmented network life-cycle, where the network performance is monitored and incremental capacity planning is triggered when a threshold has been exceeded. The incremental capacity problem consists of deciding which resources need to be added to the network to ensure some performance metrics. Two network performance metrics are usually of the interest for network operators: the GoS (i.e. blocking probability) and restorability (defined as the ratio between the number of LSPs that are successfully restored and the total number of LSPs to restore [Cl02-1]) under single link failure scenarios.

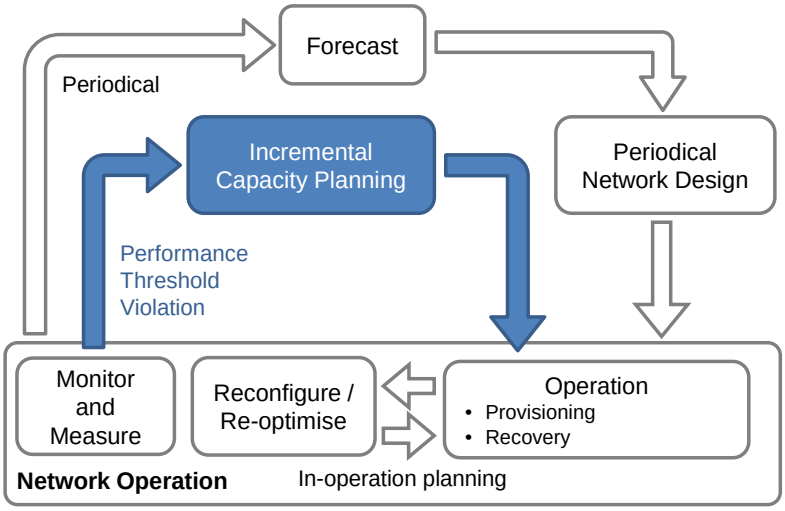


Fig. 9-1 Augmented network life-cycle.

For illustrative purposes, imagine that the network in Fig. 9-2a is currently in operation and a blocking probability threshold has been exceeded. In such case, in-operation network planning can be triggered to re-optimize network resource utilization (see in-operation network planning cycle in Fig. 9-1). In the case that no feasible solution is found for the in-operation planning problem, the only way to reduce blocking probability is by adding new resources to the network. To ensure performance metrics, an incremental capacity planning algorithm might decide adding a new link connecting nodes X4 and X6, as shown in Fig. 9-2b. To add the new fiber link, two spare line-cards must be installed in the end OXCs and connected to available optical fibers connecting end buildings.

To add a new link, the incremental capacity planning algorithm needs to know the current state of the network including state of the resources and established LSPs. Furthermore, it needs information about physical resources, even those not yet installed. In general, we can assume that the physical layout of OXCs consists of a number of chassis containing slots where cards can be plugged into. As for fiber links, we assume that they consist of a number of spans where optical amplifiers

are ready to be used to create end-to-end links. Availability of spare optical fibers, line-cards, card slots, and equipment layout is usually stored in an inventory database together with geographical information about its location. In the example in Fig. 9-2a, the inventory reflects that four spare line-cards and four fibers are available. Consequently, planning algorithms must access such inventory database to decide which line-card must be selected, based e.g., in its geographical location, and which card slot a card must be plugged into. In Fig. 9-2b, a planning algorithm selected line-cards 16 and 28 and fibers 30 and 31 to create fiber link X4-X6.

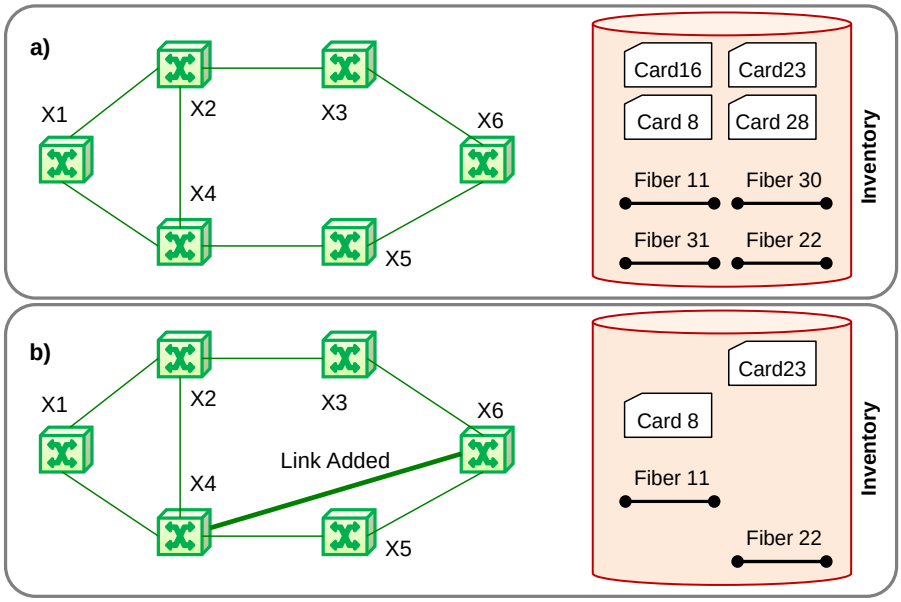


Fig. 9-2 Example of incremental capacity planning.

A general algorithm to prepare data to solve the incremental capacity problem is presented in Table 9-1. It receives the current graph G(N, E) representing the optical network topology, where N is the set of optical nodes and E the set of fiber links connecting two optical nodes in N. The set of demands D currently being served, a connection to the inventory database, and some other parameters, are also received as input data.

Operation and inventory databases are first correlated (lines 1-2 in Table 9-1). Next, free line-cards and fiber links are retrieved from the inventory database (lines 3-4) and an augmented graph $G_x(N, E_x)$ is created by adding new links (set L) to the current graph (line 5). The incremental capacity planning problem is solved using the augmented graph (line 6). Once a solution is found, the list of actions to be performed, e.g., installing line-cards, connecting fiber links to interfaces, activating fiber links, etc., is computed and returned (line 8). Note that those actions require manual intervention, so an operator should redirect them to the most appropriate engineering teams.

Table 9-1 Incremental Capacity Planning Algorithm

```
INPUT G(N,E), D, Inventory, params
OUTPUT ActionList
 1: Eq ← getEquipment(Inventory)
 2: Correlate(N, Eq)
 3: LC ← getFreeLineCards(Inventory)
 4: F ← getFreeFiberLinks(Inventory)
 5: Gx(N, Ex = E ∪ L) ← augmentGraph(G, Eq, LC, F)
 6: S ← solveIncrementalCapacity(Gx, D, params)
 7: if S = ∅ then return INFEASIBLE
 8: return computeActionList(S)
```

Once the new resources are activated and become available to control and management planes, the new capacity can be used to serve traffic and for recovery purposes.

In the next section, an architecture to support on-demand network planning is proposed.

9.2 Proposed Architecture and Inventory Model

In this section we first propose an architecture to support on-demand network planning; it relates the NMS, the operation controller (PCE) containing the TED and the LSP-DB, the planning tool, and the inventory system. Next, we present the proposed inventory model that consists of a set of related elements. Inventory, TED and LSP-DB relationship is analyzed and data access is eventually described.

9.2.1 Architecture for On-demand Network Planning

Fig. 9-3 presents an architecture to support on-demand network planning. The architecture might support any planning algorithm that need to access both, operational and inventory databases. A control plane with a centralized PCE has global view of the resources and network topology as well as of the established connections.

The central element is a planning tool (we use PLATON) that receives planning requests from the NMS. Each request identifies the specific planning algorithm to be executed, e.g. incremental capacity planning. To access data stored in the TED, the planning tool can be implemented as a bPCE being TED data synchronized using the BGP-LS [draft-bgpls] protocol. The LSP-DB can be synchronized using PCRpt messages [draft-stateful] that the PCE can sent asynchronously when the state of an LSP changes. As for the inventory database, it can be modelled using the YANG [RFC6020] language and then, the RESTCONF protocol [draft-restconf] can be used to access inventory data.

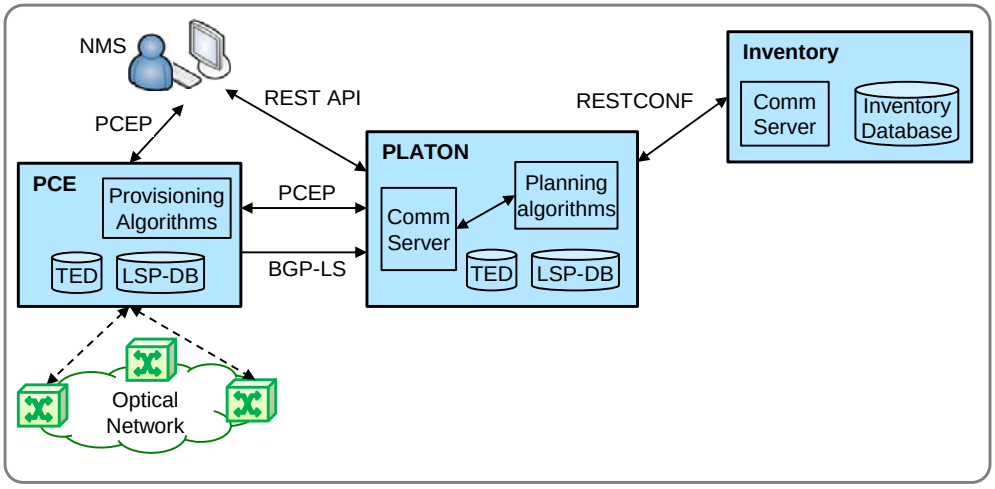


Fig. 9-3 Architecture for on-demand incremental capacity planning.

Incremental capacity planning requests are issued on-demand by the NMS and include a target threshold (message 1 in Fig. 9-4). The planning tool selects the specified algorithm, which issues a number of queries to retrieve data from the inventory (messages 2-5). Incremental capacity planning algorithms use the above queries to find available resources to be added to the augmented graph (6).

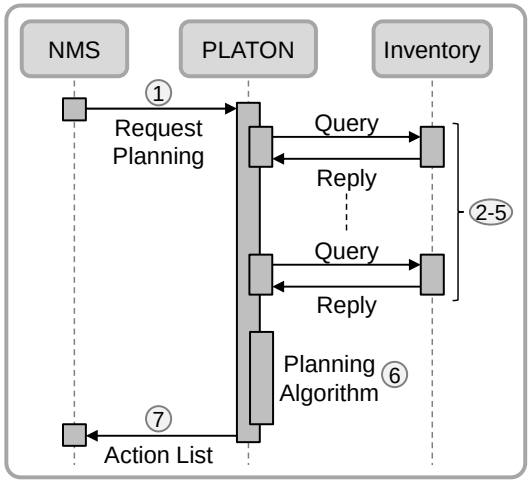


Fig. 9-4 Workflow for on-demand incremental capacity planning.

Once a solution is found, PLATON sends the list of actions to be performed back to the NMS (message 7). The defined actions include:

i) INSTALL: move a specific card from its current location and install it in a specific card slot of an existing equipment;

ii) CONNECT: connect a given port to an existing unused fiber link;

iii) ACTIVATE: put into operation an installed element so it can be discovered by control and management planes.

Actions require manual intervention, so the operator in the NMS should redirect them to the engineering team in charge. Once the new resources are activated and become available to control and management planes, the new capacity can be used to serve traffic and for recovery purposes.

9.2.2 Inventory Model and Relationships

It is worth highlighting that operational and inventory databases need to be used together and therefore, it is of paramount importance that both data models include some shared fields to correlate data between them. For instance, nodes are identified using IP addresses in the TED, so they must be used as well in the inventory to identify the physical equipment implementing each node. To be able to match data collected from different DBs, in this section we propose defining a set of common identifiers to link elements in the different repositories. Fig. 9-5 shows the relationship among elements in the TED, LSP-DB, and inventory. Note that TED and LSP-DB are already related since a LSP is defined as a pair of nodes and a hop list with nodes and interfaces, identified by their *id*.

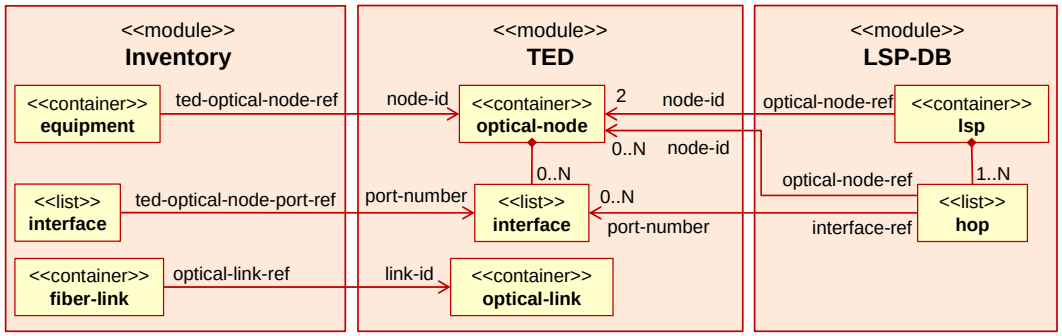


Fig. 9-5 YANG model for the operation and inventory databases relationship.

The inventory DB contains, among others, equipment, line-cards, and optical fibers that need to be related to nodes, interfaces, and links in the TED. Thus, identifiers of the elements in the TED are used in the inventory. Specifically, the *equipment* class includes the *ted-optical-node-ref* attribute linking to the *node-id* attribute of the *optical-node* class in the TED. In addition, an *interface* in the TED can be correlated with a similar class in the inventory, while an *optical-link* in the TED can be correlated with a *fiber* in the inventory.

With the proposed element linkage, planning algorithms can retrieve data and perform computations. For instance, from data about physical locations, a planning algorithm can compute the distances from a node to a set of warehouses where line-cards are stored; that way, the line-card at a shortest distance can be selected.

The proposed inventory model is divided into three main blocks of classes: *i*) hardware-related; *ii*) fiber-related; and *iii*) building-related classes (Fig. 9-6). The hardware-related block includes *equipment*, *line-card*, and *optical-amplifier* classes. The main attributes of these elements are grouped into the so called *hardware-attributes* class. An equipment is modelled as a container of card *slots*. A card (in this version of the inventory model only line-cards are considered) can be plugged into one single card slot provided that the slot and the card are compatible. Hence, the list of compatible card models for each slot is also considered.

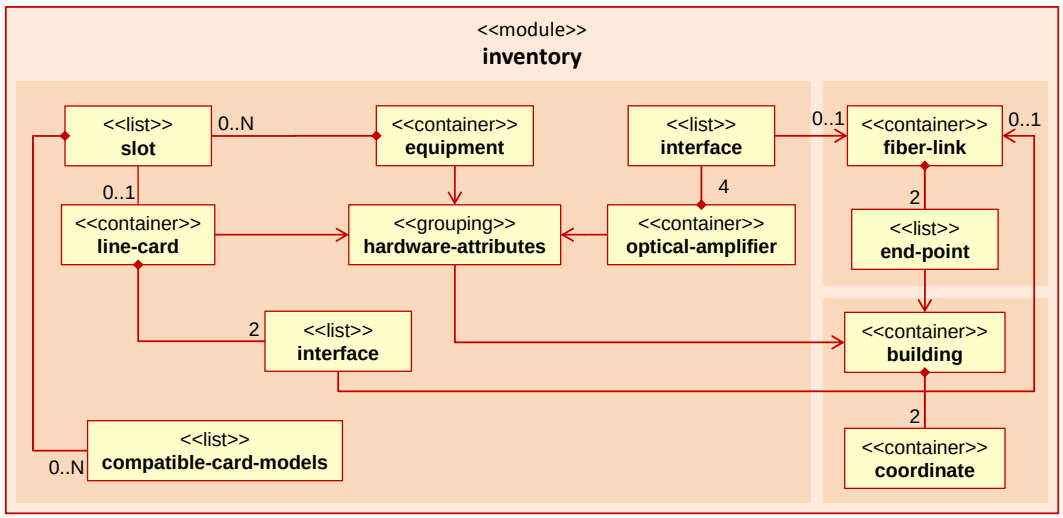


Fig. 9-6 Class diagram summarizing the proposed Inventory model.

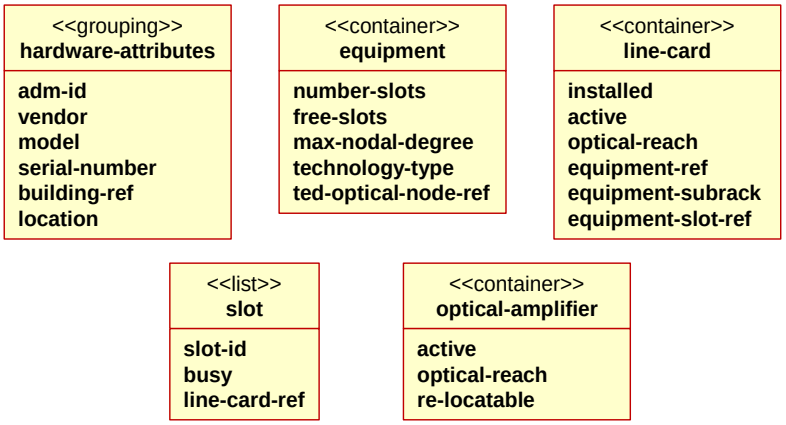


Fig. 9-7 Main Attributes of Hardware-related Classes.

Regarding line-cards, we consider each one with two *interfaces* for optical transmission and reception, with an optical signal reach. Fig. 9-7 summarizes the main attributes of key hardware-related classes. The hardware-attributes class includes the *adm-id* attribute to link each piece of hardware to any other external

DB, and the identifier of the building. Finally, note that the attribute *installed* can be used to identify spare cards.

Fig. 9-8 summarizes the main attributes of some key classes. The fiber-related block includes the *fiber-link* and the *end-point* classes; a fiber link is modelled as a number of spans with already connected optical amplifiers and the total link length. The two end-points of a fiber link include a reference to the building, the exact location of the patch panel within the building and the end-points within the patch panel where the fiber is terminated. Finally, the *building* class includes, in addition to the city and address of the building, its geographical coordinates for planning algorithms to compute distances between two buildings, e.g., using the so called *haversine formula*.

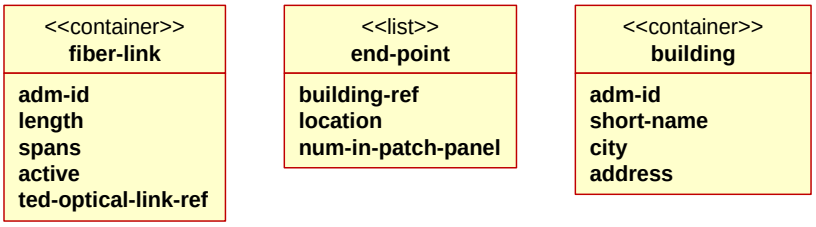


Fig. 9-8 Main Attributes of Fiber-related and Building-related Classes.

Some illustrative queries are briefly defined below, where filtering criteria are only for illustrative purposes.

- *GET /inventory?filter=/equipment[ted-optical-node='10.10.0.4']*. Gets the equipment represented by a node in the TED. Note that the query without any filtering retrieves all the nodes, which can be afterwards correlated with the TED.
- *GET /inventory?filter=/line-card[model='LongRange' and installed='false']*. Selects all spare line-cards of a given card model. Compatible models are specified in each equipment slot.
- *GET /inventory?filter=/building[adm-id='4']*. Selects the building with a given identifier. Building's data includes geographical coordinates that can be used to compute terrestrial distances.
- *GET /inventory?filter=/fiber[active='false']*. Selects all unused fiber links.

The proposed architecture and YANG model are experimentally assessed in a real scenario in the next section.

9.3 Experimental Assessment

In this section, we focus on the experimental validation of the proposed architecture. Experiments have been carried out in our SYNERGY test-bed.

PLATON, the active PCE, and the inventory module were implemented in C++ for Linux. The TED in the PCE and the inventory database were populated beforehand. Next, TEDs in the PCE and in PLATON were synchronized using BGP-LS. We implemented the network and reproduced the scenario presented in Fig. 9-2.

The capture in Fig. 9-9 shows the messages exchanged, while Fig. 9-10 presents their details. The NMS starts the network planning workflow by sending a HTTP message (1) to PLATON specifying the required threshold together with the objective function (`of`=61 in this example). The incremental planning algorithm in PLATON creates a graph with the topology in the TED augmented with data in the inventory. To that end, it issues RESTCONF queries (2-5) towards the inventory module.

	No.	Time	Src Addr	Dst Addr	Proto	Len	Info
①	4	0.0002	NMS	PLATON	HTTP	240	GET /platon?of=61&threshold=0.01 HTTP/1.1
②	9	0.0058	PLATON	INVENTORY	HTTP	137	GET /inventory?filter=/equipment HTTP/1.0
	11	0.0202	INVENTORY	PLATON	HTTP/XML	9445	HTTP/1.0 200 OK
③	19	0.0244	PLATON	INVENTORY	HTTP	182	GET /inventory?filter=/line-card[model='LongRange'%20...
	21	0.0358	INVENTORY	PLATON	HTTP/XML	713	HTTP/1.0 200 OK and%20installed='false'] HTTP/1.0
④	29	0.0384	PLATON	INVENTORY	HTTP	136	GET /inventory?filter=/building HTTP/1.0
	31	0.0433	INVENTORY	PLATON	HTTP/XML	1931	HTTP/1.0 200 OK
⑤	39	0.0454	PLATON	INVENTORY	HTTP	149	GET /inventory?filter=/fiber[active='false'] HTTP/1.0
	41	0.0534	INVENTORY	PLATON	HTTP/XML	1302	HTTP/1.0 200 OK
⑦	46	0.1652	PLATON	NMS	HTTP	772	HTTP/1.1 200 OK (application/json)

Fig. 9-9 Messages Exchanged.

Message (2) is used to retrieve all installed equipment that are correlated with the nodes in the TED. Equipment data includes vendor and model, building id and information regarding each slot, including the cards already installed and compatible card models. Using the compatible card models of each equipment, message (3) retrieves those line-cards available in warehouses (not yet installed). Message (4) retrieves building data used to compute terrestrial distances. Finally, message (5) retrieves unused fiber links. Fiber link data includes building end-points that are used to identify the nodes that can be connected using that fiber.

Once a solution is found, PLATON sends message (7) with the list of actions to be performed back to the NMS (installing line-cards, connecting fiber links to interfaces, and activating fiber links). In the example, the solution involves installing line-card with id=28 in card slot 8 in the equipment represented by node X4, and line-card with id=16 in card slot 7 in X6. Line-card 28 interface 0 must be connected to fiber link 31 and interface 1 to fiber link 30 and line-card 16 interface 0 to fiber link 30 and interface 1 to fiber link 31. Finally, line-cards 28 and 16 and fiber links 30 and 31 must be activated. The actions require manual intervention and should be redirected to the appropriate engineering teams.

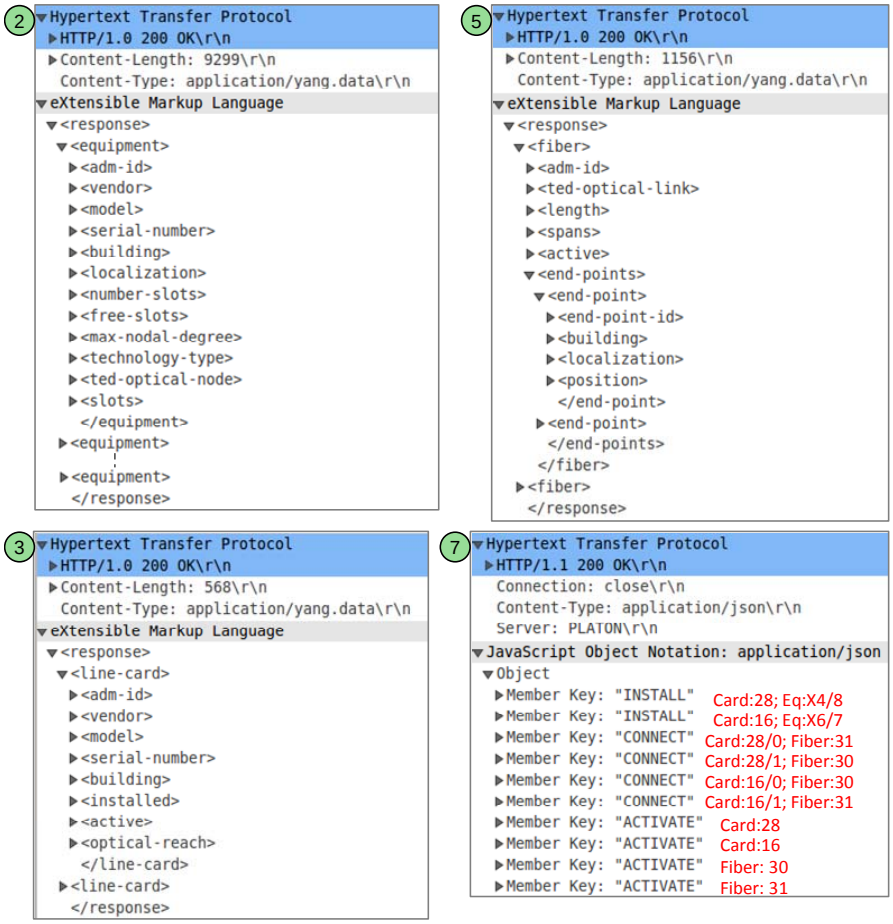


Fig. 9-10 Detail of Messages Exchanged.

9.4 Conclusions

Huge changes in the traffic to be transported by backbone optical networks, not only in volume but also in its distribution and dynamicity due to the introduction of new services, are forecast for the next few years. Hence, efficient planning methods are needed to increment the capacity of optical networks aiming at meeting performance metrics (blocking probability and restorability). By planning the network periodically, e.g., yearly, new capacity can be installed to cope with the forecast traffic for the next period. Nonetheless, periodical planning needs from predictions as exact as possible for the expected traffic volume and distribution, which, although feasible for static traffic scenarios, is unreal when dynamic traffic is considered.

In view of that, planning the network on-demand was proposed in this chapter to add new capacity when performance starts dropping as a result of traffic changes.

To support on-demand network planning, an architecture was proposed; it relates the NMS, the PCE containing the operation databases, a planning tool where algorithms run, and the inventory system. An inventory model that consists of a set of related elements and its correlation with operational databases was presented and data access described. The architecture was experimentally assessed in a distributed test-bed.

Chapter 10

Other Applications

In the last chapters, we studied a number of applications of in-operation and incremental planning. In this chapter, we go further and explore different directions of planning: multi-domain connectivity and data analytics -based VNT reconfiguration.

Regarding the application of in-operation planning to multi-domain scenarios, we consider the problem of per-domain spectrum defragmentation. A broker on top of opaquely-managed optical domains advertising their capabilities is proposed to provision multi-AS connections in multi-operator scenarios. In case of no spectrum continuity, intra-domain spectral defragmentation is performed.

As for data analytics, we propose a big data network manager architecture to support VNT adaptability based on traffic prediction from applying data analytics on the monitored traffic data.

10.1 Per-Domain Defragmentation in Multi-Operator Multi-AS Scenarios

10.1.1 Broker-based Multi-Operator Architecture

Let us assume a multi-operator multi-AS flexgrid optical network, where each AS is managed by an SDN/OpenFlow controller or an ABNO-based architecture. On top of the ASs, a broker coordinates end-to-end multi-AS provisioning (Fig. 10-1).

Each AS advertises an abstracted intra-AS link information to the broker that depends on both, internal AS policies and the specific agreement with the broker. The broker has a global view of the abstracted network topology, including full

information of the inter-AS links and abstracted intra-AS link status gathered from each AS.

In addition, an AS may agree to expose further features to the broker. For example, some ASs may have deployed specific hardware (e.g. wavelength converters/regenerators) and/or implemented optimization algorithms (e.g. spectrum defragmentation algorithms), named as *capabilities*.

To model the underlying data plane, let us assume a graph G(N, E), where N is the set of optical nodes and E is the set of optical links connecting two nodes. Graph G is structured as a set of ASs D. Every AS d consists of three differentiated subset of nodes:

- N_e: subset of edge nodes, end-points of demands;
- N_i: subset of internal AS nodes;
- N_b: subset of border AS nodes. Then, $N = N_e \cup N_i \cup N_b$ with $N_e \cap N_b = \emptyset$.
- Let S be the set of available frequency slices in each optical link.

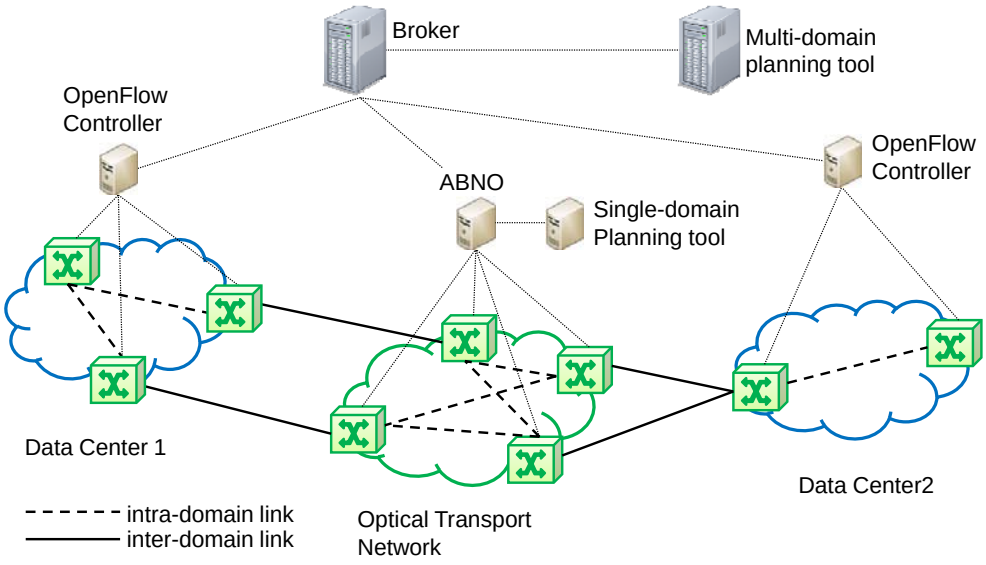


Fig. 10-1 Multi-AS architecture.

Regarding the links, two subsets are considered:

- E_i: subset of inter-AS links, connecting two nodes in N_b belonging to two different ASs;
- E_n: subset of abstracted intra-AS links. Each $e \in E_n$ abstracts connectivity between either a node in N_e and another node in N_b belonging to the request's end ASs, or between two nodes in N_i belonging to transit ASs.

Each link e is represented by a tuple $<a_e, z_e, S_e, c_e>$, where $a_e, z_e \in N_e \cup N_b$ are the end nodes, S_e is the subset of available frequency slices, and c_e is the cost.

Since both, broker and the planning tool will be requested to perform complex computations, each AS is assumed to advertise sets N_b and E_i at start time, and update the set S for each link in E_i to follow updates, independently from path computation requests. In addition, each AS advertises its capabilities (e.g., spectrum defragmentation) (Fig. 10-2a). When a computation is requested, the broker collects intra-AS data (E_n) (Fig. 10-2b), which are advertised to the planning tool in case that in-operation planning is needed (Fig. 10-2c).

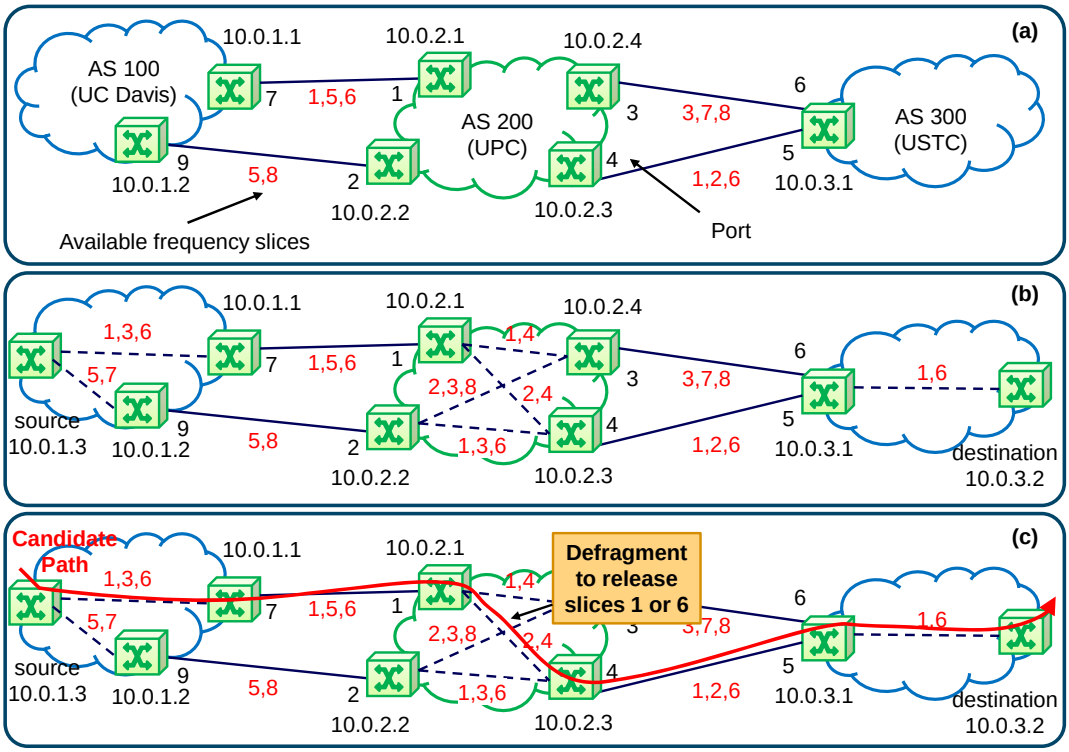


Fig. 10-2 Example of path computation.

10.1.2 Proposed Workflow

Fig. 10-3 illustrates the proposed provisioning workflow, which is divided into three main phases:

i) the *Domain Advertisement* phase is initiated when the broker first connects to the ASs controllers. The broker collects the inter-AS information, along with the AS's capabilities;

ii) the *Path Computation* phase is triggered by the arrival of a new inter-AS path computation request to an SDN controller. Next, the SDN

controller forwards the request to the broker (step 5). Afterwards, the broker gets the intra-AS connectivity (steps 6 and 7). Then, the broker makes a path computation request to the planning tool, adding in the request message the new topology information just obtained (step 8). If the planning tool finds a feasible solution it responds to the broker the multi-AS path to be set-up. Otherwise, it responds a no-path and proposes a solution using one or more capabilities (step 9). In the latter case, the broker tests if the capabilities are still available (steps 10 and 11). If the capabilities are successfully tested, the broker sends a new path computation request to the planning tool allowing the possibility of using the just tested capabilities during the computation (step 12). Eventually, the planning tool responds with the multi-AS path to be set-upped and the list of capabilities to be used (step 13);

iii) in the *Path Set-up* phase, the broker, following the solution proposed by the planning tool, instructs the SDN controllers to signal the intra-AS path and configure the borders routers (steps 14 and 15). Once all the SDN controllers finish its local set-up, the broker informs the SDN controller which made the original request that the inter-AS path is signaled.

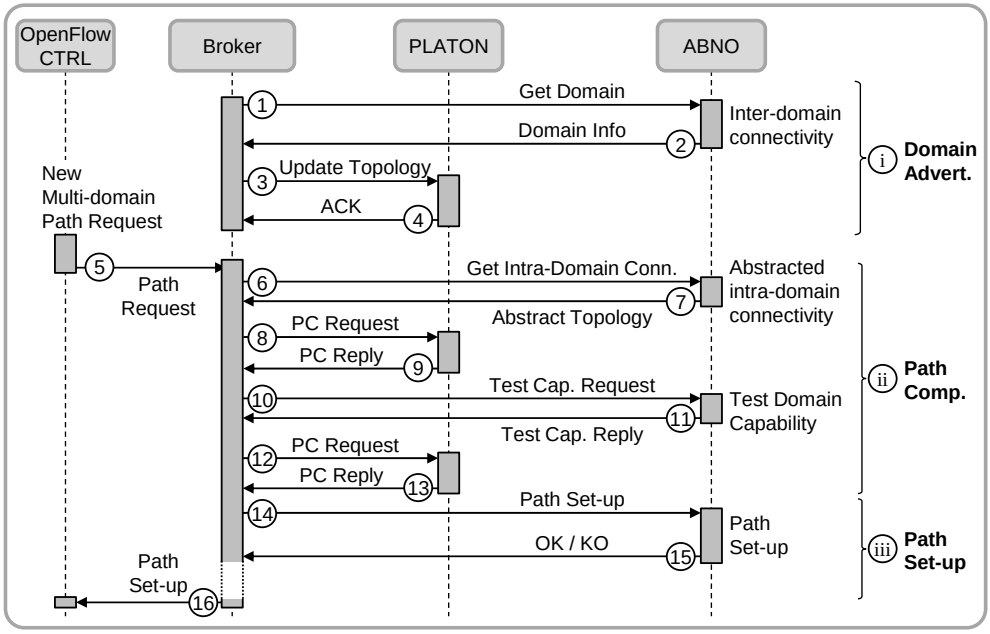


Fig. 10-3 Proposed workflow for per-domain defragmentation.

10.1.3 Experimental Assessment

The experimental validation was carried out on a distributed field trial set-up, shown in Fig. 10-4, connecting premises in UC Davis (Davis, California), USTC

(Hefei, China), and UPC. PLATON, based on the iONE module presented in Chapter 4, was deployed as a multi-domain planning tool. In addition, the ABNO architecture based also on the iONE module, was in charge of one of the domains (AS200). Specifically, the SPRING algorithm presented in Chapter 5 was deployed in the bPCE.

Regarding the management plane, to enable the broker to orchestrate the experiment, we developed an HTTP REST API at the broker, which is implemented by the SDN controllers and PLATON. For each API function a specific XML was devised.

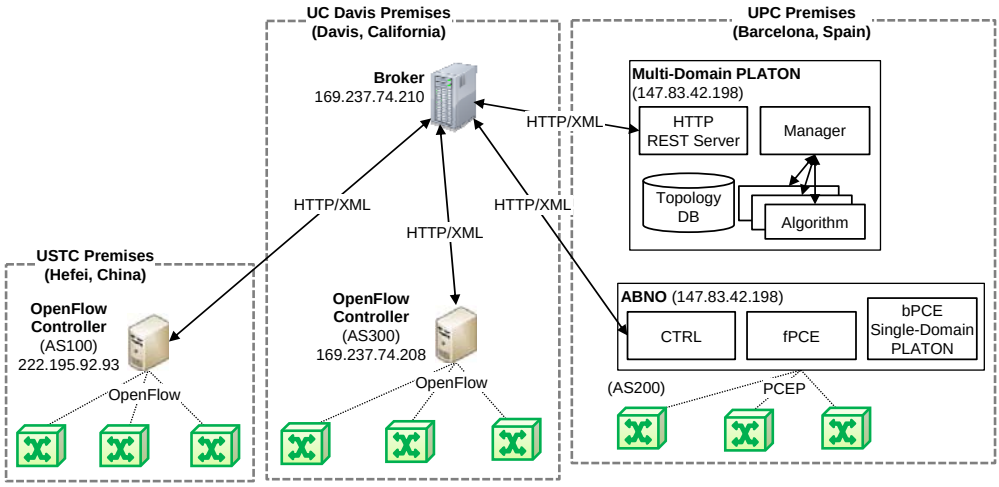


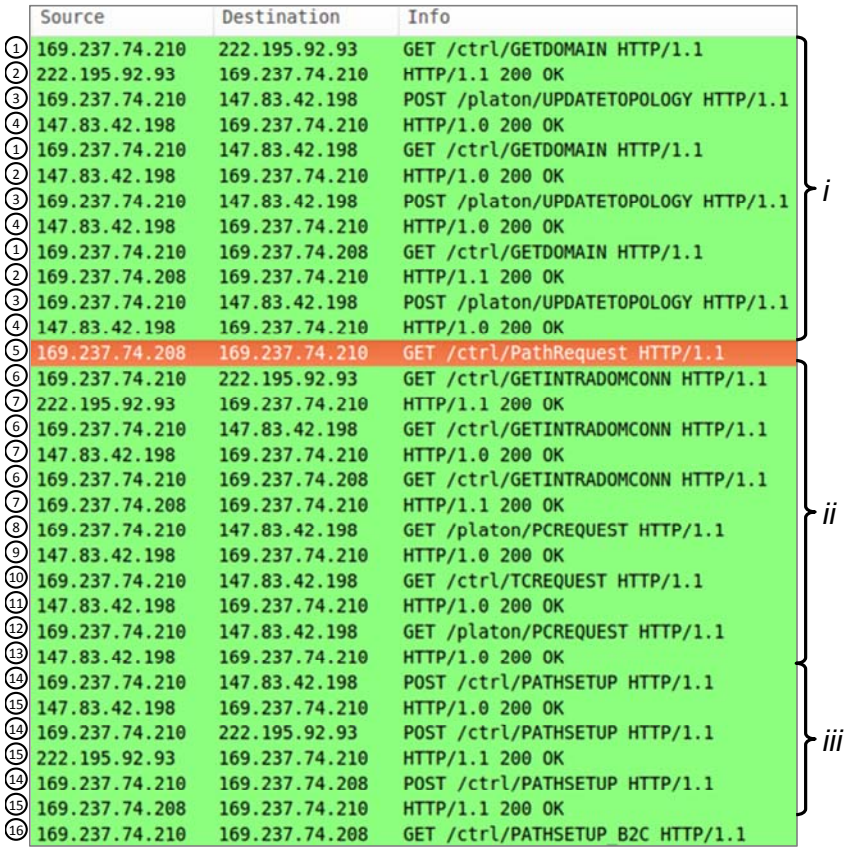
Fig. 10-4 Distributed set-up.

Fig. 10-5 shows the exchanged messages from a broker point of view. For the sake of clarity the numbers of the messages in the figures are in correspondence with each other. The XML files used as input/output parameters for the API functions are shown in Fig. 10-6 and Fig. 10-7.

The workflow starts when the broker connects to all three SDN controllers and populates its topology. Every time a new topology is obtained, a copy is sent to PLATON, in order to maintain broker and PLATON databases synchronized (steps 1-4). In the event of a path computation request received from a SDN controller (step 5), the Broker collects abstracted intra-AS connectivity and AS capabilities from every controller (steps 6-7). Afterwards, the broker sends a path computation request to PLATON (step 8). In the path computation message, the broker also includes the new topology information just learned.

PLATON, first updates its database with the new topology information contained in the request message, and then performs the path computation. Due to our set up, no solution is found. Consequently, a NoPath reply is sent to the broker. Within the reply message PLATON suggests that if defragmentation is used in the UPC AS, a solution can be found (step 9). Then, the broker accepts PLATON suggestion

and tests the defragmentation capability in the UPC AS (step 10). As a result of the test, the UPC AS responds with an OK message (step 11).



	Source	Destination	Info	
1	169.237.74.210	222.195.92.93	GET /ctrl/GETDOMAIN HTTP/1.1	
2	222.195.92.93	169.237.74.210	HTTP/1.1 200 OK	
3	169.237.74.210	147.83.42.198	POST /platon/UPDATETOPOLOGY HTTP/1.1	
4	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
1	169.237.74.210	147.83.42.198	GET /ctrl/GETDOMAIN HTTP/1.1	
2	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
3	169.237.74.210	147.83.42.198	POST /platon/UPDATETOPOLOGY HTTP/1.1	i
4	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
1	169.237.74.210	169.237.74.208	GET /ctrl/GETDOMAIN HTTP/1.1	
2	169.237.74.208	169.237.74.210	HTTP/1.1 200 OK	
3	169.237.74.210	147.83.42.198	POST /platon/UPDATETOPOLOGY HTTP/1.1	
4	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
5	169.237.74.208	169.237.74.210	GET /ctrl/PathRequest HTTP/1.1	
6	169.237.74.210	222.195.92.93	GET /ctrl/GETINTRADOMCONN HTTP/1.1	
7	222.195.92.93	169.237.74.210	HTTP/1.1 200 OK	
6	169.237.74.210	147.83.42.198	GET /ctrl/GETINTRADOMCONN HTTP/1.1	
7	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
6	169.237.74.210	169.237.74.208	GET /ctrl/GETINTRADOMCONN HTTP/1.1	
7	169.237.74.208	169.237.74.210	HTTP/1.1 200 OK	ii
8	169.237.74.210	147.83.42.198	GET /platon/PCREQUEST HTTP/1.1	
9	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
10	169.237.74.210	147.83.42.198	GET /ctrl/TCREQUEST HTTP/1.1	
11	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
12	169.237.74.210	147.83.42.198	GET /platon/PCREQUEST HTTP/1.1	
13	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
14	169.237.74.210	147.83.42.198	POST /ctrl/PATHSETUP HTTP/1.1	
15	147.83.42.198	169.237.74.210	HTTP/1.0 200 OK	
14	169.237.74.210	222.195.92.93	POST /ctrl/PATHSETUP HTTP/1.1	
15	222.195.92.93	169.237.74.210	HTTP/1.1 200 OK	iii
14	169.237.74.210	169.237.74.208	POST /ctrl/PATHSETUP HTTP/1.1	
15	169.237.74.208	169.237.74.210	HTTP/1.1 200 OK	
16	169.237.74.210	169.237.74.208	GET /ctrl/PATHSETUP_B2C HTTP/1.1	

Fig. 10-5 Messages Exchange at the broker.

Immediately after, the broker resends the path computation request to PLATON, but this time informing that the defragmentation capability can be used (step 12). Now PLATON finds a solution, and sends it to the broker. The solution in the path computation reply contains the routing and spectrum allocation and the capability to be applied (step 13).

Finally, the Broker creates the set of configurations to be forwarded to the corresponding SDN controllers (step 14). Eventually, when every controller confirms that the configuration has been set-up (step 15), the broker informs the requester SDN controller that the multi-AS path is signaled (step 16).

Fig. 10-6 XML files for steps 7, 11 and 13.

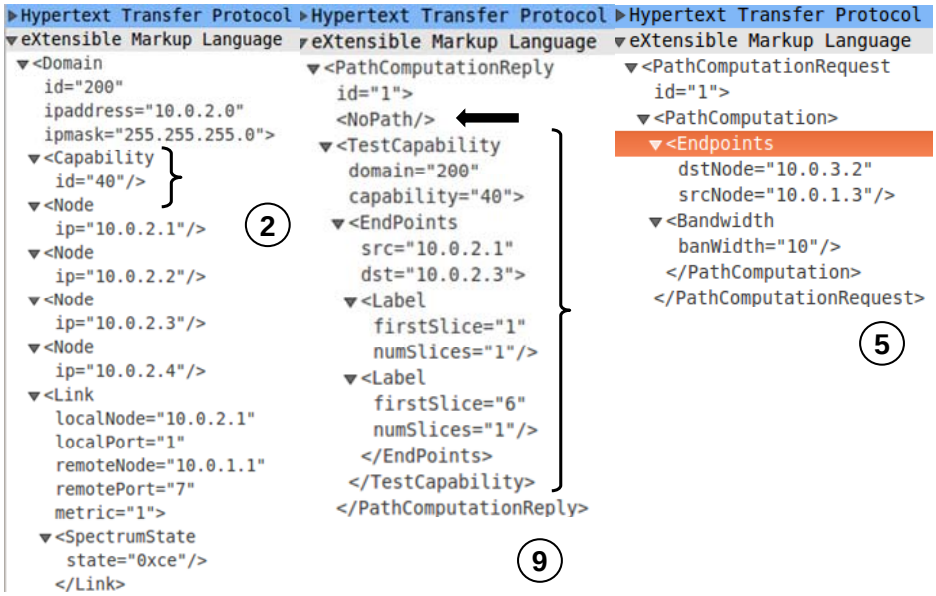


Fig. 10-7 XML files for steps 2, 5, and 9.

10.2 Data Analytics -based VNT Reconfiguration

One of the motivations of this PhD thesis is to deal with the traffic increment forecast for the years to come. That traffic increment entails that static packet network topologies have been largely overprovisioned thus, increasing network TCO. In view of that, network operators are looking for more efficient architectures able to reduce TCO, while providing the required GoS. To that end, VNTs need to be dynamically adapted not only to variations in traffic volume, but also to changes in the direction of the traffic.

To automate VNT adaptability, traffic needs to be monitored in the packet nodes and counters be accessible by the OAM Handler. In particular, the disaggregated traffic volume forwarded by each packet node to every other destination node should be available. In addition, notification can be also triggered when the used vlink capacity reach some configured threshold (e.g., 90%).

VNT adaptation requires from powerful architectures and algorithms to analyze large amounts of monitored traffic data, so as to anticipate, when possible, to traffic changes targeting at optimizing resource utilization. To that end, in this section, we propose a big data network manager architecture to support VNT adaptability based on traffic prediction from applying data analytics on the monitored traffic data.

A threshold triggered VNT reconfiguration is shown in Fig. 10-8 for a seven-node VNT. Fig. 10-8a presents monitored traffic data captured during the last two hours in node 6, where traffic from that node to every other node in the VNT (6->*N*), from node 6 to node 7 (6->7), and from node 6 to every other node except to node 7 (6->*N*\{7}) is plotted. Fig. 10-8b shows the initial VNT where every vlink is supported by a 100 Gb/s LSC LSP in the underlying optical layer; the route of PSC LSP 6-7 is also shown. A 90% threshold is configured and, in the event of threshold violation, the OAM Handler might request the ABNO controller increasing the capacity of some vlinks by establishing a parallel LSC LSP. In our example, two threshold violations for vlinks 1-6 and 1-7 are received, so the VNT is updated (Fig. 10-8c). It is worth noting that no PSC LSP is rerouted.

Threshold triggered VNT reconfiguration is able to adapt the VNT to traffic changes at the cost of using an increased number of transponders in the nodes. In the example in Fig. 10-8, four optical transponders are used for the new LSC LSPs 1-6 and 1-7. However, analyzing the plots in Fig. 10-8a we realize that traffic 6->7 is responsible for the registered traffic increment. To facilitate data analysis, monitored traffic data received in the OAM Handler is aggregated into four values (i.e. min, max, average, and last) (see algorithm in Table 10-1) so as to model the traffic behavior during the last hour (Fig. 10-9a); the algorithm follows a map-reduce scheme, where the map phase de-serializes JSON messages and the reduce phase merges sample pairs.

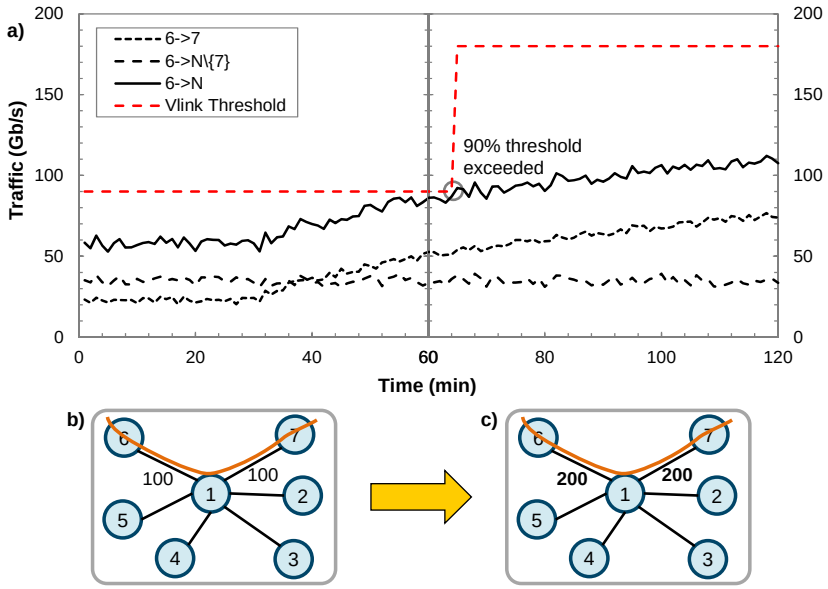


Fig. 10-8 Threshold triggered VNT reconfiguration.

Table 10-1 Aggregation Algorithm

```
INPUT T = {Text}
OUTPUT q = <t, min, max, avg, last>
define aggregator(in p1, p2; out q1)
  q1.min ← min(p1.min, p2.min)
  q1.max ← max(p1.max, p2.max)
  q1.avg ← p1.avg + p2.avg
  q1.last ← getLastAvg(p1, p2)
  q1.t ← max(p1.t, p2.t)
  return q1

P = {<t, min, max, avg>} ← T.map(jsonDeserialize)
q ← P.reduce(aggregator)
q.avg ← q.avg / |P|
return q
```

From the current VNT (Fig. 10-9b) and analyzing modelled traffic data, it can be seen that maximum and last traffics got similar values in t=60, being both over the average one. If traffic volume is large enough, vlink 6-7 can be created by establishing a LSC LSP (Fig. 10-9c) and traffic 6->7 be non-disruptively rerouted (Fig. 10-9d). Note that this solution needs from just two transponders in contrast to the threshold triggered reconfiguration. If the capacity of some vlinks can be reduced after rerouting, some of the supporting LSC LSPs will be torn-down.

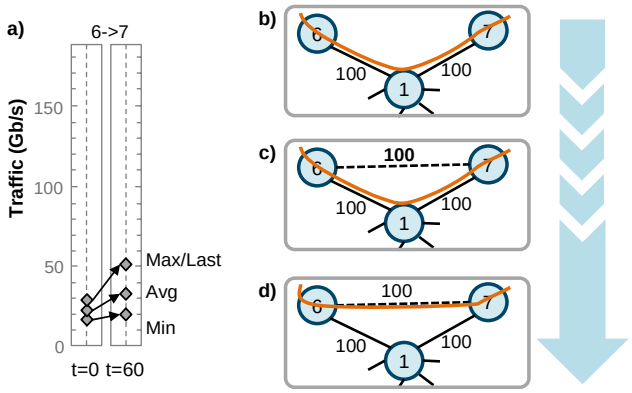


Fig. 10-9 Data analytics -based VNT reconfiguration.

10.2.1 Big Data Network Manager: Architecture and Workflow

In the proposed architecture in Fig. 10-10, monitored traffic data is sent periodically to a big data repository consisting of a distributed database and a data collector. Periodically, e.g. every minute, the OAM Handler retrieves aggregated monitored data, which is stored into a big data system ready to be analyzed.

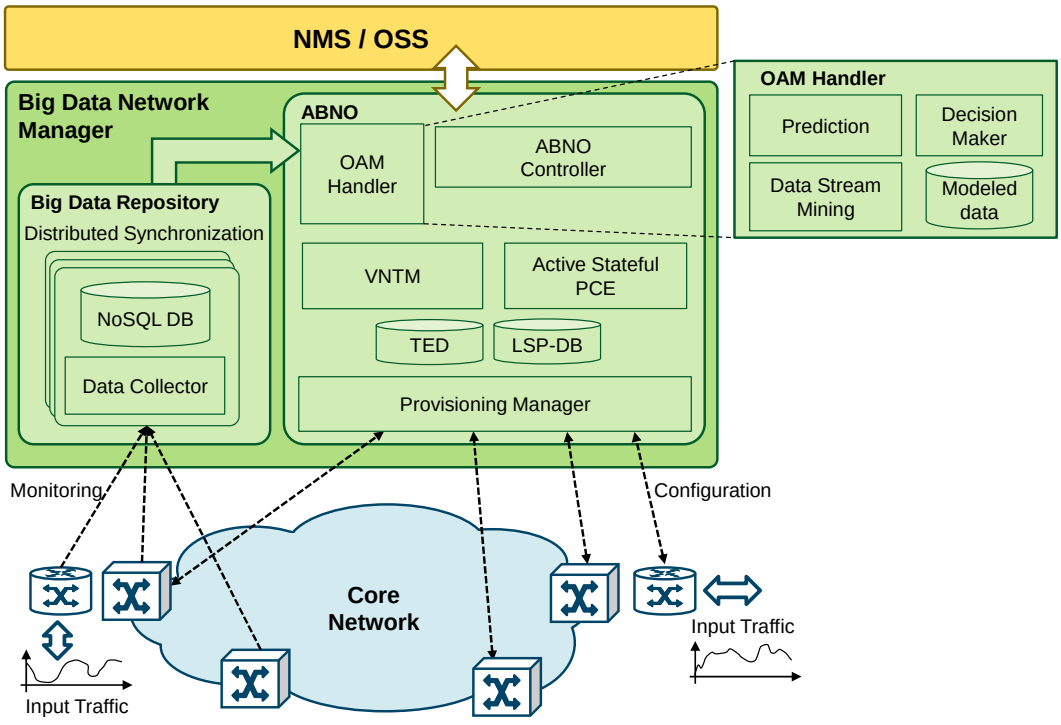


Fig. 10-10 Big data network manager architecture.

The big data OAM Handler applies data stream mining techniques on the received data and periodically (e.g. every hour) transforms monitored data into modelled

data. Modelled data is used by a *prediction* module, running machine learning algorithms to anticipate next period traffic conditions (labelled as 1 in Fig. 10-11). Based on the predicted traffic, a *decision maker* module decides whether the VNT should be updated. In case of VNT reconfiguration, the VNTM is in charge of computing the new VNT. To that end, the OAM Handler issues a request to the ABNO controller that includes the predicted traffic together with some other parameters to facilitate VNTM computation (2) and the ABNO controller initiates the workflow forwarding a request to the VNTM (3).

The VNTM computes the optimal VNT with the predicted traffic matrix received from the OAM Handler (4). Continuing with our seven-node VNT example, let us assume that the new VNT consists on adding the new virtual link 6-7 and reducing the capacity of some other vlinks. The solution is first notified to the NMS (5) and then, its implementation is divided into a sequence to avoid traffic disruption as anticipated above: firstly, LSC LSP 6-7 is created (6) and the new vlink is advertised (7); next, PSC LSPs are rerouted (8) (a make-before-brake strategy to avoid disruption can be implemented) and unused capacity in vlinks 1-3 and 1-4 removed by tearing down the underlying LSC LSPs (9); new vlinks' capacity is advertised (10). Upon VNT reconfiguration completion, VNTM replies the ABNO controller (11), which eventually replies the OAM Handler (12).

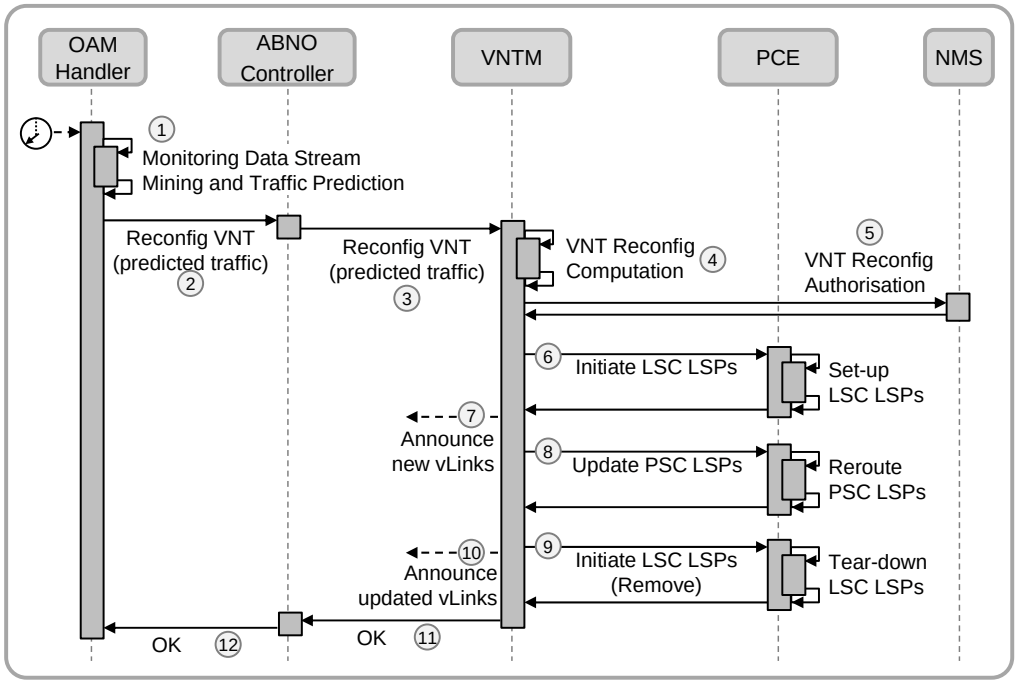


Fig. 10-11 Proposed workflow.

10.2.2 Experimental Assessment

Experiments were carried out on the UPC's SYNERGY test-bed. Apache Cassandra database [Cassandra] was used as a big data repository and the data collector module was implemented to offer an UDP-based interface to the monitors, storing the received data in Cassandra. Apache Spark [Spark] was used to implement data stream mining and machine learning techniques. Finally, ABNO modules in Fig. 10-10 were implemented as iONE modules (Chapter 4). A HTTP REST API interface was implemented between the OAM Handler and the ABNO controller and from it to VNTM, so as to convey the predicted traffic matrix. PCEP was used between VNTM, PCE, and PM. Finally, VNTM is in charge of advertising topological changes in the VNT, including vlink creation and releasing, as well as updating vlink capacity changes.

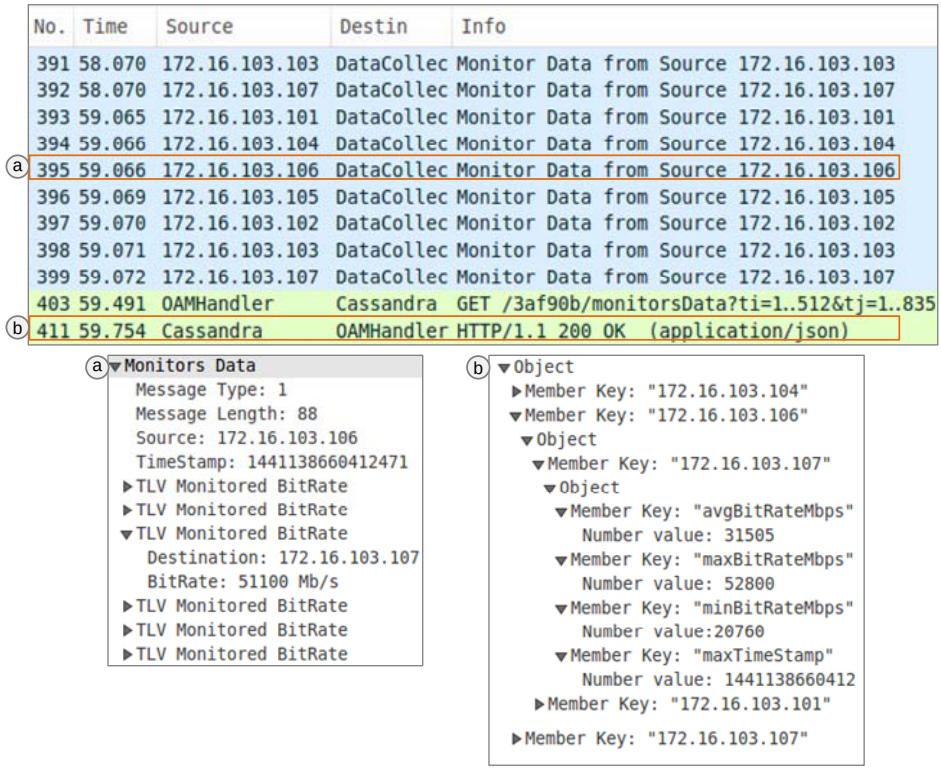


Fig. 10-12 Exchanged messages for monitored traffic Collection.

Fig. 10-12 illustrates monitored traffic data periodically being send by the packet nodes to the data collector, as well as the request that the OAM Handler issues to Cassandra's REST API to collect monitored data. UDP monitoring messages contain, among others, the source node and the timestamp of the sample, and for each aggregated flow leaving the node to a destination, its destination node and bitrate. After selecting and aggregating monitored data between the selected times

t_i and t_j, Cassandra replies with a JSON-encoded matrix specifying for each pair of source-destination the average, maximum, and minimum bitrate.

	No.	Time	Source	Destin	Protocol	Info
2	465	*REF*	OAMHandler	ABNOCtrl	HTTP/XML	POST /ctrl/VNTReconfig HTTP/1.0
3	468	0.000	ABNOCtrl	VNTManager	HTTP/XML	POST /vntm/VNTReconfig HTTP/1.0
5	471	0.028	VNTManager	NMS	HTTP/XML	POST /nms/VNTReconfig HTTP/1.0
	475	0.030	NMS	VNTManager	HTTP/XML	HTTP/1.1 200 OK
6	478	0.030	VNTManager	PCE	PCEP	Path Computation LSP Initiate (PCInitiate)
	483	0.076	PCE	VNTManager	PCEP	Path Computation LSP State Report (PCRpt)
7	484	0.077	VNTManager	PCE	BGP	UPDATE Message
	486	0.082	VNTManager	PCE	BGP	UPDATE Message
8	495	0.097	VNTManager	PCE	PCEP	Path Computation LSP Update Request (PCUpd)
	497	0.104	PCE	VNTManager	PCEP	Path Computation LSP State Report (PCRpt)
9	498	0.104	VNTManager	PCE	PCEP	Path Computation LSP Initiate (PCInitiate)
	499	0.104	VNTManager	PCE	PCEP	Path Computation LSP Initiate (PCInitiate)
10	500	0.104	VNTManager	PCE	BGP	UPDATE Message
	503	0.109	VNTManager	PCE	BGP	UPDATE Message
9	509	0.167	PCE	VNTManager	PCEP	Path Computation LSP State Report (PCRpt)
	511	0.216	PCE	VNTManager	PCEP	Path Computation LSP State Report (PCRpt)
11	513	0.217	VNTManager	ABNOCtrl	HTTP/XML	HTTP/1.0 200 OK
12	515	0.217	ABNOCtrl	OAMHandler	HTTP/XML	HTTP/1.0 200 OK

Fig. 10-13 Exchanged messages for VNT reconfiguration.

```
▼<VNTReconfig>
 ▼<Matrix
    name="bitRateMbps">
  ▶<Data
  ▶<Data
  ▶<Data
  ▶<Data
  ▶<Data
  ▼<Data
     src="172.16.103.106"
     dst_172_16_103_101="17650"
     dst_172_16_103_102="7600"
     dst_172_16_103_103="3140"
     dst_172_16_103_104="9680"
     dst_172_16_103_105="4060"
     dst_172_16_103_107="72100"/>
  ▶<Data
    </Matrix>
 ▶<Params>
  </VNTReconfig>
```

Fig. 10-14 Message (2) details.

Fig. 10-13 shows the meaningful messages exchanged between ABNO modules. For the sake of clarity, messages are identified following the workflow in Fig. 10-11. The OAM handler sends a REST API request to the ABNO controller (message 2) containing the predicted traffic matrix for the next period. The details of that message are presented in Fig. 10-14. After receiving the predicted traffic matrix, the VNT computes the optimal VNT and issues requests to the PCE to implement

the LSC LSPs supporting the new vlinks, reroute the selected PSC LSPs, and tear down unused LSC LSPs. In addition, VNT changes are advertised to the rest of ABNO modules. The total process took 217ms, from the instant the OAM handler triggered the workflow.

10.3 Conclusions

In this chapter, we complemented the applications of in-operation planning with per-domain defragmentation in multi-domain multi-AS scenarios and with our proposal of a big data analytics OAM handler to support VNT adaptability based on traffic prediction.

Firstly, a broker-based multi-operator architecture interoperating with a planning tool, where complex computations are delegated, has been proposed to provision multi-AS connections. The planning tool had a partial view of the multi-domain network topology containing: *i*) the set of border nodes in each domain, *ii*) a set of abstract links connecting these border nodes and *iii*) the capabilities, such as per-domain spectrum defragmentation, of each domain. The broker was responsible for path provisioning request; however, when no spectrum continuity is available for setting-up an end-to-end connection, the broker delegated the computation to the planning tool, since it involves a complex computation. The planning tool verified the feasibility of applying per-domain spectrum defragmentation in each domain to leverage resources for the new connection, using the broker as a gateway to interrogate the domain controllers. The feasibility of the aforementioned procedure has been studied implementing the proposed workflow and experimentally assessing it in a distributed test-bed.

Then, we focused on VNT adaptability. Packet nodes monitor incoming traffic and send monitoring data to a big data repository, based on Cassandra. Monitoring data is collected by the OAM Handler and locally stored. Periodically, e.g. every hour, collected monitoring data is transformed into modelled data and used to predict next period traffic applying machine learning techniques. A workflow has been proposed, where the VNTM module is in charge of adapting the VNT to future conditions. To that end, the VNTM finds the optimal VNT based on the predicted traffic computed by the OAM handler.

Chapter 11

Closing Discussion

11.1 Main Contributions

The main contributions of this thesis are:

- In Chapter 4, an architecture supporting in-operation planning was assessed; a bPCE, responsible for running intensive computations, such as network re-optimizations, was validated. The coordination between fPCE and bPCE was assessed in terms of computation delegation and database synchronization. The iONE module architecture used to deploy ABNO components was presented and the configuration and interoperation of a set of ABNO components was illustrated with a use case. (G1)
- An application for hitless spectrum defragmentation using the novel spectrum shifting technique, named as push-pull, was proposed in Chapter 5. The approach was experimentally assessed in a distributed test-bed. (G2.1)
- Another in-operation planning mechanism was investigated in Chapter 6; where multiple paths were used to re-optimize the network after a failure was repaired. The approach was experimentally validated in a distributed test-bed. (G2.2)
- Chapter 7 was devoted to study the problem of establishing multicast point-to-point connectivity; heuristic algorithms were proposed for two different approaches following single-layer and multi-layer schemes, respectively, to serve the multicast connectivity services. Both approaches were compared and experimentally assessed in a distributed test-bed. (G2.3)

- A new architecture for hierarchical content distribution in the telecom cloud was proposed and experimentally assessed in Chapter 8. This architecture enabled the placement of datacenters closer to the end-users thus, reducing the service latency and improving the GoS. (G2.4)
- In Chapter 9, the architecture presented in Chapter 4 was extended to explore an inventory-aware mechanism to periodically plan the network. The approach was experimentally assessed in a distributed test-bed. (G2.5)
- An application for brokered multi-domain connectivity services with per-domain spectrum defragmentation was investigated and experimentally assessed in Chapter 10. A planning tool connected to the services broker was in charge of computing the multi-domain paths and deciding where to perform per-domain spectrum defragmentation; per-domain planning tools were in charge of running intra-domain spectrum defragmentation computation. (G2.6)
- Finally, the last contribution of this thesis consisted in extending the in-operation planning architecture presented in Chapter 4 with a big-data OAM handler for data analytics-based traffic monitoring; the architecture was experimentally validated in a distributed test-bed. (G2.7)

11.2 List of Publications

11.2.1 Publications in Journals

[Gi16-1] **Ll. Gifre**, M. Tornatore, L. M. Contreras, B. Mukherjee and L. Velasco, "ABNO-driven Content Distribution in the Telecom Cloud," accepted in Elsevier Optical Switching and Networking, 2016. DOI: 10.1016/j.osn.2015.07.006.

[Go16-1] O. González de Dios, R. Casellas, F. Paolucci, A. Napoli, **Ll. Gifre**, A. Dupas, E. Hugues-Salas, R. Morro, S. Belotti, G. Meloni, T Rahman, V. López, R. Martínez, F. Fresi, M. Bohn, S. Yan, L. Velasco, P. Layec and J.P. Fernandez-Palacios, "Experimental Demonstration of Multi-vendor and Multi-domain Elastic Optical Network with data and control interoperability over a Pan-European Test-bed," IEEE/OSA Journal of Lightwave Technology, vol. 34, pp. 1610-1617, 2016.

[Ve16-1] L. Velasco, F. Morales, **Ll. Gifre**, A. Castro, O. González de Dios and M. Ruiz, "On-demand Incremental Capacity Planning in Optical Transport Networks," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 8, pp. 11-22, 2016.

[Gi15-1] **Ll. Gifre**, F. Paolucci, O. Gonzalez de Dios, L. Velasco, L.M. Contreras, F. Cugini, P. Castoldi and V. López, "Experimental Assessment of ABNO-driven Multicast Connectivity in Flexgrid Networks," (Invited Paper) IEEE/OSA Journal of Lightwave Technology (JLT), vol. 33, pp. 1549-1556, 2015.

[Gi15-2] **Ll. Gifre**, F. Paolucci, L. Velasco, A. Aguado, F. Cugini, P. Castoldi and V. López, "First Experimental Assessment of ABNO-driven In-Operation Flexgrid Network Re-Optimization," (Invited Paper) IEEE/OSA Journal of Lightwave Technology (JLT), vol. 33, pp. 618-624, 2015.

[Gi14-2] **Ll. Gifre**, F. Paolucci, A. Aguado, R. Casellas, A. Castro, F. Cugini, P. Castoldi, L. Velasco and V. López, "Experimental Assessment of In-Operation Spectrum Defragmentation," Springer Photonic Network Communications (PNET), vol. 27, pp. 128-140, 2014.

11.2.2 Publications in Conferences

[Gi16-2] **Ll. Gifre**, L. M. Contreras, V. Lopez and L. Velasco, "Big Data Analytics in Support of Virtual Network Topology Adaptability," in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2016.

[Cast15-1] A. Castro, **Ll. Gifre**, C. Chen, J. Yin, Z. Zhu, L. Velasco and S. J. B. Yoo, "Experimental Demonstration of Brokered Orchestration for end-to-end Service Provisioning and Interoperability across Heterogeneous Multi-Operator (Multi-AS) Optical Networks," in Proc. European Conference on Optical Communication (ECOC), 2015.

[Gi15-3] **Ll. Gifre**, R. Martínez, R. Casellas, R. Vilalta, R. Muñoz and L. Velasco, "Modulation Format-aware Re-Optimization in Flexgrid Optical Networks: Concept and Experimental Assessment," in Proc. European Conference on Optical Communication (ECOC), 2015.

[Gi15-4] **Ll. Gifre**, M. Ruiz, A. Asensio and L. Velasco, "Comparing Single Layer and Multilayer Approaches to Serve Multicast Requests on Flexgrid Networks," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2015.

[Gi15-5] **Ll. Gifre**, N. Navarro, A. Asensio, M. Ruiz and L. Velasco, "iONE: An Environment for Experimentally Assessing In-Operation Network Planning Algorithms," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2015.

[Go15-1] O. Gonzalez de Dios, R. Casellas, F. Paolucci, A. Napoli, **Ll. Gifre**, S. Annoni, S. Belotti, U. Feiste, D. Rafique, M. Bohn, S. Bigo, A. Dupas, E. Dutisseuil, F. Fresi, B. Guo, E. Hugues, P. Layec, V. López, G. Meloni, S. Misto, R. Morro, T. Rahman, G. Khanna, R. Martinez, R. Vilalta, F. Cugini, L. Potí, A. D'Errico, R. Muñoz, Y. Shu, S. Yan, Y. Yan, G. Zervas, R. Nejabati, D. Simeonidou, L. Velasco and J. Fernández-Palacios, "First Demonstration of Multi-vendor and Multi-domain EON with S-BVT and Control Interoperability over Pan-European Testbed," Post Deadline Paper in European Conference on Optical Communication (ECOC), 2015.

[Ve15-1] L. Velasco and **Ll. Gifre**, "iONE: A Workflow-Oriented ABNO Implementation," in Proc. IEEE/OSA Photonics in Switching Conference (PS), 2015.

[Gi14-1] **Ll. Gifre**, F. Paolucci, J. Marhuenda, A. Aguado, L. Velasco, F. Cugini, P. Castoldi, O. Gonzalez de Dios, L.M. Contreras and V. López, "Experimental Assessment of Inter-datacenter Multicast Connectivity for Ethernet services in Flexgrid Networks," in Proc. European Conference on Optical Communications (ECOC), 2014.

[Gi14-3] **Ll. Gifre**, L. Velasco, N. Navarro and G. Junyent, "Experimental Assessment of a High Performance Back-end PCE for Flexgrid Optical Network Re-optimization," in Proc. OSA Optical Fiber Communication Conference (OFC), 2014.

[Gi14-4] **Ll. Gifre**, A. Castro, M. Ruiz, N. Navarro and L. Velasco, "An In-Operation Planning Tool Architecture for Flexgrid Network Re-Optimization," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2014.

[Mar14-1] R. Martínez, **Ll. Gifre**, R. Casellas, L. Velasco, R. Muñoz and R. Vilalta, "Experimental Validation of Active Frontend - Backend Stateful PCE Operations in Flexgrid Optical Network Re-optimization," in Proc. European Conference on Optical Communication (ECOC), 2014.

[Ve14-5] L. Velasco, F. Paolucci, **Ll. Gifre**, A. Aguado, F. Cugini, P. Castoldi and V. López, "First Experimental Demonstration of ABNO-driven In-Operation Flexgrid Network Re-Optimization," in Proc. Post Deadline OSA Optical Fiber Communication Conference (OFC), 2014.

[Gi13-1] **Ll. Gifre**, L. Velasco and N. Navarro, "Architecture of a Specialized Back-end High Performance Computing-based PCE for Flexgrid Networks," in Proc. IEEE International Conference on Transparent Optical Networks (ICTON), 2013.

11.2.3 Book Chapters

[Lo16-1] R. Casellas, A. Giorgetti, **Ll. Gifre**, L. Velasco, V. López, O. González and D. King, "Chapter 11: In-Operation Network Planning," In: V. López and L. Velasco (eds.), "Elastic Optical Networks: Architectures, Technologies, and Control", accepted in Optical Networks book series, Springer, 2016.

11.3 List of Research Projects

11.3.1 European Funded Projects

IDEALIST: "Industry-Driven Elastic and Adaptive Lambda Infrastructure for Service and Transport Networks", Ref: FP7-ICT-2011-8, 2012-2015.

GEANT: "Pan-European research and education network that interconnects Europe's National Research and Education Networks", open call "Research and Experimental Assessment of Control Plane Architectures for In-Operation Flexgrid Network Re-Optimization (**REACTION**)", Ref: 238875, 2013-2015.

11.3.2 Spanish Funded Projects

SYNERGY: "Service-oriented hYbrid optical NEtwork and cloud infrastructuRe featuring high throuGhput and ultra-low latencY", Ref: TEC2014-59995-R, 2015-2017.

11.3.3 Pre-doctoral Funding Scholarship

This thesis has been co-funded by the Barcelona Supercomputing Center (BSC) and UPC, 2012-2016.

11.4 Topics for Further Research

The algorithms and architectures devised in this thesis are being used as a starting point for the PhD thesis of Fernando Morales; his thesis focuses on exploring in-operation planning mechanisms based on forecast traffic.

Appendix A. **iONE Workflow Definition**

In this appendix, a use case on spectrum defragmentation is used to illustrate how the iONE module can be configured to define complete workflows involving several ABNO components, including the ABNO controller, a fPCE, a bPCE, and a provisioning manager (PM) component (see Fig. A-1). Each component is implemented using a single iONE module running in the depicted IP address. A configuration XML file is used to define per-module contained databases as well as the interfaces and the connectivity to other modules.

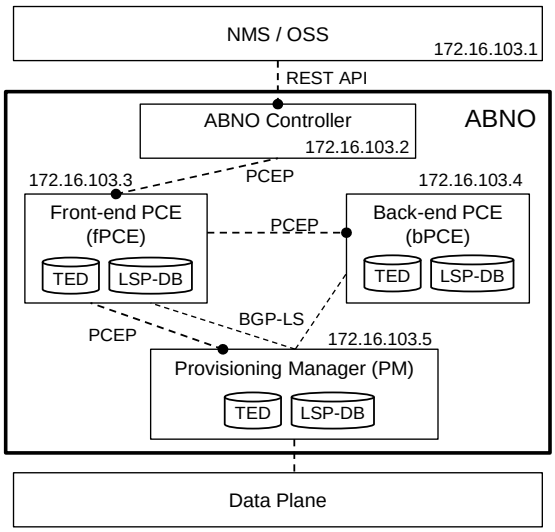


Fig. A-1 Example of iONE set-up.

In our example, the NMS is connected to the ABNO controller through a REST API for requesting connections. Internally, ABNO components use PCEP and BGP-LS interfaces to communicate among them. For instance, the fPCE needs to implement

three PCEP interfaces, one for the ABNO controller, another for offloading computations to the bPCE, and a third one for LSP operations with the PM.

PCEP is used for LSP-DB synchronization by means of PCRpt messages. The PM module generates PCRpt messages towards the fPCE, which updates its local LSP-DB and generates PCRpt messages towards the bPCE module. In addition, BGP-LS is used to synchronize the TED from PM to both fPCE and bPCE.

Specifically for the spectrum defragmentation use case, Fig. A-2a presents the sequence diagram to be implemented. For this use case we assume that the NMS issues a spectrum defragmentation request on a selected optical link based on their specific policies and metrics. Each workflow algorithm has been numbered to facilitate its identification.

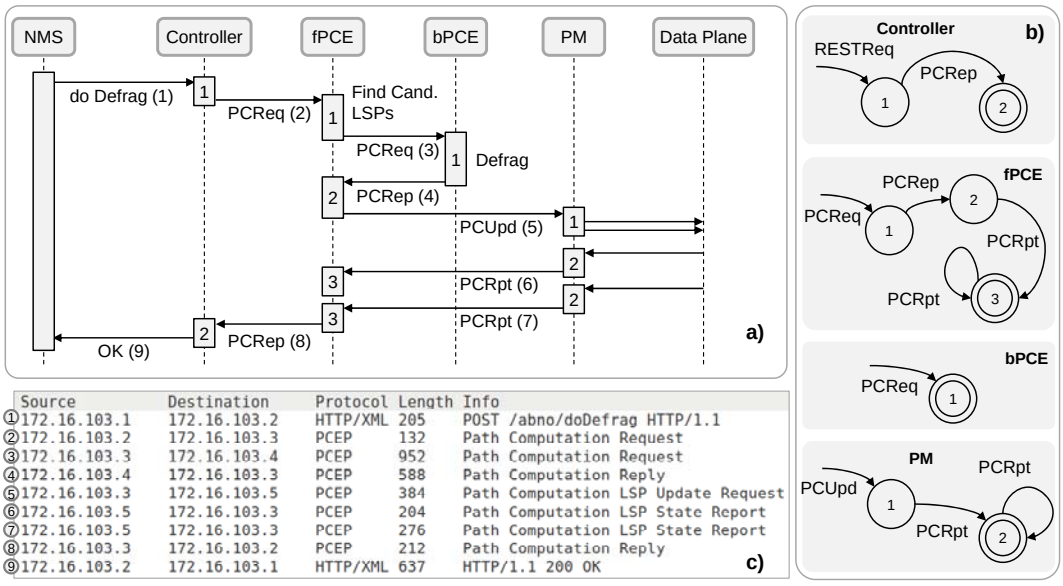


Fig. A-2 Spectrum Defragmentation.

When the ABNO controller receives a spectrum defragmentation request through the REST API interface (message 1), it issues a PCReq message (2) to the fPCE containing an `OF` object defining the re-optimization algorithm to be executed and the specific optical link to be defragmented. The fPCE finds the candidate LSPs, those using the selected optical link, and offloads the defragmentation computation to the bPCE (UPC's in-operation network planning tool named as PLATON [Ve14-1]). The PCReq message (3) contains an `OF` object so the bPCE identifies the algorithm to be executed, runs it and replies the fPCE with the solution in a PCRep message (4). The fPCE processes the received message and requests LSP updates to the PM by means of a PCUpd message (5). The fPCE collects all the replied PCRpt messages (6-7) and eventually replies to the ABNO controller using a PCRep message (8), which in turn replies to the NMS (9).

It is clear that every ABNO component needs to do a number of actions that are triggered by the received messages. As explained above, we model the actions to be done by each module as a FSM. The FSMs for each iONE module are represented in Fig. A-2b, where each state is identified using the same identifier used in Fig. A-2a; error states were omitted for simplicity.

The FSM for the ABNO controller consists of two states; the workflow is triggered by a REST Request (*RESTReq*) message and the algorithm for state (1) is run, and upon the reception of a PCReq message the algorithm for state (2) is run. The FSM for the fPCE is slightly more complex since it involves iteratively executing state 3 until all PCRpt messages are collected. The FSM for the bPCE is the simplest one since its operations (process incoming PCReq message, execute spectrum defragmentation algorithm, and reply solution) are executed strictly sequentially, thus allowing to be encapsulated in a single state.

Finally, Fig. A-2c shows the exchanged REST and PCEP messages captured.

List of Acronyms

ABNO	Application-Based Network Operations
AFRO	After Failure Repair Optimization
AS	Autonomous System
ASO	Application Service Orchestrator
bPCE	Back-end PCE
C2C	Core-To-Core
CAPEX	Capital Expenditures
OPEX	Operation Expenditures
CP	Content Provider
CP-VNT	Content Provider Virtual Network Topology
DB	Database
DC	Datacenter
EON	Elastic Optical Networking
ERO	Explicit Route Object
FCAPS	Fault, Configuration, Accounting/Administration, Performance and Security
FPGA	Field Programmable Gate Array
fPCE	Front-end PCE
GCO	Global Concurrent Optimization
GMPLS	Generalized MPLS
GoS	Grade of Service

GPU	Graphics Processing Unit
HPC	High Performance Computing
HW	Hardware
ILP	Integer Linear Program
IETF	Internet Engineering Task Force
IRO	Include Route Object
IT	Information Technology
JIT	Just-in-time
JSON	JavaScript Object Notation
L0 PCE	Layer 0 PCE
L2	Layer 2
L2 PCE	Layer 2 PCE
LSC	Lambda Switch Capable
LSP	Label Switched Path
LSP-DB	Label Switched Path Database
M2C	Metro-To-Core
MILP	Mixed Integer Linear Program
MP	Multi-path
MP-AFRO	Multi-path After Failure Repair Optimization
MPLS	Multi-Protocol Label Switching
MST	Minimum Spanning Tree
NBI	North-bound Interface
NMS	Network Management System
OAM	Operations, Administration and Management
OD	Origin-Destination
OF	Objective Function
ONF	Optical Networking Foundation
OSS	Operations Support System
OXC	Optical Cross-connect
p2mp	Point-To-MultiPoint
p2p	Point-to-Point

PCC	Path Computation Client
PCE	Path Computation Element
PCEP	PCE Protocol
PCInit	Path Computation Initiate
PCNtf	Path Computation Notification
PCRep	Path Computation Reply
PCReq	Path Computation Request
PCRpt	Path Computation Report
PM	Provisioning Manager
PSC	Packet Switch Capable
REST	Representational State Transfer
RMSA	Routing, Modulation Formats and Spectrum Allocation
RRO	Record Route Object
RSA	Routing and Spectrum Allocation
RSVP-TE	Resource Reservation Protocol – Traffic Engineering
SBI	South-bound Interface
SBVT	Sliceable Bandwidth-Variable Transponder
SD-EON	Software-defined Elastic Optical Networking
SDN	Software Defined Networking
SERO	Secondary ERO
SPRING	Spectrum Shifting
SSS	Spectrum Selective Switches
TCO	Total Cost of Ownership
TE	Traffic Engineering
TED	Traffic Engineering Database
TLV	Type-Length-Value
TM	Topology Module
vlink	Virtual Link
VNT	Virtual Network Topology
VNTM	Virtual Network Topology Manager
WDM	Wavelength Division Multiplexing

WSS	Wavelength Selective Switch
XML	eXtended Markup Language

List of References

[Agr09-1] F. Agraz, L. Velasco, J. Perelló, M. Ruiz, S. Spadaro, G. Junyent, and J. Comellas, "Design and Implementation of a GMPLS-Controlled Grooming-capable Optical Transport Network", IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 1, pp. A258-A269, 2009.

[Agr10-1] F. Agraz, J. Perelló, M. Angelou, S. Azodolmolky, L. Velasco, S. Spadaro, P. Kokkinos, E. Varvarigos, and I. Tomkos, "Experimental Evaluation of Path Restoration for a Centralised Impairment-Aware GMPLS-Controlled All-Optical Network," in Proc. European Conference on Optical Communication (ECOC), 2010.

[Agu14-1] A. Aguado, V. López, J. Marhuenda, O. Gonzalez de Dios, and J. P. Fernández-Palacios, "ABNO: a feasible SDN approach for multi-vendor IP and optical networks," in Proc. OSA Optical Fiber Communications Conference (OFC), 2014.

[Agu15-1] A. Aguado, M. Davis, S. Peng, M. V. Alvarez, V. Lopez, T. Szyrkowiec, A. Autenrieth, R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, R. Martínez, N. Yoshikane, T. Tsuritani, R. Nejabati, and D. Simeonidou, "Dynamic Virtual Network Reconfiguration over SDN Orchestrated Multi-Technology Optical Transport Domains," accepted in IEEE/OSA Journal of Lightwave Technology (JLT), 2015.

[Ah11-1] J. Ahmed, F. Solano, P. Monti, and L. Wosinska, "Traffic Re-Optimization Strategies for Dynamically Provisioned WDM Networks," in Proc. IEEE International Conference on Optical Network Design and Modeling (ONDM), 2011.

[Alcatel13-1] "Metro Network Traffic Growth: An Architecture Impact Study," Alcatel-Lucent Bell Labs strategic white paper, December 2013 [Online]. Available: http://resources.alcatel-lucent.com/asset/171568.

[An10-1] D. Andrei, M. Tornatore, C. U. Martel, D. Ghosal, and B. Mukherjee, "Provisioning Subwavelength Multicast Sessions With Flexible Scheduling Over WDM Networks," IEEE/OSA Journal of Optical

Communications and Networking (JOCN), vol. 2, pp. 241-255, 2010.

[As14-1] A. Asensio and L. Velasco, "Managing Transfer-based Datacenter Connections," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 6, pp. 660-669, 2014.

[As15-1] A. Asensio, L.M. Contreras, M. Ruiz, V. López, and L. Velasco, "Scalability of Telecom Cloud Architectures for Live-TV Distribution," in Proc. OSA Optical Fiber Communications Conference (OFC), 2015.

[Az11-1] S. Azodolmolky, J. Perelló, M. Angelou, F. Agraz, L. Velasco, S. Spadaro, Y. Pointurier, A. Francescon, C. V. Saradhi, P. Kokkinos, E. Varvarigos, S. Al Zahr, M. Gagnaire, M. Gunkel, D. Klonidis, and I. Tomkos, "Experimental Demonstration of an Impairment Aware Network Planning and Operation Tool for Transparent/Translucent Optical Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 29, pp. 439-448, 2011.

[Bon13-1] E. Bonetto, L. Chiaraviglio, F. Idzikowski, and E. Le Rouzic, "Algorithms for the multi-period power-aware logical topology design with reconfiguration costs," in IEEE/OSA Journal of Communications and Networking (JOCN), vol.5, pp.394-410, 2013.

[Bou05-1] E. Bouillet, J.F. Labourdette, R. Ramamurthy, and S. Chaudhuri, "Lightpath Re-Optimization in Mesh Optical Networks," IEEE/ACM Transactions on Networking, vol. 13, pp. 437-447, 2005.

[Bu14-1] J. Buysse, M. De Leenheer, L. M. Contreras, J. Aznar, J. Rodriguez, G. Landi, and C. Develder, "NCP+: an integrated network and IT control plane for cloud computing", Elsevier Optical Switching and Networking (OSN), vol. 11, pp. 137-152, 2014.

[Case11-1] R. Casellas, R. Martínez, R. Muñoz, L. Liu, T. Tsuritani, I. Morita, and M. Tsurusawa, "Dynamic virtual link mesh topology aggregation in multi-domain translucent WSON with hierarchical-PCE," OSA Optics Express, vol. 19, pp. B611-B620, 2011.

[Case13-1] R. Casellas, R. Muñoz, R. Martínez, and R. Vilalta, "Applications and Status of Path Computation Elements [Invited]," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. A192-A203, 2013.

[Cassandra] The Apache Cassandra Project [Online]. Available: http://cassandra. apache.org/.

[Cast12-1] A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, J. P. Fernández-Palacios, and D. Careglio, "Dynamic routing and spectrum (re)allocation in future flexgrid optical networks", Elsevier Computer Networks, vol. 56, pp. 2869-2883, 2012.

[Cast13-1] A. Castro, F. Paolucci, F. Fresi, M. Imran, B. Bhowmik, G. Berrettini, G. Meloni, A. Giorgetti, F. Cugini, L. Velasco, L. Poti, and P. Castoldi, "Experimental Demonstration of an Active Stateful PCE Performing Elastic Operations and Hitless Defragmentation," in Proc. European

Conference on Optical Communication (ECOC), 2013.

[Cast13-2] A. Castro, L. Velasco, J. Comellas, and G. Junyent, "Dynamic Restoration in Multi-layer IP/MPLS-over-Flexgrid Networks," in Proc. IEEE International Conference on Design of Reliable Communication Networks (DRCN), 2013.

[Cast14-1] A. Castro, R. Martínez, R. Casellas, L. Velasco, R. Muñoz, R. Vilalta, and J. Comellas, "Experimental Assessment of Bulk Path Restoration in Multi-layer Networks using PCE-based Global Concurrent Optimization," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 32, pp. 81-90, 2014.

[Cast14-2] A. Castro, L. Velasco, J. Comellas, and G. Junyent, "On the benefits of Multi-path Recovery in Flexgrid Optical Networks," Springer Photonic Network Communications (PNET), vol. 28, pp. 251-263, 2014.

[Cast14-3] A. Castro, "Off-Line and In-Operation Optical Core Networks Planning," PhD. thesis report, 2014.

[Che14-1] C. Chen, X. Chen, M. Zhang, S. Ma, Y. Shao, S. Li, M.S. Suleiman, and Z. Zhu, "Demonstrations of Efficient Online Spectrum Defragmentation in Software-Defined Elastic Optical Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 32, pp. 4701-4711, 2014.

[Chu07-1] X. Chu, T. Bu, and X. Li, "A Study of Lightpath Rerouting Schemes in Wavelength-Routed WDM Networks," in Proc. IEEE International Conference on Communications (ICC), 2007.

[Cisco14-1] "Cisco Global Cloud Index: Forecast and Methodology, 2013-2018," Cisco white paper, October 2014 [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf.

[Cl02-1] M. Clouqueur and W. Grover, "Availability analysis of span-restorable mesh networks," IEEE Journal on Selected Areas in Communications (JSAC), vol. 20, pp. 810-821, 2002.

[Col14-1] C. Colman-Meixner, F. Dikbiyik, M. F. Habib, M. Tornatore, C.-N. Chuah, and B. Mukherjee, "Disaster-survivable cloud-network mapping," Springer Photonic Network Communications (PNET), vol. 27, pp. 141-153, 2014.

[Con12-1] L. M. Contreras, V. López, O. Gonzalez de Dios, A. Tovar, F. Muñoz, A. Azañón, J. P. Fernandez-Palacios, and J. Folgueira, "Toward Cloud-Ready Transport Networks," IEEE Communications Magazine, vol. 50, pp. 48-55, 2012.

[CPLEX] IBM CPLEX Optimizer [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[Cu12-1] F. Cugini, G. Meloni, F. Paolucci, N. Sambo, M. Secondini, L. Gerardi, L. Potí, and P. Castoldi, "Demonstration of Flexible Optical Network Based on Path Computation Element," IEEE/OSA Journal of Lightwave

Technology (JLT), vol. 30, pp. 727-733, 2012.

[Cu13-1] F. Cugini, F. Paolucci, G. Meloni, G. Berrettini, M. Secondini, F. Fresi, N. Sambo, L. Potí, and P. Castoldi, "Push-pull defragmentation without traffic disruption in flexible grid optical networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 31, pp. 125–133, 2013.

[De12-1] C. Develder, M. de Leenheer, B. Dhoedt, M. Pickavet, D. Colle, F. de Truck, and P. Demeester, "Optical Networks for Grid and Cloud Computing Applications," in Proc. of the IEEE, vol. 100, pp. 1149-1167, 2012.

[Di14-1] F. Dikbiyik, M. Tornatore, and B. Mukherjee, "Minimizing the Risk From Disaster Failures in Optical Backbone Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 32, pp. 3175-3183, 2014.

[draft-bgpls] H. Gredler, J. Medved, S. Previdi, A. Farrel, and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP," IETF draft draft-ietf-idr-ls-distribution-13, work in progress, 2015.

[draft-flexlabel] A. Farrel, D. King, Y. Li, and F. Zhang, "Generalized Labels for the Flexi-Grid in Lambda Switch Capable (LSC) Label Switching Routers," IETF draft draft-ietf-ccamp-flexigrid-lambda-label-05, work in progress, 2015.

[draft-initiate] E. Crabbe, I. Minei, S. Sivabalan, and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model," IETF draft draft-ietf-pce-pce-initiated-lsp-05, work in progress, 2015.

[draft-intlayer] E. Oki, T. Takeda, J-L Le Roux, A. Farrel, and F. Zhang, "Extensions to the Path Computation Element communication Protocol (PCEP) for Inter-Layer MPLS and GMPLS Traffic Engineering," IETF draft draft-ietf-pce-inter-layer-ext-08, work-in-progress, 2014.

[draft-restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF draft draft-ietf-netconf-restconf-09, work in progress, 2015.

[draft-stateful] E. Crabbe, I. Minei, J. Medved, and R. Varga, "PCEP Extensions for Stateful PCE," IETF draft draft-ietf-pce-stateful-pce-13, work in progress, 2015.

[Ei15-1] A. Eira, J. Pedro, and J. Pires, "Optimal Multi-Period Provisioning of Fixed and Flex-Rate Modular Line Interfaces in DWDM Networks" IEEE/OSA Journal of Communications and Networking (JOCN), vol. 7, pp. 223-234, 2015.

[Ericsson12-1] "The Telecom Cloud Opportunity," Ericsson discussion paper, March 2012 [Online]. Available: http://www.ericsson.com/res/site_AU/docs/2012/ericsson_telecom_cloud_discussion_paper.pdf.

[Floodlight] Project Floodlight [Online]. Available: http://www.projectfloodlight.org/.

[G.694.1] "Spectral grids for WDM applications: DWDM frequency grid," ITU-T Rec. G.694.1, 2012.

[G.8080] "Architecture for the Automatically Switched Optical Network," ITU-T

Recommendation G.8080, 2012.

[Gea01-1] N. Geary, A. Antonopoulos, E. Drakopoulos, and J. O'Reilly, "Analysis of optimisation issues in multi-period DWDM network planning," in Proc. IEEE International Conference on Computer Communications (INFOCOM), vol.1, pp.152-158, 2001.

[Ger12-1] O. Gerstel, M. Jinno, A. Lord, and S.J.B. Yoo, "Elastic optical networking: a new dawn for the optical layer?" IEEE Communications Magazine, vol. 50, pp. s12-s20, 2012.

[Gh12-1] M. Gharbaoui, B. Martini, and P. Castoldi, "Anycast-Based Optimizations for Inter-Data-Center Interconnections [Invited]," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 4, pp. B168-B178, 2012.

[Gr04-1] W. D. Grover, "Mesh-Based Survivable Networks," Prentice Hall PTR, New Jersey, 2004.

[Ha04-1] J. R. K. Hartline, R. Libeskind-Hadas, K. M. Dresner, E. W. Drucker, and K. J. Ray, "Optimal virtual topologies for one-to-many communication in WDM paths and rings," IEEE/ACM Transactions on Networking, vol. 12, pp. 375-383, 2004.

[Hu10-1] S. Huang, M. Xia, C. Martel, and B. Mukherjee, "A Multistate Multipath Provisioning Scheme for Differentiated Failures in Telecom Mesh Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 28, pp. 1585-1596, 2010.

[Ji09-1] M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka, "Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies," IEEE Communications Magazine, vol. 47, pp. 66-73, 2009.

[Ji10-1] M. Jinno, B. Kozicki, H. Takara, A. Watanabe, Y. Sone, T. Tanaka, and A. Hirano, "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path (SLICE) network," IEEE Communications Magazine, vol. 48, pp. 138-145, 2010.

[Ji12-1] M. Jinno, H. Takara, Y. Sone, K. Yonenaga, and A. Hirano, "Multiflow Optical Transponder for Efficient Multilayer Optical Networking," IEEE Communications Magazine, vol. 50, pp. 56-65, 2012.

[Ka12-1] N. Katica and A. Tahirović "Opportunities for telecom operators in cloud computing business," in Proc. IEEE 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2012.

[Lin13-1] R. Lin, M. Zukerman, G. Shen, and W. Zhong, "Design of Light-Tree Based Optical Inter-Datacenter Networks," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. 1443-1455, 2013.

[M.3010] "Principles for a Telecommunications Management Network," ITU-T Recommendation M.3010, 2000.

[Mah11-1] A. Mahimkar, A. Chiu, R. Doverspike, M. D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. L. Woodward, and J. Yates, "Bandwidth on Demand for Inter-Data Center Communication," in Proc. 10th ACM Workshop on Hop Topics in Networks (HotNets-X), 2011.

[Mas14-1] E. Masala, A. Servetti, S. Basso, J. C. De Martin, "Challenges and Issues on Collecting and Analyzing Large Volumes of Network Data Measurements,", In: B. Catania, T. Cerquitelli, S. Chiusano, G. Guerrini, M. Kämpf, A. Kemper, B. Novikov, T. Palapanas, J. Pokorný, and A. Vakali, "New Trends in Databases and Information Systems", in Advances in Intelligent Systems and Computing series, Springer, 2014.

[Mat12-1] F. Mata, J. García-Dorado, J. Aracil, "Detection of traffic changes in large-scale backbone networks: The case of the Spanish academic network," Elsevier Computer Networks, vol. 56, pp. 686–702, 2012.

[Mel01-1] M. Mellia, A. Nucci, A. Grosso, E. Leonardi, and M.A. Marsan, "Optimal design of logical topologies in wavelength-routed optical networks with multicast traffic," in Proc. IEEE Global Communications Conference (GLOBECOM), 2001.

[Meu10-1] C. Meusburger and D. Schupke, "Optimizing the migration of channels with higher bitrates," IEEE/OSA Journal of Lightwave Technology (JLT), vol.28, pp.608-615, 2010.

[Mo16-1] F. Morales, M. Ruiz, and L. Velasco, "Virtual Network Topology Reconfiguration based on Big Data Analytics for Traffic Prediction," accepted in IEEE/OSA Optical Fiber Communication Conference (OFC), 2016.

[Nap15-1] A. Napoli, M. Bohn, D. Rafique, A. Stavdas, N. Sambo, L. Potì, M. Nölle, J.K. Fischer, E. Riccardi, A. Pagano, A. Di Giglio, M.S. Moreolo, J.M. Fabrega, E. Hugues-Salas, G. Zervas, D. Simeonidou, P. Layec, A. D'Errico, T. Rahman, and J.P. Fernández-Palacios, "Next Generation Elastic Optical Networks: the Vision of the European Research Project IDEALIST," IEEE Communications Magazine, vol. 53, pp. 152-162, 2015.

[Nvidia] NVIDIA – What is GPU Computing [Online], Available: http://www.nvidia.com/object/what-is-gpu-computing.html.

[OIF15-1] "Framework for Transport SDN: Components and APIs," Optical Internetworking Forum white paper, May 2015 [Online]. Available: http://www.oiforum.com/wp-content/uploads/OIF-FD-Transport-SDN-01-0.pdf.

[OMNet] OMNeT++ [Online]. Available: http://www.omnetpp.org/.

[ONF] Open Networking Foundation [Online]. Available: https://www.opennetworking.org/.

[ONF-TR509] "Optical Transport Use Cases," Optical Networking Foundation ONF TR-509, 2014 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/optical-transport-use-

cases.pdf.

[OpenFlow] OpenFlow [Online]. Available: http://www.openflow.org/.

[OpenIRIS] OpenIRIS: The Recursive SDN Openflow Controller by ETRI [Online]. Available: http://openiris.etri.re.kr/.

[OpenStack] OpenStack [Online]. Available: http://www.openstack.org/.

[Pa11-1] F. Paolucci, M. Gharbaoui, A. Giorgetti, F. Cugini, B. Martini, L. Valcarenghi, and P. Castoldi, "Preserving Confidentiality in PCE-based Multi-domain Networks," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 3, pp. 465-474, 2011.

[Pao13-1] F. Paolucci, F. Cugini, A. Giorgetti, N. Sambo, and P. Castoldi, "A Survey on the Path Computation Element (PCE) Architecture," IEEE Communications Surveys & Tutorials, vol. 15, pp. 1819-1841, 2013.

[Pao14-1] F. Paolucci, A. Castro, F. Cugini, L. Velasco, and P. Castoldi, "Multipath Restoration and Bitrate Squeezing in SDN-based Elastic Optical Networks," (Invited Paper) Springer Photonic Network Communications (PNET), vol. 28, pp. 45-57, 2014.

[Pao15-1] F. Paolucci, A. Castro, F. Fresi, M. Imran, A. Giorgetti, B. Bhownik, G. Berrettini, G. Meloni, F. Cugini, L. Velasco, L. Potí, and P. Castoldi, "Active PCE Demonstration performing Elastic Operations and Hitless Defragmentation in Flexible Grid Optical Networks," Springer Photonic Network Communications (PNET), vol. 29, pp. 57-66, 2015.

[Par14-1] S. M. Park, S. Ju, and J. Lee, "Efficient routing for traffic offloading in Software-defined Network," Elsevier Procedia Computer Science vol 34, pp. 674-679, 2014.

[Pe02-1] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm," Journal of the ACM, vol. 49, pp. 16-34, 2002.

[Per11-1] J. Perelló, S. Spadaro, F. Agraz, M. Angelou, S. Azodolmolky, Y. Qin, R. Nejabati, D. Simeonidou, P. Kokkinos, E. Varvarigos, S. Al Zahr, M. Gagnaire, and I. Tomkos, "Experimental Evaluation of Centralized Failure Restoration in a Dynamic Impairment-Aware All-Optical Network," in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2011.

[Pet02-1] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm," Journal of the ACM, vol. 49, pp. 16-34, 2002.

[POX] POX SDN Controller [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki

[Qin13-1] C. Qin, R. Proietti, B. Guan, Y. Yin, R. Scott, R. Yu, and S. J. B. Yoo, "Demonstration of Multi-channel Hitless Defragmentation with Fast Auto-tracking Coherent RX LOs," in Proc. IEEE/OSA Optical Fiber Communication Conference (OFC), 2013.

[Qiu11-1] J. Qiu, Y. Liu, G. Mohan, and K. C. Chua, "Fast spanning tree reconnection mechanism for resilient Metro Ethernet networks," Elsevier

Computer Networks, vol. 55, pp. 2717-2729, 2011.

[Rah09-1] T. Rahman, G. Ellinas, and M. Ali, "Lightpath- and Light-Tree-Based Groupcast Routing and Wavelength Assignment in Mesh Optical Networks," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 1, pp. A44-A55, 2009.

[Ramas95-1] R. Ramaswami and K. Sivarajan, "Routing and wavelength assignment in all-optical networks," IEEE Transactions on Networking, vol. 3, pp. 489-500, 1995.

[RFC3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF RFC 3209, 2001.

[RFC3945] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," IETF RFC3945, 2004.

[RFC4655] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture," IETF RFC4655, 2006.

[RFC5212] K. Shiomoto, D. Papadimitriou, JL. Le Roux, M. Vigoureux, and D. Brungard, "Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)," IETF RFC5212, 2008.

[RFC5376] N. Bitar, R. Zhang, K. Kumaki, "Inter-AS Requirements for the Path Computation Element Communication Protocol (PCECP)," IETF RFC5376, 2008.

[RFC5440] JP. Vasseur and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)," IETF RFC5440, 2009.

[RFC5441] JP. Vasseur, R. Zhang, N. Bitar, JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths," IETF RFC5441, 2009.

[RFC5557] Y. Lee, JL. Le Roux, D. King, and E. Oki, "Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization," IETF RFC5557, 2009.

[RFC5623] E. Oki, T. Takeda, JL. Le Roux, and A. Farrel, "Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering," IETF RFC5623, 2009.

[RFC6006] Q. Zhao, D. King, F. Verhaeghe, T. Takeda, Z. Ali, and J. Meuric, "Extensions to the Path Computation Element Communication Protocol (PCEP) for Point-to-Multipoint Traffic Engineering Label Switched Paths," IETF RFC6006, 2010.

[RFC6007] I. Nishioka and D. King, "Use of the Synchronization VECtor (SVEC) List for Synchronized Dependent Path Computations," IETF RFC6007, 2010.

[RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," IETF RFC6020, 2010.

[RFC6805] D. King and A. Farrel, "The Application of the Path Computation Element

Architecture to the Determination of a Sequence of Domains in MPLS and GMPLS," IETF RFC6805, 2012.

[RFC7491] D. King and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations," IETF RFC7491, 2015.

[RFC7536] M. Linsner, P. Eardley, T. Burbridge, F. Sorensen, "Large-Scale Broadband Measurement Use Cases," IETF RFC7536, 2015.

[RFC7698] O. González de Dios, R. Casellas, F. Zhang, X. Fu, D. Ceccarelli, and I. Hussain, "Framework and Requirements for GMPLS-based control of Flexi-grid Dense Wavelength Division Multiplexing (DWDM) Networks," IETF RFC7698, 2015.

[Ri12-1] O. Rival, A. Morea, H. Drid, N. Brochier, and E. Le Rouzic, "Upgrading Optical Networks with Elastic Transponders," ," in Proc. European Conference on Optical Communication (ECOC), 2012.

[Ru13-1] M. Ruiz, M. Pióro, M. Zotkiewicz, M. Klinkowski, and L. Velasco, "Column Generation Algorithm for RSA Problems in Flexgrid Optical Networks," Springer Photonic Network Communications (PNET), vol. 26, pp. 53-64, 2013.

[Ru14-1] M. Ruiz and L. Velasco, "Performance Evaluation of Light-tree Schemes in Flexgrid Optical Networks," IEEE Communications Letters, vol. 18, pp. 1731-1734, 2014.

[Ru14-2] M. Ruiz, A. Lord, D. Fonseca, M. Pióro, R. Wessäly, L. Velasco, and J.P. Fernández-Palacios, "Planning Fixed to Flexgrid Gradual Migration: Drivers and Open Issues," IEEE Communications Magazine, vol. 52, pp. 70-76, 2014.

[Ru14-3] M. Ruiz, M. Zotkiewicz, A. Castro, M. Klinkowski, L. Velasco, and M. Pioro, "After Failure Repair Optimization in Dynamic Flexgrid Optical Networks," in Proc. OSA Optical Fiber Communication Conference (OFC), 2014.

[Sah99-1] L. H. Sahasrabuddhe and B. Mukherjee, "Light-trees: Optical multicasting for improved performance in wavelength-routed networks," IEEE Communications Magazine, vol. 37, pp. 67-73, 1999.

[Sam13-1] N. Sambo, G. Meloni, G. Berrettini, F. Paolucci, A. Malacarne, A. Bogoni, F. Cugini, L. Potì, and P. Castoldi, "Demonstration of data and control plane for optical multicast at 100 and 200 Gb/s with and without frequency conversion," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. 667-676, 2013.

[Sam15-1] N. Sambo, P. Castoldi, A. D'Errico, E. Riccardi, A. Pagano, M.S. Moreolo, J.M. Fàbrega, D. Rafique, A. Napoli, S. Frigerio, E. Hugues-Salas, G. Zervas, M. Nölle, J.K. Fischer, A. Lord, and J.P. Fernández-Palacios, "Next generation sliceable bandwidth variable transponders," IEEE Communications Magazine, vol. 53, pp. 163-171, 2015.

[Sav14-1] S. Savas, F. Dikbiyik, M. F. Habib, M. Tornatore, and B. Mukherjee,

"Disaster-aware service provisioning with manycasting in cloud networks," Springer Photonic Network Communications (PNET), vol. 28, pp. 123-134, 2014.

[Sh07-1] N. Shinomiya, T. Hoshida, Y. Akiyama, H. Nakashima, and T. Terahara, "Hybrid link/path-based design for translucent photonic network dimensioning," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 25, pp. 2931–2941, 2007.

[Sou07-1] A. Soule, F. Silveira, H. Ringberg, and C. Diot, "Challenging the supremacy of traffic matrices in anomaly detection," in Proc. ACM Special Interest Group on Data Communication (SIGCOMM), 2007.

[Son11-1] Y. Sone, A. Watanabe, W. Imajuku, Y. Tsukishima, B. Kozicki, H. Takara, M. Jinno, "Bandwidth squeezed restoration in spectrum-sliced elastic optical path networks (slice)," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 3, pp. 223-233, 2011.

[Spark] The Apache Spark Project [Online]. Available: http://spark.apache.org/

[St06-1] S. Strauss, A. Kirstadter, and D. Schupke, "Multi-period planning of WDM-networks: comparison of incremental and EoL approaches," in Proc. International Conference in Central Asia on Internet (ICI), 2006.

[StrongestD34] "Final report WP3 activity and performance analysis," ICT STRONGEST Project WP3 deliverable D3.4, 2012. [Online]. Available: http://www.ict-strongest.eu/upload-files/deliverables-2/deliverable-d3-4-48.

[Ve10-1] L. Velasco, F. Agraz, R. Martínez, R. Casellas, S. Spadaro, R. Muñoz, and G. Junyent, "GMPLS-based Multi-domain Restoration: Analysis, Strategies, Policies and Experimental Assessment," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 2, pp. 427-441, 2010.

[Ve13-1] L. Velasco, A. Asensio, J.Ll. Berral, V. López, D. Carrera, A. Castro, and J.P. Fernández-Palacios, "Cross-Stratum Orchestration and Flexgrid Optical Networks for Datacenter Federations," IEEE Network Magazine, vol. 27, pp. 23-30, 2013.

[Ve14-1] L. Velasco, D. King, O. Gerstel, R. Casellas, A. Castro, and V. López, "In-Operation Network Planning," IEEE Communications Magazine, vol. 52, pp. 52-60, 2014.

[Ve14-2] L. Velasco, A. Castro, M. Ruiz, and G. Junyent, "Solving Routing and Spectrum Allocation Related Optimization Problems: from Off-Line to In-Operation Flexgrid Network Planning," (Invited Tutorial) IEEE/OSA Journal of Lightwave Technology (JLT), vol. 32, pp. 2780-2795, 2014.

[Ve14-3] L. Velasco, A. Asensio, J.Ll. Berral, A. Castro, and V. López, "Towards a Carrier SDN: An example for Elastic Inter-Datacenter Connectivity," (Invited Paper) OSA Optics Express, vol. 22, pp. 55-61, 2014.

[Ve14-4] L. Velasco, A. Asensio, J. Ll. Berral, E. Bonetto, F. Musumeci, and V. López, "Elastic Operations in Federated Datacenters for Performance and

Cost Optimization," Elsevier Computer Communications, vol. 50, pp. 142-151, 2014.

[Ve15-2] L. Velasco, L.M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernández-Palacios, "A Service-Oriented Hybrid Access Network and Cloud Architecture," IEEE Communications Magazine, vol. 53, pp. 159-165, 2015.

[Wal14-1] K. Walkowiak, A. Kasprzak, and M.Klinkowski, "Dynamic routing of anycast and unicast traffic in Elastic Optical Networks," in Proc. IEEE International Conference on Communications (ICC), 2014.

[Wan13-1] R. Wang and B. Mukherjee, "Provisioning in Elastic Optical Networks with Non-Disruptive Defragmentation," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 31, pp. 2491-2500, 2013.

[Wo13-1] S. L. Woodward, W. Zhang, B. G. Bathula, G. Choudhury, R. K. Sinha, M. D. Feuer, J. Strand, and A. L. Chiu, "Asymmetric Optical Connections for Improved Network Efficiency," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. 1195-1201, 2013.

[Wr13-1] P. Wright, A. Lord, and L. Velasco, "The Network Capacity Benefits of Flexgrid," in Proc. IEEE International Conference on Optical Network Design and Modeling (ONDM), 2013.

[Wu11-1] J. Wu, "A survey of WDM network reconfiguration: Strategies and triggering methods," Elsevier Computer Networks, vol. 55, pp. 2622-2645, 2011.

[Xilinx] Xilinx – SDN/NFV Solutions Powered by Xilinx [Online]. Available: http://www.xilinx.com/applications/megatrends/sdn-nfv.html.

[Ya03-1] D.-N. Yang and W. Liao, "Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks," in Proc. IEEE International Conference on Computer Communications (INFOCOM), vol.1, pp.32-41, 2003.

[Yo14-1] S. J. B. Yoo, "Multi-domain Cognitive Optical Software Defined Networks with Market-Driven Brokers," in Proc. European Conference on Optical Communication (ECOC), 2014.

[Yu13-1] X. Yu, G. Xiao, and T. Cheng, "Dynamic multicast traffic grooming in optical WDM mesh networks: Lightpath versus light-tree," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 5, pp. 870-880, 2013.

[Zha04-1] J. Zhang and B. Mukherjee, "A Review of Fault Management in WDM Mesh Networks: Basic Concepts and Research Challenges," IEEE Network, vol. 18, pp. 41-48, 2004.

[Zho05-1] Y. Zhou and G.-S. Poo, "Optical multicast over wavelength-routed WDM networks: A survey," Elsevier Optical Switching and Networking (OSN), vol. 2, pp. 176-197, 2005.

[Zhu12-1] Z. Zhu, X. Chen, F. Ji, L. Zhang, F. Farahmand, and J. Jue, "Energy-

Efficient Translucent Optical Transport Networks with Mixed Regenerator Placement," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 30, pp. 3147-3156, 2012.

[Zhu15-1] Z. Zhu, X. Chen, C. Chen, S. Ma, M. Zhang, L. Liu, and S. J. B. Yoo, "OpenFlow-Assisted Online Defragmentation in Single-/Multi-Domain Software-Defined Elastic Optical Networks," (Invited paper) IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 7, pp. A7-A15, 2015.

[Zo15-1] M. Zotkiewicz, M. Ruiz, M. Klinkowski, M. Pióro, and L. Velasco, "Reoptimization of Dynamic Flexgrid Optical Networks After Link Failure Repairs," IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 7, pp. 49-61, 2015.