Experimental Assessment of In-Operation Spectrum Defragmentation

Ll. Gifre[1*], F. Paolucci[2], A. Aguado[3], R. Casellas[4], A. Castro[1], F. Cugini[2], P. Castoldi[2], L. Velasco[1], and V. López[3]

[1] Optical Communications Group (GCO), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.
[2] Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Pisa, Italy.
[3] Telefónica Investigación y Desarrollo (TID), Madrid, Spain.
[4] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain

*Corresponding author: lgifre@ac.upc.edu

Abstract—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 extend that spectrum fragmentation, the central frequency of already established connections can be shifted to create wider spectrum contiguous fragments to be allocated to incoming connections; this is called spectrum defragmentation. In this paper, we propose using the so called ABNO architecture, currently under standardization in the IETF, to deal with the defragmentation use case while the network is in operation. A workflow involving several elements in the ABNO architecture is proposed and experimentally assessed in a distributed test-bed connecting facilities in three mayor European cities.

***Keywords*: Flexgrid optical networks, Spectrum Defragmentation, In-Operation Planning.**

1 INTRODUCTION

The recent availability of novel switching technologies is driving the introduction of flexgrid optical networks [1], [2]. In flexgrid networks, the fixed grid spacing is removed and an optical connection is allocated to a *frequency slot* composed of a number of contiguous portions of spectrum resources, named *slices*, of a fixed spectral width (e.g., 6.25GHz). To set up an optical connection on a flexgrid optical network, the Routing and Spectrum Allocation (RSA) [3] needs to be solved to find a physical route with a set of slices *continuous* along the route and *contiguous* in the optical spectrum. The width of the allocated slot should be according to the requested bitrate, the selected modulation format, and the considered grid. The slices are allocated around a central frequency and hence, the number of allocated slices must be an even number.

In dynamic scenarios, spectrum fragmentation appears in flexgrid networks subject to both the continuity and the contiguity constraints. As a consequence of the high cost of spectrum converters, they are rarely used in optical networks and therefore, spectrum fragmentation appears, increasing the blocking probability of connection requests degrading the network Grade of Service (GoS).

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 [4]-[7]; those strategies can be divided into *proactive* and *reactive* considering the way they are triggered [4]. The proactive strategy focuses on minimizing fragmentation itself at given period of time, whereas *reactive*, also known as *provisioning-triggered*, focuses on making enough room for a given connection request if it cannot be established with current resources allocation. Periodic defragmentation, requiring long computation times as a result of the amount of data to be processes, is essentially performed during low activity periods, e.g. during nights [5]. Conversely, path-triggered defragmentation, involving only a limited set of already established connections, might provide solutions in shorter times and can be run in real time [6], [7].

To minimize traffic disruption, the standardized *make-before-break* rerouting technique, included in the standardized resource reservation protocol traffic engineering (RSVP-TE) protocol [8], can be used for signaling the new connection for rerouting or reallocation and transferring traffic from the current connection to the new one before the old path is finally torn down. In optical networks, make-before-break requires additional resources to support two parallel connections such as spare transponders, which entail additional costs.

Alternatively, the recently introduced *push-pull* technique [9] can be used to shift the central frequency of established optical connections without traffic disruption. Push-pull consists in re-tuning the transmitter laser from the original to the target nominal central frequency, while the receiver is automatically pulled to track the signal shifting. The

limitation of the central frequency shifting performed using the push-pull technique resides on the fact that connections can be allocated to new slots as long as no other connections are established between the current and the new slot, reducing thus the set of reallocations available for each connection.

To perform spectrum defragmentation in a network under operation, i.e. transporting real traffic, a carefully analysis of the functionalities that are needed, not only at the data plane, but also at the control and management planes, has to be carried out. For the control plane, in this work we rely on the recently proposed Applications Based Network Operations (ABNO) architecture [10]. The ABNO architecture describes an SDN-based framework, combining existing technologies and functional elements for managing information regarding topology and available resources in a network, and for requesting path computations, connection provisioning or reserving network resources. The key component within ABNO is the Path Computation Element (PCE) [11], [12], which can be used for computing paths and is further extended to provide policy enforcement capabilities for ABNO.

As for the management plane, a Network Management System (NMS) or an Operations Support System (OSS) is used to operate the network. In particular, the NMS issues high-level service requests to the ABNO Controller and establishes the provisioning policies to exploit network resources and functionalities [10].

To perform optimization-related computations, the PCE can be extended with a module featuring Global Concurrent Optimization (GCO) [13], [14]. However, because of its specialization, it might be desirable that a separated element, a so called planning tool, able to perform complex optimization-related computations, is deployed. Aiming at reducing network cost by minimising the over-provisioning, optimization requests can be issued to the planning tool to reconfigure and/or re-optimize the network on demand and in real-time. Re-optimization performed while the network is being operated is called in-operation planning [15]. The planning tool is also part of the ABNO architecture provided that standard interfaces are used for optimization requests and responses.

The remainder of this paper is organized as follows. Section 2 first introduces the provisioning-triggered spectrum defragmentation problem. Then, the control and management architecture considered in this paper is presented and the proposed in-operation spectrum defragmentation use case is described in terms of interrelation among the elements in the considered architecture. Current standards are eventually reviewed to verify how they can deal with the defragmentation problem. Section 3 formally states the hitless defragmentation problem and presents an Integer Linear Programming (ILP) to solve it. As a result of the stringent computation times in which the problem needs to be solved, a heuristic algorithm is proposed. Section 4 presents our proposal for provisioning-triggered defragmentation. The distributed algorithm to be executed is presented and auxiliary algorithms are proposed together with the specific contents and semantic of exchanged messages that we suggest to deal with the use case. Section 5 focuses on experimentally evaluated the feasibility of our proposal. To that end, a distributed test-bed connecting Telefonica's, CNIT's and UPC's premises is presented. Finally, section 6 concludes the paper.

2 ARCHITECTURE AND USE CASE

Aiming at illustrating spectrum fragmentation, Fig.1 shows an example on the small network topology depicted in Fig.1a, where each node and link is labeled. The entire spectrum width consists of 16 slices. Fig.1b represents the utilization of each frequency slice in the network, where a number of optical connections are already established. In this scenario, the connection request between nodes 4 and 7 requesting 4 slices cannot be served. 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. 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.1c, connections *p4* and *p5* have been shifted making enough room for *newP*.

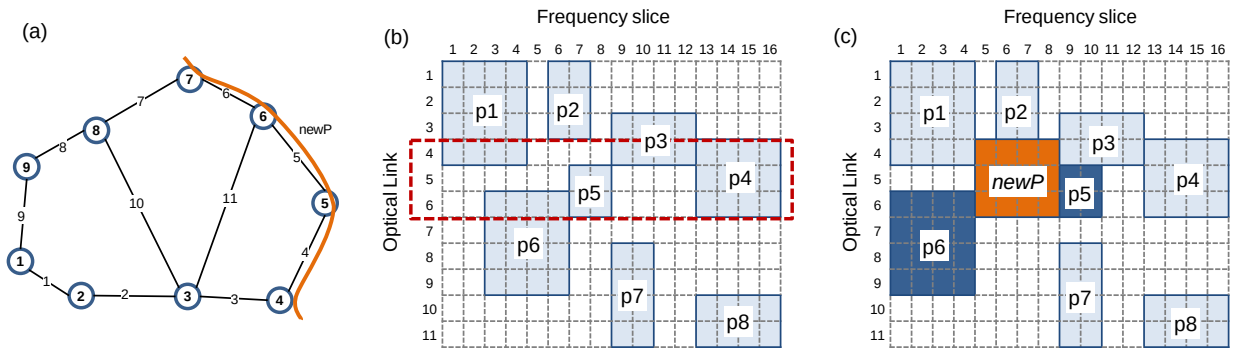

Fig.1 Example of spectrum reallocation by shifting the central frequency of some already established optical connections.

Let us now analyze how spectrum defragmentation can be implemented in real networks. Next section focuses on the proposed control and management planes architecture.

2.1 Control and management planes

The control plane of the network presented here is implemented in terms of the ABNO architecture [10], currently under standardization in the IETF. The ABNO architecture, illustrated in Fig.2, includes the functional components needed to efficiently satisfy high-level service requests issued by the network management system (NMS).

The *path computation element* (PCE) is able to perform constrained path computation on a graph representing a network. The PCE can be implemented with stateless or stateful functionality. In the stateless architecture, the PCE relies on a traffic engineering database (TED), which includes information about resource utilization. The stateful PCE architecture extends the stateless one maintaining individual Label Switched Path (LSP) state information in the LSP State Database (LSP-DB) [16]. LSP state information includes its route, bandwidth and spectrum allocation, switching types and LSP constraints. Additionally, a stateful PCE may also include the active functionality that enables the PCE to issue recommendations to the network, e.g. to dynamically update LSP parameters.

Besides the PCE and the *topology module* containing network databases, i.e. TED and LSP-DB, the architecture includes the following main elements: *i*) The *ABNO controller* is responsible for orchestrating the rest of the elements, invoking them in the right order. The controller listens for requests arriving through the *north-bound interface* coming from the NMS or any technology-specific OSS and selects the appropriate workflow to follow so as to satisfy each request; *ii*) The *policy agent* is the entity that regulates the use and access to network functionalities and resources. It is responsible for propagating those policies to other components in the ABNO architecture; *iii*) The *provisioning manager* is in charge of communicating the network elements in the data plane new configurations, including connections provisioning and updating. To that end, several protocols can be used including Generalized Multi-Protocol Label Switching (GMPLS) signaling (i.e. RSVP-TE) or even directly *programming* individual network devices by means of the OpenFlow protocol [17]. In this paper, without loss of generality, we assume the former.

Some other elements are also included in the ABNO architecture. However, since they are not used in this work we refer the reader to [10].

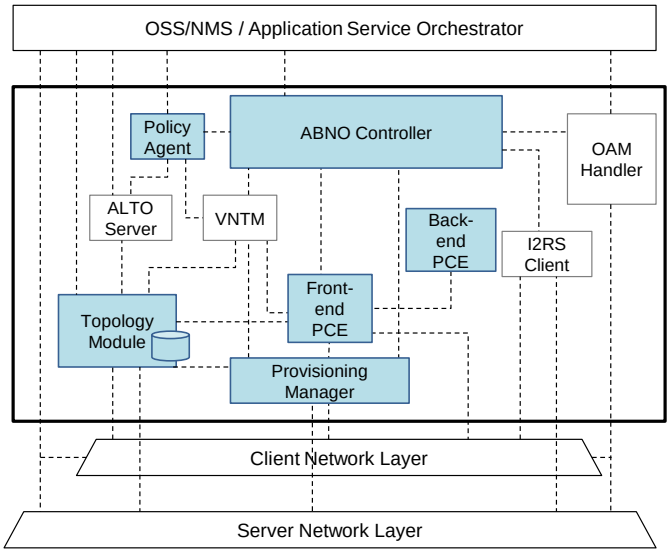

Fig.2 ABNO architecture

In addition to the standard ABNO architecture, we assume the front-end/back-end PCE architecture proposed in [11]. For the defragmentation use case, the front-end PCE must be active and stateful to be capable of updating existing connections and must include instantiation capabilities to be able to initiate connections set-up [18]. The back-end PCE is a planning tool prepared to solve optimization problems such as spectrum defragmentation. Consequently, the front-end PCE plays the role of a Path Computation Client (PCC), delegating some complex computations to the back-end.

2.2 Use case description

The main purpose of spectrum defragmentation is improving network resource utilization and thus, its GoS. 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. Some other attributes, not covered in this work, can be updated, e.g. its allocated spectrum width (i.e., due to a change of modulation formats or bitrate).

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 front-end PCE 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 front-end PCE, in charge of computing and finally coordinate connection establishment.

The workflow that represents the provisioning-triggered defragmentation use case is detailed in Fig.3. 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.3). 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 front-end PCE to compute the route and eventually set up the optical connection (3).

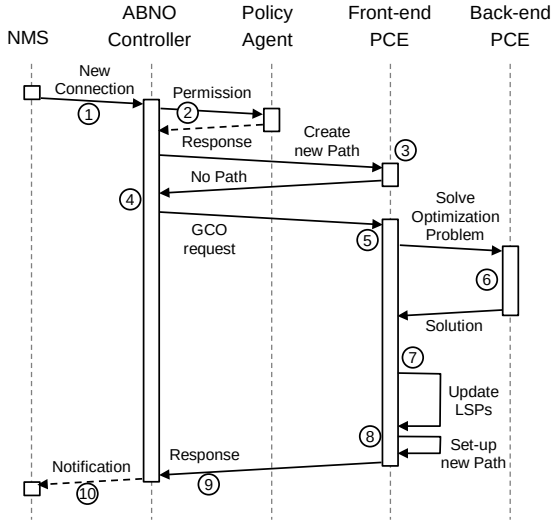

Fig.3 Defragmentation workflow

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 front-end PCE (5). When the front-end PCE receives the request for defragmentation, it checks its feasibility and gathers information to create a GCO request that is sent towards the back-end PCE to solve the optimization problem (6). When the back-end PCE ends, it sends back the solution found. The front-end PCE 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 front-end PCE proceeds to establish the requested connection (8). Upon its completion, the front-end PCE notifies the ABNO controller (9), which in turn notifies the NMS (10).

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

All interactions between ABNO and PCEs are done by exchanging PCE protocol (PCEP) messages. In particular, Path Computation Request (PCReq) and Path Computation Reply (PCRep) messages are exchanged between front-end and back-end PCEs (step 6 in Fig.3). Next, we analyze up to what extent current standards can deal with the use case presented.

2.3 Current standards review

Let us firstly to start with the selection of the front-end PCE architecture. Since the defragmentation use case involves the front-end PCE updating already established connections, it is clear that related information must be available at the front-end PCE and hence a stateful PCE architecture is required. Note that, in contrast to the stateless PCE, the stateful PCE architecture explicitly includes a connection database (LSP-DB).

In addition, currently defined PCEP RFCs and ongoing drafts have partial support for the defragmentation use case

presented herein. In support to network optimization, PCEP extensions for GCO were standardized [13]. A GCO path computation request will simultaneously consider the state of network resources in the TED and a set of connections together with their respective constraints. GCO can be applied to the defragmentation use case, since can be applied to re-compute the route and spectrum allocation of a set of existing connections, taking into account also routing new services.

When GCO is used to request a defragmentation procedure, the front-end PCE specifies, at each request, the existing connections to be considered. For that end, it would be desirable to identify connections, e.g. by means of the SYMBOLIC_NAME attribute within the `LSP` object defined in [16]. Unfortunately, current drafts do not support a path computation request referring to existing LSPs in the LSP-DB and therefore, they need to be explicitly conveyed and identified using its end points and used resources, i.e. route and spectrum allocation. This is achieved by using the `END_POINTS` and record route (`RRO`) objects. This may entail scalability issues in case 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 the request for the new connection: a `RP` object identifying the request, its endpoints and relevant constraints such as `BANDWIDTH`, or Include Route Object (`IRO`). The request for the new connection is complemented with a set of requests related to existing connections candidate to be shifted. All these requests are grouped using the synchronized vector (`SVEC`) object to ensure that existing connections and the new one are jointly considered in the optimization. Finally, note that the Objective Function (`OF`) object can be used to specify the desired network-wide GCO related criterion, such as "Defragmentation".

Regarding PCRep messages, the back-end 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 [19] or the `NO-PATH` object in case that no feasible solution was found. In the latter case, disregarding the computed `ERO`, no shifting should be performed to the existing connections.

Spectrum allocation can be described using the tuple {central frequency, slot width}. The central frequency can be defined in terms of the number *n* of slices (positive, negative or 0) from a reference frequency (193.1 THz), whereas the number *m* of slices at each side of the central frequency can be used for the slot width. Therefore, the tuple {*n*, *m*} unambiguously describes any spectrum allocation.

Because the relative execution order among existing connections is important, a TLV can be appended to the `RP` object of each response to specify the *delete* and *set-up* order of each existing connection. By convention, receiving set-up and delete orders with the same value implies that the connection stays unmodified. Nonetheless, that *by default* behavior must be modified since connections are neither torn down nor set-up, simply updated, i.e. its central frequency shifted.

Regarding the interactions with other modules in the ABNO architecture, no standards are available yet. This is the case of the Policy Manager and the north-bound interface of the ABNO controller. As a consequence, we have used our own implementation.

3 THE SPECTRUM SHIFTING (SPRING) PROBLEM

As discussed in section 2, to avoid traffic disruption we propose using spectrum shifting for defragmentation. To that end in this section we firstly formally state the problem and then present an ILP model to solve the defragmentation problem. The model is based on our previous work in [6], 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.

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 $e \in E$,
- 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.

An ILP model for the SPRING problem is presented next. The 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 $P \backslash Pm$ 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.

Additionally, the decision variables are:

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 ILP for the SPRING problem is as follows:

$$\text{(SPRING)} \quad \text{minimize} \quad \sum_{p \in Pm} y_p \tag{1}$$

subject to:

$$\sum_{c \in C(p)} x_{pc} = 1 \quad \forall p \in Pm \cup \{newP\} \tag{2}$$

$$x_{pc} - \omega_{pc} \le y_p \quad \forall p \in Pm, c \in C(p) \tag{3}$$

$$\sum_{p \in P(e)} \sum_{c \in C(p)} \delta_{cs} \cdot x_{pc} \le \eta_{es} \quad \forall e \in E, s \in S \tag{4}$$

$$\sum_{c' \in C(p')} c' \cdot x_{p'c'} - \sum_{c \in C(p)} c \cdot x_{pc} < \beta_{pp'} \cdot |S| \quad \forall p \in Pm,\ p' \in P(p) \tag{5}$$

$$\sum_{c' \in C(p')} c' \cdot x_{p'c'} - \sum_{c \in C(p)} c \cdot x_{pc} > (\beta_{pp'} - 1) \cdot |S| \quad \forall p \in Pm,\ p' \in P(p) \tag{6}$$

The objective function (1) minimizes the number of LSPs that need to be reallocated in the spectrum so *newP* can be served. Constraint (2) ensures that every candidate LSP and the received request for *newP* have one slot allocated. Constraint (3) stores whether LSP p needs to be shifted comparing current and assigned slots. Constraint (4) guarantees that each frequency slice in every optical link is assigned to one path at most. Constraints (5) and (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 to use the heuristic algorithm described in Table 1.

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 to *s* (s^+) and the set of LSPs allocated using *s* (lines 3-10 in Table 1).

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 LPSs is stored as the best solution and the number of LSPs involved is updated (lines 12-17). The best solution found is eventually returned.

Table 1. Algorithm for the SPRING problem

INPUT: E, n_d
OUTPUT: *Solution*

```
1:  Solution ← Ø
2:  numLSPs ← INFINITE
3:  for s = 1..|S| do
4:      P+ ← Ø, P- ← Ø, Ps ← Ø
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
```

In the next section, we describe our proposal to integrate SPRING into the control plane using the ABNO architecture.

4 DESCRIPTION OF THE PROPOSAL

In this section we present our proposal for provisioning-triggered hitless defragmentation. Firstly, we describe the complete distributed algorithm and identify the different steps that need to be performed, together with the ABNO module in charge. Second, we define the specific contents and semantic of PCReq and PCRep messages to deal with defragmentation. We assume that a back-end PCE is used to solve optimization problems, so the SPRING algorithm is implemented in that back-end PCE and can be invoked from the front-end PCE using a specific code carried in OF objects.

4.1 Proposed distributed algorithm for provisioning-triggered defragmentation

The workflow for provisioning-triggered defragmentation was introduced in Fig.3. In this section we focus on the steps performed once a request arrives at the front-end PCE (step 3 in Fig.3). From that point on, the distributed algorithm shown in Fig.4 is executed. To facilitate the explanation, steps in Fig.4 relate to those in Fig.3.

After a request arrives at the front-end PCE, 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 (4), the ABNO controller decides to perform a defragmentation process and sends a PCReq message to the PCE (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 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 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).

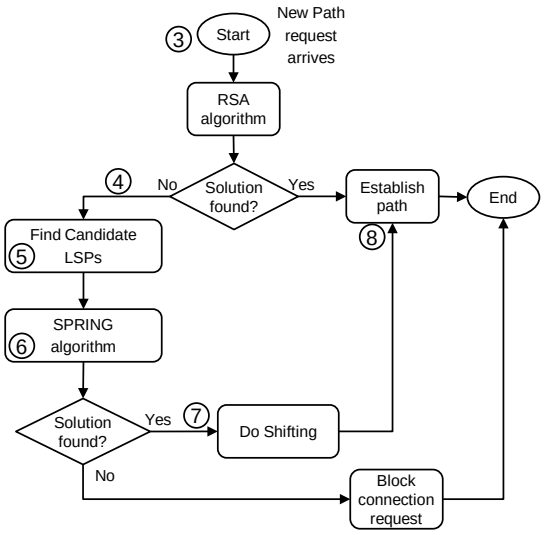

Fig.4 Defragmentation distributed algorithm

Table 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:  m ← computeSlotSize(R, bitrate, sliceWidth)
4:  for each e ∈ E(newP) do
5:      if freeSlices(e) < m then
6:          return {Ø, Ø}
7:      P ← P ∪ getLSPs(LSP-DB, e)
8:  return {P, newP}
```

Once the set of candidate LSP is found, the PCE sends a PCReq message to the back-end PCE to solve the optimization problem (6). When the back-end PCE ends, it sends back a PCRep message detailing the solution found. The front-end PCE 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 PCC of a LSP that needs to be shifted (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 (8).

4.2 PCEP issues

Owing to the input parameters needed to solve SPRING, the algorithm in Table 2 finds the route and the set of candidate LSPs. That input data has to be coded and sent in a PCReq message; Table 3 shows the specific format that we use to convey the parameters.

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 back-end PCE 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 4 shows the specific format that we use to convey the solution.

All the received requests have to be replied. To that end, the `response-list` contains an `ERO` object of each of the re-optimization requests. The route in the `ERO` must be invariant, since no rerouting is performed, however, the spectrum allocation can be either the same or a different one. Hence, an algorithm needs to be used in the front-end PCE to find the LSPs to be updated; that algorithm is detailed in Table 5.

Table 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>
```

Table 4. PCRep message contents

```
   <PCRep Message> ::= <Common Header>
                       <response-list>
   where:
      <response-list>::=<response>[<response-list>]
      <response>::=<RP>
                    <NO-PATH> | <ERO>
```

Table 5 Procedure Do_Shifting

INPUT *LSP_DB, P*

1: **for each** *p* in *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 in Spain, and Pisa (Italy). The next section describes the scenario and the tests performed.

5 EXPERIMENTAL ASSESSMENT

In this section, we experimentally validate our proposal for in-operation spectrum defragmentation. We start with a brief description of the deployed distributed set-up including the implemented modules. Next, the set-up is used to run the experiment in Fig.1. Protocol captures show the exchanged PCEP messages.

5.1 Scenario

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

The Telefonica's ABNO controller [20] has been developed in Java and supports multiple workflows (e.g. connection set-up or re-optimization); the specific workflow to be executed is defined in the incoming request. As stated in section 2, no standard north-bound interface with the ABNO controller is defined. Our implementation is based on a representational state transfer (REST) interface with the following parameters:

- *Operation ID*: uniquely identifies each user operation. The value is used to correlate different operations within the ABNO controller.
- *Operation Type*: maps the workflow to be executed by the ABNO controller.
- *Source and Destination Node*: IP addresses for the source and destination nodes.
- *Bandwidth*: Requested connection bitrate.

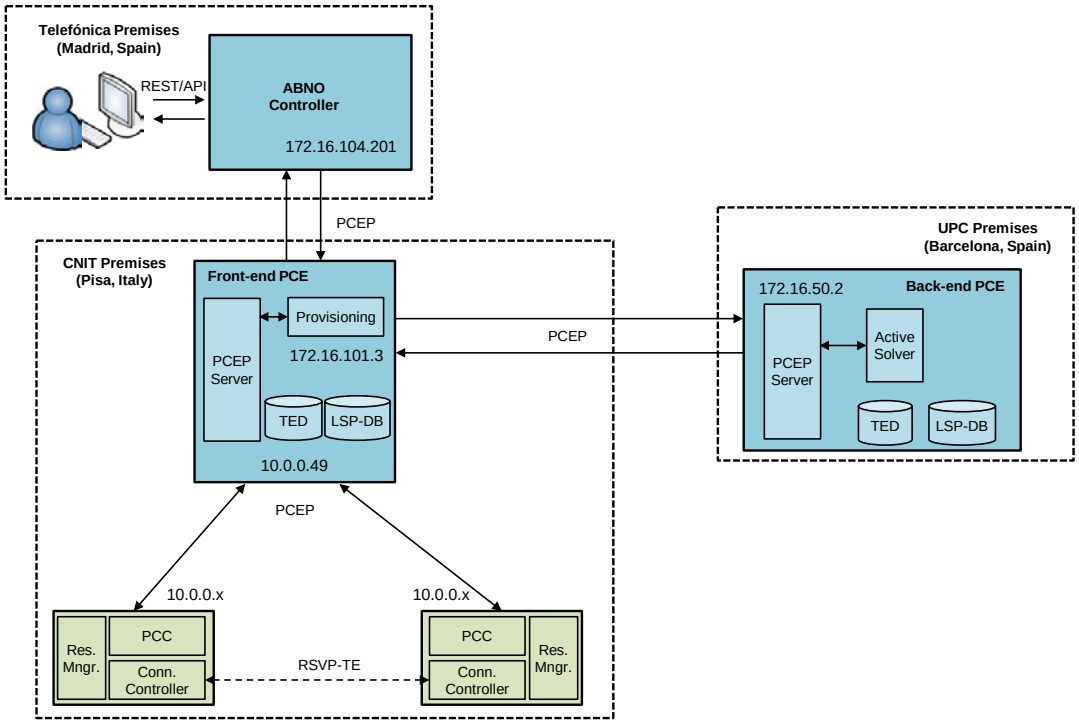

Fig. 5. Distributed test-bed set-up. IP addresses used in each interface are shown.

The CNIT front-end PCE has been implemented in C/C++ for Linux, extended to support active stateful path computation (i.e., PCUpd, PCRpt, and PCInit messages), with respect to previous versions focused on single-domain multi-format impairment-aware [21] and inter-domain path computation [22], respectively. The front-end PCE includes a PCEP Server module, a Path Solver module and the databases (i.e., the TED and the LSP-DB). The TED and the LSP-DB are kept updated by means of PCNtf and PCRpt messages sent by LSP source nodes. The PCEP Server 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 2.

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 [9]. GMPLS controllers communicate with the front-end PCE 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, the UPC's PLATON back-end PCE [23] has been developed in C++ for Linux operating system and is organized into 3 main building blocks. The first block is responsible for managing communications with other ABNO modules using standard protocols. The second block, the controller, manages PLATON execution: when incoming messages arrive, the controller decides the action to be taken among updating either the local network topology or the state of network resources, and running optimization algorithms. The last block handles optimization algorithms, which are deployed as dynamically linked libraries to allow that third party algorithms can be easily added into PLATON.

5.2 Experimental results

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.1 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. 6. For the sake of easily follow the subsequent discussion, messages are correlated to the steps described in the workflow in Fig.3. The capture, done at the front-end PCE, includes PCEP messages exchanged inside the ABNO architecture (IP sub-network 172.16.x.x) as well as those exchanges between the front-end PCE and the PCCs (IP sub-network 10.10.x.x).

	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. 6. PCEP messages exchanged.

Fig. 7 presents the contents of the PCEP messages exchanged between the ABNO controller and the front-end PCE. PCReq message 3 includes the IP address of the end nodes and the requested bitrate (50Gb/s). When no solution is found in the front-end PCE for the request, the PCRep message 4 including a `NO-PATH` object is sent back to the ABNO controller.

Next, PCReq messages 5 includes an `OF` object with the objective function code “defragmentation”. When the defragmentation process finishes, the front-end PCE 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).

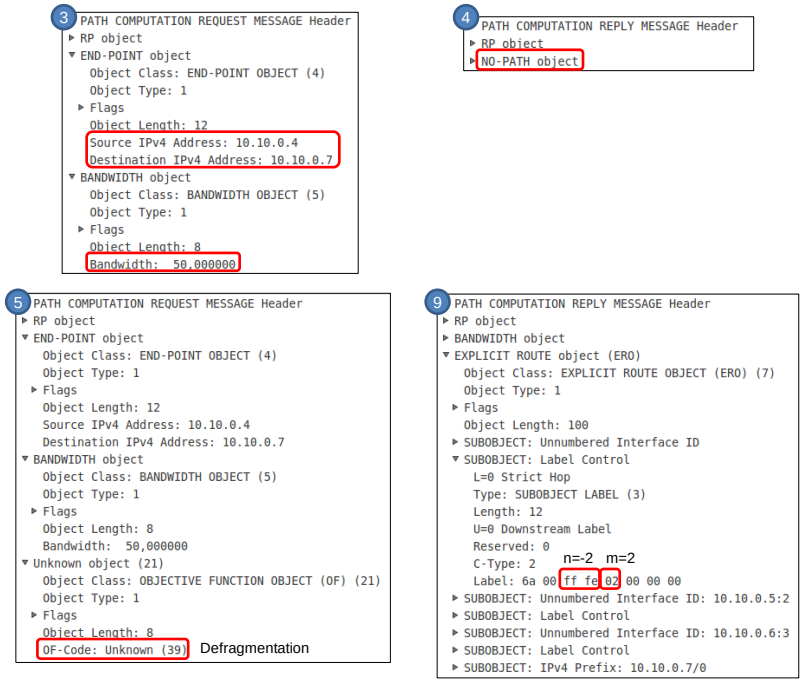

Fig. 7. PCEP messages exchanged between the ABNO controller and the front-end PCE.

Finally, Fig. 8 shows the PCEP messages exchanged between front-end and back-end PCEs. The front-end PCE 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. 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 back-end PCE sends back to the front-end PCE the PCRep message 6 shown in Fig. 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.

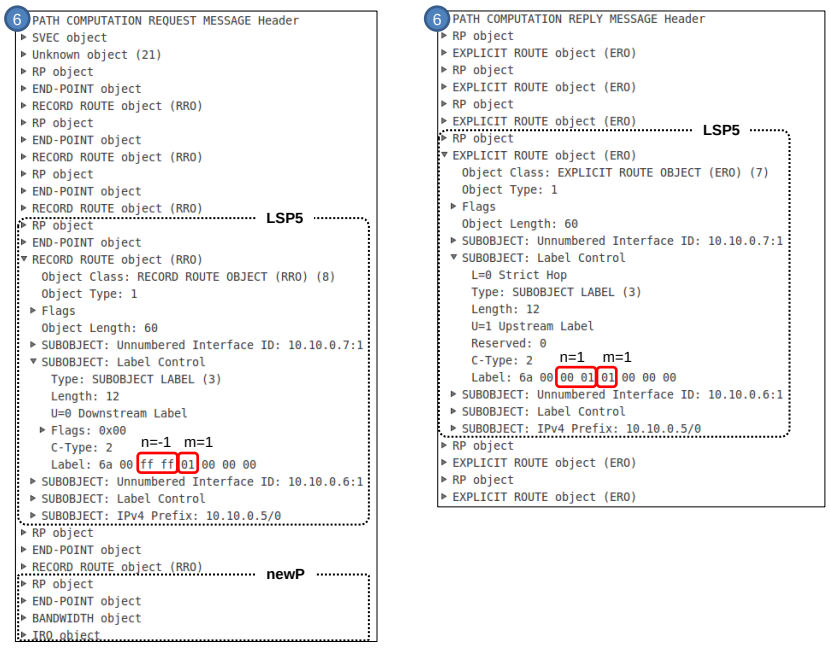

Fig. 8. PCEP messages exchanged between front-end and back-end PCE at step 6.

Once the front-end PCE 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. 6 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.

6 CONCLUSIONS AND OPEN ISSUES

In this work, the ability of ABNO architecture to deal with the defragmentation use case, which arises when operating dynamically flexgrid networks, has been experimentally assessed. To that end, the provisioning-triggered spectrum defragmentation use case was firstly investigated. The use case, an example of in-operation planning, starts when a connection request cannot be served as a consequence of spectrum fragmentation in the links of the network. As subset of already established connections are candidate to be part of the defragmentation problem, where it is possible that, by shifting the central frequency of some of them, enough room is made to serve the new connection requested. Aiming at evaluating the feasibility of the network control plane to deal with such use case, ABNO, the latest architecture of control plane currently under standardization in the IETF, has been evaluated. In particular, a front-end/back-end PCE architecture was considered. The defragmentation use case was modeled as a workflow and the relation among ABNO modules was examined. Specifically, the possibility of using the standard PCEP protocol to convey complex requests and responses between front-end to back-end PCEs, was analyzed.

Next, the spectrum shifting (SPRING) problem was formally stated and modelled using an ILP formulation, based on previous spectrum defragmentation works and adding specific constraints to limit connection reallocations where connections can be reallocated in any part of the optical spectrum, to just central frequency shifting that can be done in a hitless manner. An algorithm to be run within the back-end PCE was also proposed.

Our proposal for provisioning-triggered defragmentation was presented afterwards. The proposed workflow, in the form of distributed algorithm running in several modules of the ABNO architecture, was detailed and the PCEP messages exchanged between front-end and back-end PCEs were specified.

Finally, the feasibility of the ABNO architecture to deal with the defragmentation use case was experimentally demonstrated on a distributed test-bed connecting Telefonica's premises in Madrid, CNIT's premises in Pisa, and UPC's premises in Barcelona. The relevant PCEP messages were shown and its contents analyzed.

It is worth noting that in a distributed PCE architecture such that the one proposed in this work, link-state and traffic engineering information stored in PCE databases need to be synchronized. To that end, the BGP protocol is currently being extended within the IETF.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 317999 IDEALIST project.

REFERENCES

[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 Commun Mag., vol. 47, pp. 66-73, 2009.

[2] O. Gerstel, M. Jinno, A. Lord, S. Ben Yoo, "Elastic Optical Networking: A New Dawn for the Optical Layer?" IEEE Commun Mag., vol. 50, pp. s12-s20, 2012.

[3] L. Velasco, M. Klinkowski, M. Ruiz, and J. Comellas, "Modeling the Routing and Spectrum Allocation Problem for Flexgrid Optical Networks," Springer Photonic Network Communications, vol. 24, pp. 177-186, 2012.

[4] 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.

[5] 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.

[6] 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", Comp. Networks, vol. 56, pp. 2869-2883, 2012.

[7] 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.

[8] Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF RFC 3209, 2001.

[9] 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.

[10] D. King and A. Farrel, "A PCE-based Architecture for Application-based Network Operations", IETF draft, work in progress, 2013.

[11] R. Casellas, R. Muñoz, R. Martínez, R. Vilalta, "Applications and Status of Path Computation Elements [Invited]," IEEE/OSA Journal of Optical Communications and Networking, vol. 5, pp. A192-A203, 2013.

[12] 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.

[13] Y. Lee, JL. Le Roux, D. King, E. Oki, "Path Computation Element Communication Protocol (PCEP) Requirements and Protocol Extensions in Support of Global Concurrent Optimization," IETF RFC 5557, 2009.

[14] A. Castro, R. Martínez, R. Casellas, L. Velasco, R. Muñoz, R. Vilalta, 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.

[15] L. Velasco, D. King, O. Gerstel, R. Casellas, A. Castro, and V. López, "In-Operation Network Planning," IEEE Communications Magazine, vol. 51, pp. 52-60, 2014.

[16] E. Crabbe, J. Medved, I. Minei, R. Varga, "PCEP Extensions for Stateful PCE," IETF draft, work in progress, 2013.

[17] OpenFlow. http://www.openflow.org

[18] E. Crabbe, I. Minei, S. Sivabalan, R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model," IETF draft, work in progress, 2013.

[19] O. Gonzalez de Dios, R. Casellas, F. Zhang, X. Fu, D. Ceccarelli, I. Hussain, "Framework and Requirements for GMPLS based control of Flexi-grid DWDM networks," IETF draft, work in progress, 2013.

[20] A. Aguado, V. López J. Marhuenda, Ó. González de Dios, and J. P. Fernández-Palacios: "ABNO: a feasible SDN approach for multi-vendor IP and optical networks," in Proc. OSA Optical Fiber Conference (OFC), 2014.

[21] G. Meloni, F. Paolucci, N. Sambo, F. Cugini, M. Secondini, L. Gerardi, L Poti, P. Castoldi, "PCE Architecture for Flexible WSON enabling Dynamic Rerouting with Modulation Format Adaptation, " in Proc. European Conference on Optical Communication (ECOC), 2011.

[22] F. Paolucci, O. Gonzalez de Dios, R. Casellas, S. Duhovnikov, P. Castoldi, R. Munoz, R. Martinez, "Experimenting Hierarchical PCE Architecture in a Distributed Multi-platform Control Palne Testbed", in Proc. OSA Optical Fiber Communication Conference (OFC), 2012.

[23] 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.