Application-aware Service Provisioning and Restoration in SDN-based Multi-layer Transport Networks[☆]

Marco Savi[a,*], Domenico Siracusa[a]

[a]*Fondazione Bruno Kessler (FBK), CREATE-NET Research Center, Via alla Cascata 56/D Povo, Trento, Italy*

Abstract

Future 5G networks will run into the massive spread of applications with diverse and stringent requirements in terms of latency, reliability and security. Current multi-layer transport networks lack coordination between their constituent IP/MPLS and optical layers, and thus cannot provide to applications a full service differentiation all the way down to the optical layer. This limitation can be overcome by the adoption of a Software-Defined Network (SDN) orchestrator between the underlying multi-layer transport network and the applications, with the aim of offering them tailored network connectivity and minimizing network resources' consumption. The orchestrator must thus be able to dynamically compute multi-layer paths and allocate optical and IP/MPLS resources in an application-aware fashion (i.e, to meet applications' requirements), both when applications require service provisioning for the first time and when their connectivity must be restored after any network failure. In this paper, we propose a novel auxiliary-graph-based application-aware service provisioning algorithm, able to dynamically provision applications' service requests at runtime according to their needs. Simulations show that, unlike a benchmark application-unaware algorithm, it prevents the violation of application requirements (in terms of bandwidth, latency and availability) while guaranteeing a similar blocking probability. Following the same principles, we then propose an application-aware restoration algorithm, where fine-grained application traffic flows can be individually restored according to their tolerance to service disruption: a faster IP/MPLS restoration is considered more appropriate for applications with low tolerance, while slower multi-layer restoration is chosen for the others. Also this approach is proved to always comply with application requirements, and it can successfully restore much more application traffic flows than simple optical restoration.

Keywords: Application-centric Networking, Transport SDN, Network

[☆]A preliminary version of this paper appeared in M. Savi, F. Pederzolli, D. Siracusa, An Application-Aware Multi-Layer Service Provisioning Algorithm based on Auxiliary Graphs, in Optical Fiber Communication Conference (OFC), Mar. 2017

[*]Corresponding author. *Email address:* m.savi@fbk.eu, *Tel:* +390461312478

Orchestration, Dynamic Service Provisioning, Auxiliary Graphs, Restoration

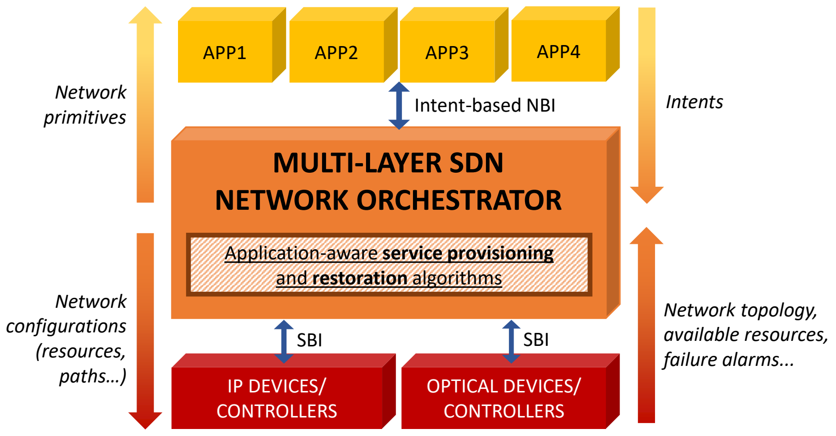
1. Introduction

Modern Internet traffic is generated by a multitude of different applications (e.g., video streaming, online gaming, financial transactions, etc.), each with its own requirements in terms of bandwidth, latency, reliability etc. This Internet traffic heterogeneity is expected to further increase in the future *5G era* [1] with the proliferation of innovative applications with very stringent latency and reliability requirements, such as *tactile Internet* applications [2] (e.g. remote surgery, exoskeleton control, etc.) or *autonomous driving* [3]. Future 5G networks are expected to go beyond the simple empowering of radio interfaces, thus requiring the definition of a new end-to-end system able to differentiate applications' traffic over multiple segments of the networks [4]. Specifically, in the transport network segment, network engineers have already tried to improve service differentiation by designing advanced traffic engineering techniques at IP/Multi Protocol Label Switching (MPLS) packet layer. However, such effort is severely limited by the fact that application traffic with diverse requirements is eventually groomed into the same few large optical connections and treated in the same way. For this reason a more comprehensive approach, able to achieve full service differentiation for applications all the way down to the optical layer, is needed.

Application-centric networking is an emerging paradigm that aims at providing fully-tailored connectivity services to applications [5][6]. This can only be achieved if the network is able to *(i)* learn information about application requirements and *(ii)* steer traffic according to application needs (i.e., in an *application-aware* fashion) at both IP/MPLS and optical layers (i.e., in a *multi-layer* fashion). The former information gathering is enabled by the novel *intent-based networking* paradigm [7] according to which an application specifies, through an intent-based interface, *what* it wants from the network in terms of requirements (i.e., the *intent*) without indicating *how* the network should achieve it. The network is then responsible for configuring itself, by allocating IP/MPLS and optical resources in a convenient way that guarantees the requested service.

From an architectural point of view, one of the most promising and recently-investigated solutions to accomplish this goal consists of adopting a (logically) centralized multi-layer Software Defined Networking (SDN) *network orchestrator* [8][9]. Even though describing in detail the functionalities and supported protocols by such a network orchestrator is out the scope of this paper[1], it is important to briefly recall its high-level architecture. The orchestrator exposes an intent-based Northbound Interface (NBI) to the applications, is able to gather application-specific intents and to compute appropriate multi-layer paths based

[1]Please refer to the ACINO H2020 European project (www.acino.eu) for further details on an application-centric orchestrator implementation (Deliverable D4.3 [10])



APP1
APP2
APP3
APP4
Network primitives
Intent-based NBI
Intents
MULTI-LAYER SDN NETWORK ORCHESTRATOR
Application-aware service provisioning and restoration algorithms
Network configurations (resources, paths…)
SBI
SBI
Network topology, available resources, failure alarms…
IP DEVICES/ CONTROLLERS
OPTICAL DEVICES/ CONTROLLERS

Figure 1: Architecture of an application-aware multi-layer network orchestrator.

(i) on applications' preferences and *(ii)* on the current network status. The allocation of resources on the computed paths is then orchestrated at both optical and IP/MPLS layers through convenient Southbound Interfaces (SBIs) controlling packet and optical network devices or controllers, as shown in Fig. 1. It is then clear that equipping the SDN orchestrator with appropriate path computation and resource allocation modules is of paramount importance to move a step towards an application-centric networking vision. In this paper, we specifically consider two different but related problems concerning these aspects.

First, we focus on dynamic service provisioning of *service requests* (i.e., intents), which arrive and leave over time and are generated by applications with heterogeneous requirements. We define a novel algorithm, called *application-aware service provisioning*, able to compute paths and jointly allocate IP/MPLS and optical resources while meeting multiple optional application requirements (in this paper we consider bandwidth, latency, availability). It exploits the concept of *auxiliary graphs* (firstly formalized in [11]), and extends the auxiliary graph construction methodology proposed in [12] to provide application-awareness to each service request. Moreover, the algorithm has a modular design, i.e., it can be easily extended to consider other application requirements.

Second, we further leverage on auxiliary graphs for the design of an *application-aware service restoration* algorithm. The algorithm allows to reactively reroute provisioned service requests on new computed paths every time they are affected by network failures (e.g. fiber cuts). It relies on two novel restoration strategies, called *application-aware IP/MPLS restoration* and *application-aware multi-layer restoration*: the former provides quicker reaction times (no optical connections are altered), while the latter is able to repair a larger set of failures at the expense of establishing new optical resources, which is a slow process. The two strategies are complementary, and enable the algorithm to restore service requests with different tolerances to service disruption. Moreover, unlike today's restoration schemes (e.g. IP Fast Reroute [13] and optical restoration

[14]), they both are *fine-grained*, i.e., able to restore each application traffic flow individually rather than restoring all the affected traffic on the same path.

We ran extensive simulations to evaluate our application-aware service provisioning and restoration algorithms. Results show that they both outperform benchmark strategies lacking application awareness at no considerable additional costs in terms of used optical and IP/MPLS resources. In fact, they are able to better distribute the application traffic flows according to application needs, thus improving network performance from a service differentiation perspective.

The remainder of this paper is organized as follows. Section 2 provides an overview of the related work. Section 3 formalizes the application-aware service provisioning and restoration problems and describes our auxiliary graph model. Sections 4 and 5 detail our auxiliary-graph-based service provisioning and restoration algorithms to solve the related problems. They also recall the application-unaware benchmark strategies for performance assessment of our approaches, which is performed in Section 6. Eventually, Section 7 concludes the paper.

2. Related Work

Our work moves a step towards the concept of *(i)* application-centric networking, with the definition of novel application-aware algorithms for both *(ii)* dynamic service provisioning and *(iii)* restoration in single and multi-layer networks. In the following, we recall the related work and our advancement on each of these topics.

2.1. Application-centric networking

In literature, *application-centric* (or *application-aware*, or, more-broadly, *QoS-aware*) networking [15][16] refers to the need of differentiating the network traffic depending on application or customer requirements. While Ref. [15] shows how some traffic differentiation can be obtained through an application-aware packet aggregation, Ref. [16] proposes a methodology to differentiate traffic grooming according to some QoS traffic classes. However, in these works applications cannot explicitly specify their requirements to the network, and their solutions cannot thus be fully considered as application-centric.

With the identification of the SDN paradigm as key enabler of application-centric networking, many works have dealt with the definition SDN-based architectures towards this direction. Ref. [17] proposes an architecture for SDN application-aware resource management, while Refs. [18][19] specify how an SDN controller can collect network and application statistics to increase networks' application awareness. Specifically, Ref. [18] presents a framework that enables application classification by adopting machine learning techniques, while Ref. [19] shows how, by reactively collecting information through deep packet inspection and direct input from applications, the SDN controller can enhance the user experience.

However, these works focus on SDN control in a single-layer scenario. Refs. [8][9] move a step further, facing the problem of designing an application-aware *multi-layer* orchestrator. Ref. [8] introduces the general concept of multi-layer orchestration in application-centric networking, while Ref. [9] proposes a preliminary architecture for such an orchestrator. In this paper, we leverage on the same principles and architecture of Refs. [8][9], but we focus on two aspects that have not been thoroughly investigated yet: how to provide application-aware service provisioning and restoration paths to application-generated traffic and how to allocate resources in the multi-layer network.

2.2. Dynamic service provisioning in single and multi-layer networks

The problem of dynamically provisioning connectivity requests while taking into account the multi-layer nature of transport networks has already been investigated in literature. Ref. [20] proposes an algorithm for efficient grooming of traffic flows in IP-over-WDM multi-layer networks, showing that it outperforms conventional shortest path algorithms in terms of blocking probability. However, they consider a fixed IP topology, while in our work the IP topology can be dynamically updated by adding new optical resources. Ref. [21] proposes a path computation algorithm that favors the accommodation of connection requests at IP layer, which is more dynamic, over the establishment of new optical resources: our approach follows the same principle. Ref. [22] describes an algorithm for path provisioning in multi-layer networks that takes into account both network and cloud resources. Even though our work only deals with network resources, it can be extended towards the same direction.

Refs. [12][23] propose instead two dynamic algorithms for virtual network mapping onto a multi-layer physical network substrate. Such works focus on the problem of Virtual Network Embedding (VNE), which relies on two subproblems, i.e., link and node mapping. Specifically, the link mapping subproblem is analogous to the end-to-end provisioning problem we want to solve. In our work, we exploit the same methodology of Ref. [12], by exploiting a similar auxiliary graph model that can be manipulated to guarantee application awareness.

While all these works do not consider any service differentiation in the connectivity provisioning, some other works take instead into account this aspect and pave the way towards an application centric vision. Ref. [24] proposes a dynamic and multi-layer resource allocation algorithm able to guarantee differentiated end-to-end QoS to various connection classes. Our approach is similar, but we consider a finer differentiation, i.e., on a per-application basis. Ref. [25] proposes an adaptive QoS-aware routing scheme based on a reinforcement learning technique. The authors consider multiple requirements such as loss, delay and throughput, and design a distributed multi-layer hierarchical control plane to minimize the signaling delay. However, in their implementation they eventually focus on a single-layer scenario, while our approach is inherently multi-layer. Ref. [26] proposes an algorithm for application-driven service provisioning but only considers *(i)* bandwidth and latency requirements, and *(ii)* a single (i.e., packet) layer. Our work moves a step forward by considering an additional requirement (i.e., availability) and focusing on a multi-layer scenario.

2.3. Service restoration in single and multi-layer networks

Many restoration schemes for the reactive mitigation of service disruption due to network failures have been designed and can be found in literature. Most of the existing schemes focus on single-layer restoration, either at optical [27] or IP/MPLS [28][29] layers. However, single-layer restoration techniques have been proved to be partially ineffective, since they are not able to recognize and handle outages happening in other layers [30].

Some works have thus dealt with multi-layer restoration, where information from optical and IP/MPLS layers (e.g. failure alarms) is jointly exploited to effectively react to any kind of network outages. Specifically, Ref. [31] analyzes the needed interaction between optical and IP/MPLS layers to enable multi-layer restoration, while Ref. [32] proposes an integer linear programming model and a heuristic algorithm to solve the problem. Refs. [33][34] focus on techno-economic aspects, showing that multi-layer restoration is beneficial from a cost perspective, while Refs. [35][36] focus on architectural solutions to enable it. Ref. [37] proposes a multi-layer scheme where IP/MPLS restoration is favored over optical restoration. Our work follows the same approach.

However, none of these works guarantees that QoS is maintained for affected services after restoration. Some works have dealt with this issue both in single- and multi-layer network scenarios. Refs. [38][39] focus on QoS-based restoration in a single-layer scenario. Ref. [38] defines an optical restoration scheme that takes into account two requirements (i.e., hop count and delay), while Ref. [39] proposes a framework for VNE where QoS performance is maintained for affected virtual networks. Our approach follows similar principles, but we focus on end-to-end connectivity and on a multi-layer scenario.

Some recent works have moved a step even further, towards application-aware restoration in multi-layer networks. Ref. [40] defines a multi-layer architecture for differentiated traffic restoration, but only considers two application classes (gold and best effort). Our application-aware service restoration is instead able to restore individually (i.e., in a fine-grained way) many different application flows according to their requirements. Ref. [41] demonstrates a multi-layer solution that can orchestrate IP/MPLS and optical restoration depending on application-requested policies. The proposed solution allows applications to explicitly select the type of restoration they want, but misses the possibility of specifying other relevant requirements (such as bandwidth, latency and availability). In our work we fill this gap.

3. Problem Statement and Approach

We consider as multi-layer network a 2-layer transport network with a transparent Dense Wavelength-Division Multiplexing (DWDM) optical layer and an IP/MPLS packet layer, whose topology can be abstracted by the network orchestrator. An example of abstracted topology at a specific point in time (i.e., a snapshot of the network) is depicted in Fig. 2a. At the optical layer, nodes (i.e., reconfigurable optical add-drop multiplexers, ROADMs) are interconnected by

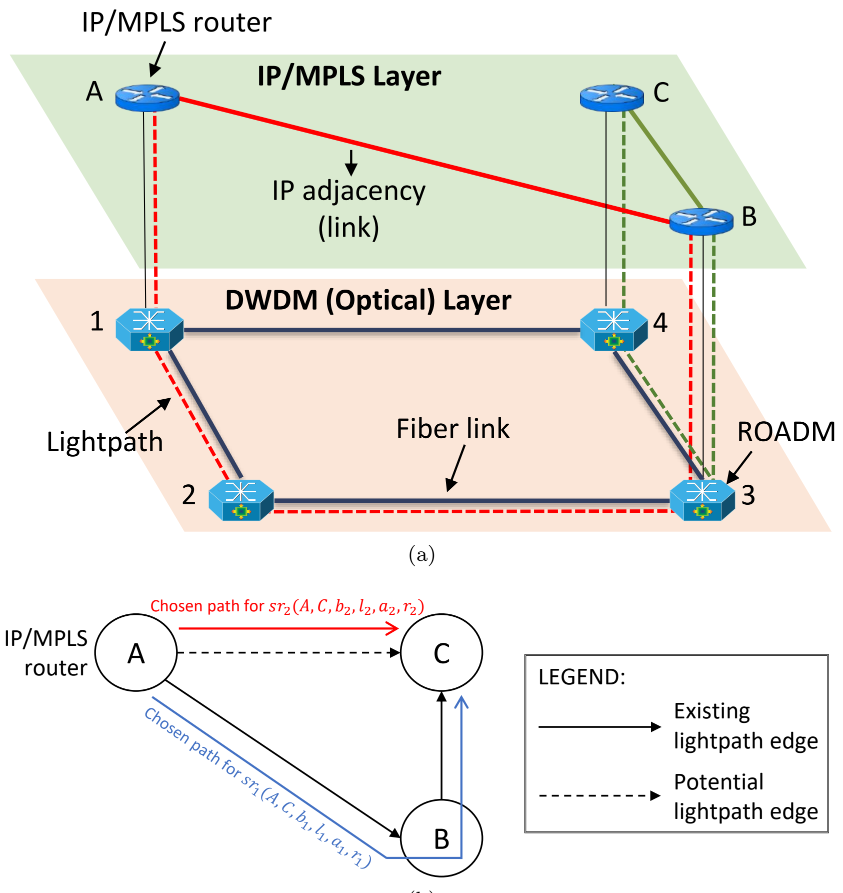


Figure 2: Example of an IP/MPLS and optical multi-layer network topology as abstracted by the orchestrator at a specific point in time (a) and example of an auxiliary graph based on such multi-layer network topology (b).

fiber links supporting multiple wavelengths, while at the IP/MPLS layer nodes (i.e., IP/MPLS routers) are dynamically interconnected by IP adjacencies (or links). Each IP adjacency is realized through a *lightpath* (i.e., a transparent optical connection) in the optical layer, meaning that a link in the IP/MPLS layer is always associated to a path in the optical layer.

Multiple service requests are generated over time by business applications running at the edge of the network (e.g. virtual machine migration or distributed database synchronization among different branches of the same enterprise), each requesting connectivity between a pair of IP/MPLS nodes. Each service request sr is represented by a $sr(s, d, b, l, a, r)$ tuple, where s and d are the IDs of source and destination IP/MPLS nodes, respectively, b is the worst-case requested bandwidth, l the maximum tolerable end-to-end latency, a the minimum tolerable path availability and r specifies, in case the service request is affected by a network failure, whether its restoration must be necessarily `fast` or can be `slow`. Note that b, l, a, r are the application requirements, and that the r requirement is associated to the *maximum service downtime* that can be tolerated by the service request: $r =$ `fast` means that the service request must be restored fast to minimize the service downtime (e.g. for live streaming applications), while in case of $r =$ `slow` the service request can tolerate a longer downtime (e.g. for smart metering applications).

In this dynamic scenario, the network orchestrator must be able to:

- Provide connectivity to an application-generated service request as soon as it is delivered to the orchestrator (i.e., in real-time) by computing an end-to-end multi-layer *application-aware path*, i.e., a path satisfying the b, l, a application requirements at the same time, and by allocating optical and IP/MPLS resources in a proper way. We refer to this problem as *application-aware service provisioning.*

- Restore a service request, whose connectivity has been affected by a network failure, on a new application-aware path that meets the b, l, a, r application requirements and circumvents the failure. We refer to this problem as *application-aware service restoration.*

Even though we only focus on b, l, a, r application requirements, a service request could require additional ones. For example, in our previous work [42], we also considered the e application requirement, according to which a service request should be provisioned on an end-to-end encrypted optical path. Without any loss of generality, in this paper we confine the investigation to a limited number of requirements. This allows a better comprehension of our proposed solutions to solve the application-aware service provisioning and restoration problems, as shown in Sections 4 and 5. Our solutions leverage the construction of *auxiliary graphs*, whose model will be described next.

3.1. Auxiliary graph model description

Auxiliary graph models [11] are the main approach for path computation and dynamic resource allocation to connection requests in multi-layer networks. In this paper, we propose an auxiliary graph model for the computation of application-aware paths that is based on the IP/MPLS-layer network topology. To construct it, all the IP/MPLS nodes are first added to the auxiliary graph. Then, two different types of edges can be added, i.e., existing and potential lightpath edges:

- An *existing lightpath* edge is added if a lightpath has been previously established between two IP/MPLS nodes. An existing lightpath already carries some network traffic and is associated to an IP adjacency.

- A *potential lightpath* edge represents whether a transparent lightpath (and, consequently, an IP adjacency) could be established between two IP/MPLS nodes given the current status of network resources in the optical layer (e.g. number of available wavelengths). When a potential lightpath edge is added to the auxiliary graph, the corresponding optical resources (i.e., a continuous wavelength) are reserved all over the path, but not allocated.

In general, which potential lightpath edges are added to the auxiliary graph depends on *(i)* the adopted routing and wavelength assignment algorithms in the optical layer and *(ii)* the potential lightpath edge addition policy, as we will point out in the following section.

Figure 2b shows an example of auxiliary graph considering the abstracted multi-layer topology of Fig. 2a. The graph consists of:

- The three IP/MPLS nodes A, B and C;
- The existing lightpath edges (A,B) and (B,C). In Fig. 2a, the corresponding lightpaths are (A,1,2,3,B) and (B,3,4,C), which are depicted in red and green dashed lines, respectively;
- A potential lightpath edge (A,C). The addition of such potential lightpath edge guarantees that the optical layer can provide a continuous wavelength to provision a new lightpath between A and C, for example on the (A,1,4,C) path.

The definition of such an auxiliary graph greatly simplifies the problem of resource allocation in multi-layer networks. First, it allows to represent a multi-layer topology on a single-layer graph, which is an augmented version of the IP/MPLS topology taking into consideration both IP/MPLS and optical layer resources. Second, the auxiliary graph can be properly manipulated to drive the selection of application-aware paths for provisioning and restoration, as shown in the next two sections.

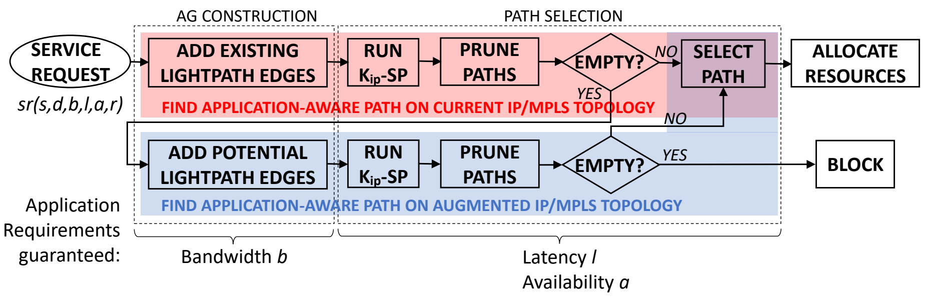
4. Application-aware Service Provisioning Algorithm

4.1. Description of the algorithm

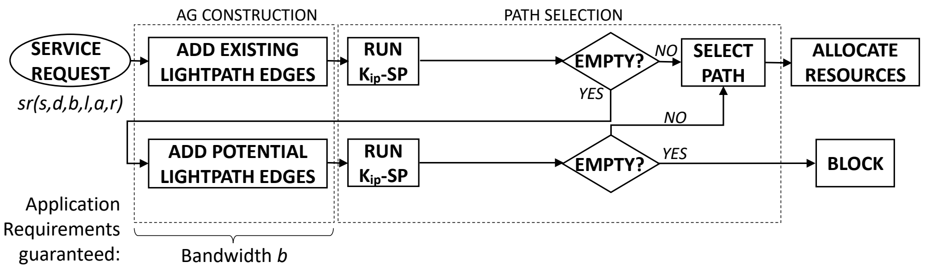
In this section we describe the auxiliary-graph-based algorithm we have designed to solve the application-aware service provisioning problem. The flow diagram of the algorithm is shown in Fig. 3a. It is executed every time a service request $sr(s, d, b, l, a, r)$ arrives and needs to be provisioned.

First, the algorithm attempts to find an application-aware path that reuses existing lightpaths. This approach avoids the consumption of new optical resources, since no new lightpath is established at this stage. Three main phases are involved in the process: *(a)* auxiliary graph construction, *(b)* path selection and *(c)* resource allocation.

In the *auxiliary graph construction* phase, an auxiliary graph including only existing lightpath edges is created. Such an auxiliary graph is an exact copy of the current IP/MPLS topology. Among all the existing lightpath edges, the ones not meeting b, i.e., with available bandwidth $B_{lp} < b$, are pruned. The


AG CONSTRUCTION
PATH SELECTION
SERVICE REQUEST
sr(s,d,b,l,a,r)
ADD EXISTING LIGHTPATH EDGES
RUN K_ip-SP
PRUNE PATHS
EMPTY?
NO
YES
SELECT PATH
ALLOCATE RESOURCES
FIND APPLICATION-AWARE PATH ON CURRENT IP/MPLS TOPOLOGY
NO
ADD POTENTIAL LIGHTPATH EDGES
RUN K_ip-SP
PRUNE PATHS
EMPTY?
YES
BLOCK
FIND APPLICATION-AWARE PATH ON AUGMENTED IP/MPLS TOPOLOGY
Application Requirements guaranteed:
Bandwidth b
Latency l
Availability a

(a)



(b)

Figure 3: *Application-aware* (a) and *baseline* [12] (b) service provisioning algorithms.

pruning guarantees that all the remaining existing lightpath edges meet the requirement b for the service request sr.

After properly constructing the auxiliary graph, the *path selection* phase is started to find a suitable application-aware path. In this phase, the K_{ip}-Shortest Path (K_{ip}-SP) algorithm between the IP/MPLS source s and destination d nodes is run on it. In principle, many different weights could be chosen for each existing lightpath edge of the auxiliary graph. In this work, we choose as weight the lightpath physical length, so that the K_{ip}-SP computed paths are increasingly ordered according to their end-to-end length. K_{ip}-SP gives as output up to K_{ip} *candidate paths*, all meeting the b application requirement. Among such paths, the algorithm prunes those with latency $L_{path} > l$ and with availability $A_{path} < a$. Note that the end-to-end path latency is calculated according to the following formula:

$$L_{path} = \sum_{node \in path} L_{node} + \sum_{link \in path} L_{link} \tag{1}$$

where L_{node} is the latency added by the traversed IP/MPLS and optical nodes (e.g. because of queuing, processing and transmission) and L_{link} is the propagation delay added by the traversed optical links. Instead, the end-to-end path availability is calculated as follows [43]:

$$A_{path} = \prod_{node \in path} A_{node} \cdot \prod_{link \in path} A_{link} \tag{2}$$

where A_{node} is the availability of the traversed IP/MPLS and optical nodes and A_{link} is the availability of the traversed optical links. Note that the availability of each component (i.e., node or link) is computed in the following way [44]:

$$A_{node/link} = \frac{MTTF_{node/link}}{MTTF_{node/link} + MTTR_{node/link}} \tag{3}$$

where $MTTF_{node/link}$ and $MTTR_{node/link}$ are the Mean Time To Fail and the Mean Time To Repair of the node or link itself. The information on the node/link estimated latency, MTTF and MTTR can either be explicitly made available to the orchestrator through IP/MPLS and optical layer SBIs or computed according to SBIs-gathered information (e.g. failure/repair alarms).

After the path pruning process, all remaining candidate paths meet the b, l, a application requirements. Among those, if any, the first one in the list (i.e., the shortest) is chosen as application-aware path for the service request sr, and the requested bandwidth b is *allocated* over the traversed existing lightpaths. An example of such a path is shown in Fig. 2b, where the service request sr_1 is provisioned over the path depicted in blue between A and C, which meets b_1, a_1, l_1 and uses the existing lightpath edges (A,B) and (B,C).

Alternatively, if the set of candidate paths is empty, the algorithm performs a second stage to augment the auxiliary graph constructed earlier by adding potential lightpath edges. As potential lightpath edge addition policy, the algorithm considers all the possible IP/MPLS node pairs of the auxiliary graph and adds a potential lightpath edge between a node pair only if *(i)* no existing lightpath edge already exists between those nodes, and *(ii)* there are enough optical resources to establish it if needed. The K_{dwdm}-SP First-Fit (FF) routing and wavelength assignment algorithm is run on the optical layer topology to check whether enough resources exist to potentially establish a lightpath and, if so, to reserve them. As link weight, the algorithm considers the length of each optical link. Note that, with the addition of potential lightpath edges to the auxiliary graph, the chance of finding an application-aware path in the *path selection* phase is increased, at the expense of having to set up one or more new lightpaths in the network. The path selection phase then exactly works as in the first stage, i.e., it computes K_{ip}-SP and prunes those paths not meeting l and a application requirements.

If the list of candidate paths is not empty, the first one is selected as application-aware path, the chosen wavelengths are *allocated* for the traversed potential lightpath edges, their state is changed from *potential* to *existing*, and lastly the requested bandwidth b is allocated over the traversed existing lightpaths. Note that, in this second stage, resources need to be allocated both at optical (i.e., wavelengths) and IP/MPLS layers (i.e., bandwidth). An example of such an application-aware path is shown in Fig. 2b, where the service request sr_2 is provisioned over the red path, which meets b_2, l_2, a_2 and consists only on the potential lightpath edge (A,C). Such a service request may require a stringent $l_2 > l_1$ value that cannot be met by a path using the existing lightpath edges (A,B) and (B,C), as instead done for sr_1. The algorithm thus needs to augment the auxiliary graph with the (A,C) potential lightpath edge to find a suitable solution. The reserved wavelength is then allocated on the A,1,4,C path (Fig. 2a) and the service request is provisioned.

Otherwise, if the set of candidate paths is still empty, the service request is blocked, because there are not enough resources either at optical and IP/MPLS layers to provision it.

One of the advantages of our provisioning algorithm is its *modularity*, since additional application requirements can be included and evaluated by properly modifying either the auxiliary graph construction phase, or the path selection phase, or both. For example, if a service request requires end-to-end optical encryption e, as recalled in Section 3, the auxiliary graph construction phase can be modified to prune all the existing lightpath edges that both are not encrypted and end-to-end. This has been evaluated in our previous work [42].

4.2. Computational complexity analysis

We express the algorithm complexity in terms of number of calls to K-SP, which can be implemented as a polynomial-time algorithm [45]. In the worst case, K-SP is run once in the first phase of the algorithm, i.e., while searching paths on the auxiliary graph with only existing lightpaths added. Then, it runs up to $N(N-1)$ times for potential lightpaths addition to the auxiliary graph (where N is the number of IP/MPLS routers). Finally, K-SP is run once again to find an application-aware path on the augmented auxiliary graph. This means that the overall complexity of the algorithm is $O(N^2)$ K-SP calls (i.e., polynomial-time complexity).

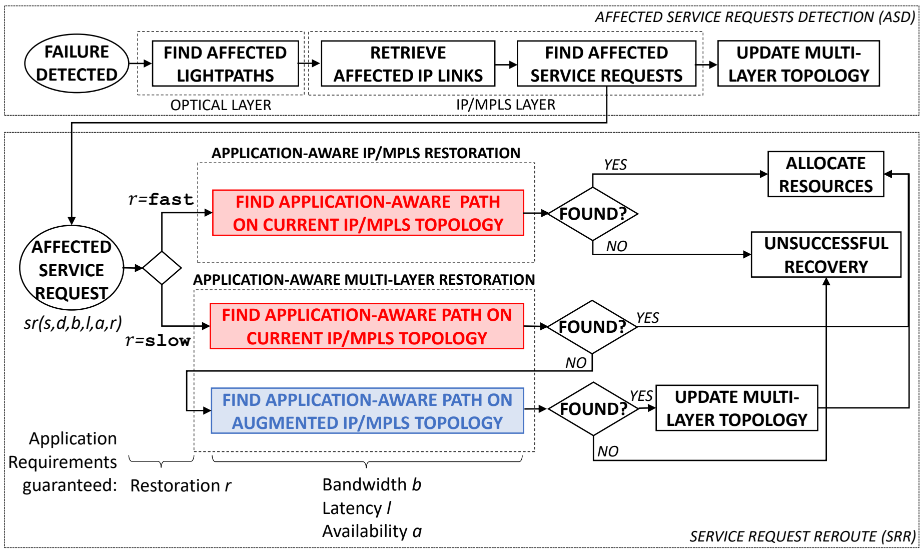
4.3. Description of the benchmark service provisioning algorithm

To evaluate the effectiveness of our application-aware service provisioning algorithm, we compare it with a benchmark *baseline provisioning* algorithm. The baseline provisioning algorithm is built upon the link mapping algorithm proposed in [12]. As shown in Fig. 3b, the baseline provisioning algorithm is a simplified version of our proposed provisioning algorithm lacking application awareness. It works in two stages and tries to provision a service request by using existing lightpath edges first and by only considering the bandwidth b application requirement. This means that, unlike our application-aware algorithm, it does not perform any path pruning, and thus l and a requirements cannot be guaranteed. A comparison between application-aware service provisioning and baseline algorithms allows to fully disclose the benefits of our approach with respect to the current best practices, as we will show in Section 6.

5. Application-aware Service Restoration Algorithm

5.1. Description of the algorithm

This section shows how the auxiliary graph model presented in Section 3 can be exploited for a fine-grained application-aware restoration of service requests once they are affected by network failures. Our application-aware restoration



AFFECTED SERVICE REQUESTS DETECTION (ASD)
FAILURE DETECTED
FIND AFFECTED LIGHTPATHS
RETRIEVE AFFECTED IP LINKS
FIND AFFECTED SERVICE REQUESTS
UPDATE MULTI-LAYER TOPOLOGY
OPTICAL LAYER
IP/MPLS LAYER
APPLICATION-AWARE IP/MPLS RESTORATION
r=fast
FIND APPLICATION-AWARE PATH ON CURRENT IP/MPLS TOPOLOGY
FOUND?
YES
NO
ALLOCATE RESOURCES
UNSUCCESSFUL RECOVERY
AFFECTED SERVICE REQUEST
sr(s,d,b,l,a,r)
APPLICATION-AWARE MULTI-LAYER RESTORATION
r=slow
FIND APPLICATION-AWARE PATH ON CURRENT IP/MPLS TOPOLOGY
FOUND?
YES
NO
FIND APPLICATION-AWARE PATH ON AUGMENTED IP/MPLS TOPOLOGY
FOUND?
YES
NO
UPDATE MULTI-LAYER TOPOLOGY
Application Requirements guaranteed:
Restoration r
Bandwidth b
Latency l
Availability a
SERVICE REQUEST REROUTE (SRR)

Figure 4: *Application-aware* service restoration algorithm.

algorithm is shown in Fig. 4 and consists of two phases: an *Affected Service Requests Detection* (ASD) and a *Service Request Reroute* (SRR) phase.

The ASD phase aims at finding the service requests affected by a failure when it happens. Note that in this paper we focus on *bidirectional optical link failures* (e.g. fiber cuts), but the concept can be easily extended to other kinds of link and/or node failures. When a failure occurs, the algorithm detects all the *affected lightpaths*, i.e., the lightpaths that have been tiered down by such failure since they traverse a failed optical link. The algorithm then retrieves the corresponding *affected IP links* at the IP/MPLS layer. From this information it is possible to get the list of *affected service requests*. An affected service request has been previously provisioned on an application-aware path crossing one of the affected IP links. The ASD phase terminates by updating the multi-layer topology, as stored by the orchestrator, by removing the failed optical and affected IP links, since they cannot be used anymore.

The SRR phase aims at restoring individually each affected service request from the list. The type of restoration is chosen according to the application requirement r. If $r = \texttt{fast}$, the network opts for *application-aware IP/MPLS restoration*, otherwise it opts for *application-aware multi-layer restoration*. The rationale behind this choice is that application-aware IP/MPLS restoration reroutes the service request at the IP/MPLS layer, which is a fast process taking no more than few hundreds of milliseconds. Application-aware multi-layer restoration may instead require the setup of new lightpaths in the network, which is a slow process taking tens of seconds/minutes [31].

When *application-aware IP/MPLS restoration* is chosen, a new path is searched at the IP/MPLS layer to circumvent the failed optical and affected IP links. An auxiliary graph with only existing lightpath edges is created and a path pruning process is performed to find an application-aware path meeting b, l, a application requirements (red boxes in Figs. 3 and 4). If no path can be found, it is impossible to meet the r application requirement and the service request recovery fails, otherwise IP/MPLS resources are allocated on the new application-aware path to restore the service. If the request recovery fails, resources are released all over the corrupted path, to make them available for future requests. Even though this restoration strategy relies on available resources at IP/MPLS layer and is then a *single-layer* approach, the ASD phase updates the multi-layer topology by taking into account the failure propagation across optical and IP/MPLS layers. This solves the issue of current IP/MPLS restoration approaches [29], which cannot rely on any information about optical layer failures.

When *application-aware multi-layer restoration* is chosen, a new application-aware path is searched at the IP/MPLS layer first, as it happens for application-aware IP/MPLS restoration. If found, the service request is rerouted on the path. This guarantees that *(i)* the service is restored fast even though $r = \texttt{slow}$ (good for the application) and *(ii)* already-established IP/MPLS resources are reused (good for the network). If no path is found, an application-aware path is then searched on an auxiliary graph augmented with potential lightpath edges, by adopting the same procedure shown in the blue box of Fig. 3. If a path is found, the multi-layer topology is updated including the added lightpath(s), flagged as *establishing.* These lightpaths can be used to reroute other affected service requests with $r = \texttt{slow}$, but not with $r = \texttt{fast}$, since they are still being established and cannot guarantee fast service request recovery. The IP/MPLS and optical resources are then allocated, while if no application-aware path is found the service request recovery fails, and resources are released all over the corrupted path. This mechanism is inherently *multi-layer*, since it involves both optical and IP/MPLS layers in the whole restoration process [34]. Note also that reusing lightpaths that are being established for the provisioning of service requests with $r = \texttt{slow}$ allows to keep the number of newly-set-up lightpaths low in the overall restoration process.

The performance of our restoration algorithm, in terms of number of successfully-restored service requests, depends on the order in which the affected service requests are restored. The best strategy is restoring first all the affected service requests with $r = \texttt{fast}$. Prioritizing the service requests with $r = \texttt{fast}$ guarantees that the spare IP/MPLS resources are first re-used for the requests that actually need fast restoration, thus increasing the chances of successfully restoring them on the limited and already-established IP/MPLS resources. Then, if any spare IP/MPLS resources are left, they can be used to provide fast restoration to service requests with $r = \texttt{slow}$, before establishing new lightpaths. Our algorithm guarantees this prioritization by properly sorting the list of affected service requests at the end of the ASD phase.

Note that our restoration algorithm is fully *reactive*, i.e., no resources is pre-planned for backup paths before that restoration is triggered by a failure. Pre-planning of resources to increase availability and further reduce service downtime has been tackled in our previous work [46].

	Requirements				
Type of restoration	b	l	a	Restoration time	Fine-grained?
Application-aware IP/MPLS	✓	✓	✓	Fast (ms)	Yes
Application-aware multi-layer	✓	✓	✓	Slow (s, min)	Yes
Baseline IP/MPLS	✓			Fast (ms)	Yes
Baseline multi-layer	✓			Slow (s, min)	Yes
Optical	✓			Slow (s, min)	No

Table 1: Considered restoration strategies and their features.

5.2. Computational complexity analysis

Similarly to Section 4.2, we express the algorithm complexity in number of calls to K-SP. *Application-aware IP/MPLS restoration* requires to run K-SP once (i.e., on the auxiliary graph including only existing lightpaths), so the overall complexity of such strategy is O(1) K-SP calls. *Application-aware multi-layer restoration* instead requires to augment the auxiliary graph with potential lightpaths, so its complexity is the same of the service provisioning algorithm as shown in Section 4.2, i.e., $O(N^2)$ K-SP calls. The overall complexity of the *application-aware service restoration algorithm*, which is executed when a failure is detected, depends on the number M_{as} of affected service requests. The worst case in terms of time execution happens when all the service requests have r = `slow` and thus require multi-layer restoration. Considering this case, the overall complexity of the algorithm is $O(M_{as} \cdot N^2)$ K-SP calls.

5.3. Description of the benchmark restoration strategies

We compare our application-aware restoration algorithm with two different approaches. The first one, called *baseline restoration* algorithm, is analogous to application-aware restoration, but lacks of application awareness. Service requests with r = `fast` are restored using *baseline IP/MPLS restoration*, while service requests with r = `slow` are restored using *baseline multi-layer restoration.* In both cases, the affected service requests are restored by finding a path only meeting b and neglecting l and a, as done in Fig. 3b. The baseline IP/MPLS restoration process works similarly to existing schemes in IP/MPLS networks such as IP Fast Reroute [13], but *(i)* on a fine-grained fashion and *(ii)* by exploiting, through the ASD phase, the failure propagation information across optical and IP/MPLS layers. Baseline multi-layer restoration recalls instead well-known state-of-the-art multi-layer restoration algorithms [30][40].

The second approach is *optical restoration* [14], where all the traffic crossing a tiered-down lightpath is rerouted on a newly-set-up one. This approach is neither fine-grained, since traffic flows for multiple affected service requests are always recovered on the same path, nor application-aware, since it does not guarantee that l and a application requirements are met after restoration.

Note that application-aware multi-layer, baseline multi-layer and optical restoration all can lead to similar high service downtime (tens of seconds/minutes),

Parameter	Value
DWDM optical layer	
Number of ROADMs	30
Number of bidirectional fiber links (N_{links})	56
Number of wavelengths per fiber link	80
Wavelength capacity	100 Gb/s
IP/MPLS layer	
Number of IP/MPLS routers	14
Discrete-event simulation	
Service request generation	Poisson process
Service request holding time	Exponential
Number of service requests	100000
Number of transitory events	10000
Number of simulation runs	25
Provisioning and restoration algorithms	
K_{ip}	50
K_{dwdm}	5

Table 2: List of general simulation parameters and their values.

since require (or may require, in case of multi-layer restoration) the set up of new lightpaths. Same happens for application-aware IP/MPLS and baseline IP/MPLS restoration schemes, which both act at IP/MPLS layer and can be performed fast (few hundreds of milliseconds). A recap of the restoration strategies considered in this paper is shown in Tab. 1.

6. Performance Evaluation

6.1. General simulation settings

We implemented and simulated our proposed algorithms in Net2Plan [47]. As multi-layer topology, we used the Telefonica Spain national transport network topology. The optical layer consists of 30 ROADMs interconnected by 56 bi-directional fiber links (each one carrying up to 80 wavelengths, with a capacity of 100 Gb/s each) and 14 IP/MPLS routers [48]. We doubled the propagation delay for each fiber to consider any possible processing and queuing delay introduced by nodes connected to each fiber link. Even though this leads to a rough processing and queuing delay estimation, it ensures that any computed end-to-end path latency upper bounds the real one, since the propagation delay can be considered as dominant delay factor in modern transport networks [49]. The service requests are generated according to a Poisson process with an exponential holding time (i.e., following the M/M/∞ queuing model). Once a service request expires, the resources allocated to it are released. All the simulations start from an empty network, with no pre-provisioned lightpaths and IP links. For each simulation run, we simulated the processing of 10^5 service requests,

Parameter	Value
DWDM optical and IP/MPLS layer	
$MTTF_{node/link}$	$\{1000, 10000, 100000\}$ time units (uniform)
$MTTR_{node/link}$	1 time unit
$A_{node/link}$	$\{99.9, 99.99, 99.999\}$ % (uniform)
Service requests / Application requirements	
Traffic matrix	Uniform or TID (depends on simulation)
Requested bandwidth b	$\{1, 10, 100\}$ Gb/s (uniform)
Maximum latency l	$\{10, \infty\}$ ms (uniform)
Minimum availability a	$\{99.75, 0\}$ % (uniform)
Restoration r	Omitted

Table 3: List of parameters for service provisioning simulations.

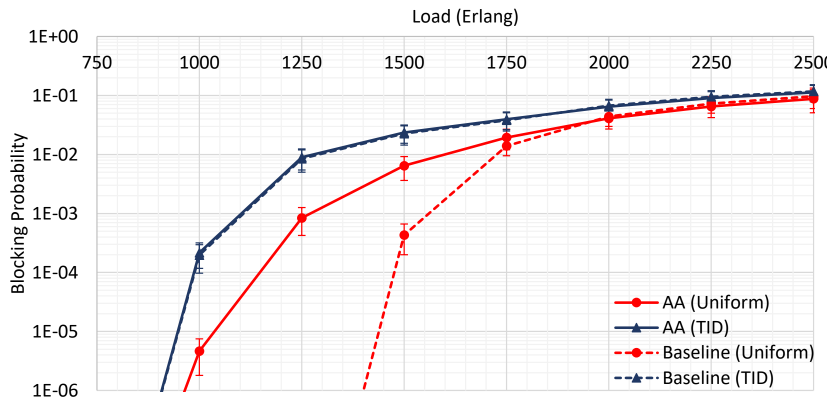


Figure 5: Blocking probability comparison of *application-aware* (AA) and *baseline* service provisioning algorithms in case of *uniform* and *TID* traffic matrices. Results are reported along with 95% confidence interval.

starting to collect statistics after the first 10^4 requests, so as to reach a desired network utilization and avoid collecting statistics during the initial transitory phase. Since many simulation parameters are randomized, we performed 25 simulation runs for each simulation setting with different random seeds, and we averaged the obtained results. Concerning the application-aware service provisioning and restoration algorithm parameters, we set $K_{dwdm} = 5$ and $K_{ip} = 50$. The complete list of general parameters, which are never changed, is reported in Tab. 2.

6.2. Evaluation of application-aware service provisioning

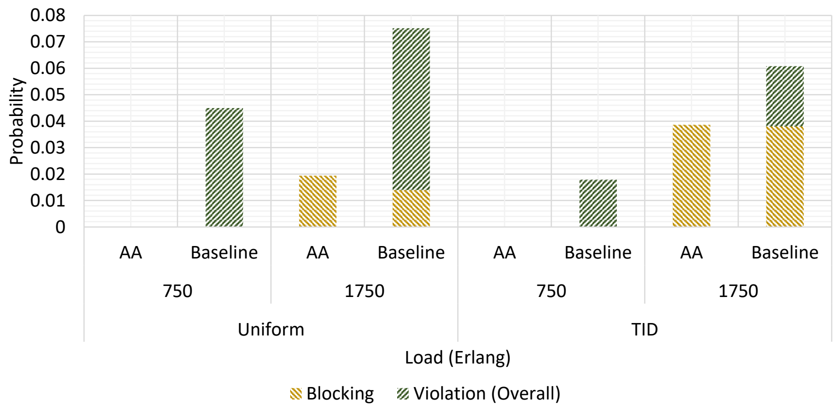
In this subsection, we focus on the evaluation of the application-aware service provisioning algorithm described in Section 4. We consider a scenario where no failures occur in the network, and thus the r application requirement can be omitted. The MTTR of ROADMs, fiber links and IP/MPLS routers is set to

$MTTR_{node/link} = 1$ time unit, while their MTTF is randomly chosen from the following set: $MTTF_{node/link} = \{1000, 10000, 100000\}$ time units. This means that, according to Eq. 3, the node/link availability is uniformly distributed among the values in the following set: $A_{node/link} = \{99.9\%, 99.99\%, 99.999\%\}$.

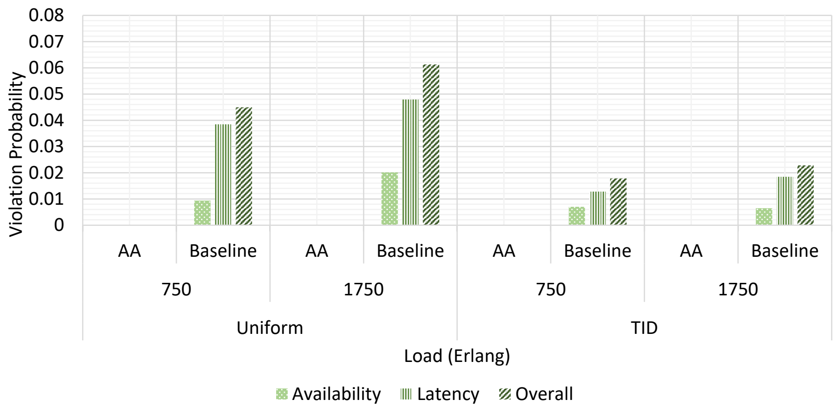
We consider two different traffic matrices: *(i)* a *uniform* traffic matrix, where the start s and destination d endpoints for each service request are uniformly distributed among all the IP/MPLS nodes and *(ii)* a realistic non-uniform traffic matrix provided by Telefonica Spain, where most of the traffic is routed from/to IP/MPLS nodes around the capital city (i.e., Telefonica Investigacion y Desarrollo (TID) traffic matrix). For each service request, the b, l, a application requirements are uniformly chosen from the following sets: $b = \{1, 10, 100\}$ Gb/s, $l = \{15, \infty\}$ ms, $a = \{0, 99.75\}$ %. $l = \infty$ ($a = 0\%$) means that the service request is not constrained by latency (availability). The list of parameters for this scenario is reported in Tab. 3.

Figure 5 shows the *blocking probability* of service requests as a function of the network load for the two different traffic matrices. The network load is expressed in terms of average number of service requests in the network (Erlang). We compare the performance of our application-aware (AA) service provisioning algorithm and of the baseline algorithm shown in Fig. 3. A uniform traffic distribution leads to lower blocking probability than the TID traffic distribution for both algorithms. This happens because, in case of TID traffic, optical and IP links around the capital city become congested faster than in case of uniform traffic, where the load is more balanced. For this reason, it is harder to find a suitable path for a service request. We also notice that, in case of uniform traffic, the application-aware algorithm leads to higher blocking than the baseline algorithm, while this does not happen in case of TID traffic, where blocking probability is comparable for the two algorithms. Generally speaking, the application-aware algorithm leads to higher blocking probabilities because it has been designed to block a service request if it is impossible to meet *all* the application requirements at the same time, while the baseline algorithm focuses on finding a path with only enough bandwidth, while neglecting the other requirements. This phenomenon is not visible in case of TID traffic matrix for the same topological reason explained above, i.e., because most of the IP links are concentrated around the capital area and thus it is harder to find alternative paths than the shortest ones.

Figures 6a and 6b focus also on another evaluation metric, i.e., *violation probability.* This metric evaluates the probability that the path provided to a service request violates one or more application requirements. For the sake of clarity, we only consider two network load values (750 and 1750 Erlang). Figure 6a shows both blocking and violation probability for the two network load values and for the two different traffic matrices. Violation probability is always zero for the application-aware algorithm. In fact, the algorithm has been inherently designed to always meet all the application requirements. Conversely, the baseline algorithm always leads to high violation probabilities (up to 6%), since it allocates resources in an application-unaware manner, meaning that it may happen that the l and a application requirements are not met by a pro-



(a)



(b)

Figure 6: Violation and blocking probability comparison (a) and violation probability breakdown (b) for *application-aware* (AA) and *baseline* service provisioning algorithms in case of *uniform* and *TID* traffic matrices.

visioned path. If we consider the sum of blocking and violation probabilities as a measure of the goodness of an algorithm (i.e., a service request is *mishandled* either if it is blocked or it does not meet all the application requirements at the same time), our application-aware algorithm significantly outperforms the baseline algorithm. Figure 6a also shows that violation probability, for the baseline algorithm, is lower in case of TID traffic than in case of uniform traffic, and it increases as the traffic load increases. In fact, considering higher traffic load conditions, provisioned paths are in average longer, raising the chance of violating either l (higher propagation delay), or a (higher number of crossed components) or both. This is well shown in Fig. 6b, which reports the breakdown of violation probability into the different application requirements. We can see that violations are mainly due to latency rather than availability, and a small number of service requests violate both, since the sum of latency and

	Uniform		**TID**	
Load	**AA**	**Baseline**	**AA**	**Baseline**
750	733.2 ± 1.9	706.9 ± 2.7	647.3 ± 2.5	638.5 ± 2.9
1750	2240.9 ± 10.7	2487.9 ± 17.7	1901.9 ± 10.8	1983 ± 14.2

Table 4: Average number of active lightpaths (i.e., IP links) in the network. Results are reported along with 95% confidence intervals.

	Uniform		**TID**	
Load	**AA**	**Baseline**	**AA**	**Baseline**
750	0.943 ± 0.001	0.955 ± 0.001	0.930 ± 0.002	0.947 ± 0.002
1750	0.969 ± 0.001	0.973 ± 0.001	0.971 ± 0.0002	0.978 ± 0.0003

Table 5: Average IP links utilization. Results are reported along with 95% confidence intervals.

availability violation probabilities is slightly higher than the overall violation probability.

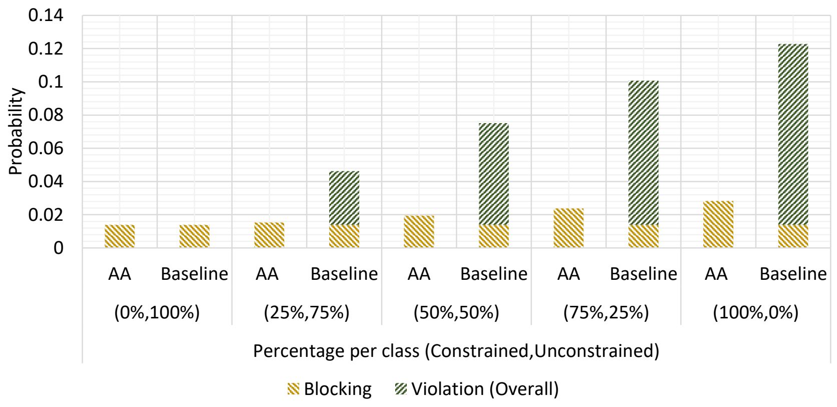
Tables 4 and 5 show the average number of active lightpaths and average IP link utilization for the two algorithms, considering 750 and 1750 Erlang network loads. The average number of active lightpaths (Tab.4) increases as the network load increases for both algorithms, i.e., as more service requests need to be provisioned. For a load of 750 Erlang, the average number of active lightpaths is slightly more for the application-aware algorithm with respect to the baseline algorithm, since more lightpaths need to be activated to meet latency and availability requirements. Conversely, for a load of 1750 Erlang, the trend is the opposite, meaning that our application-aware algorithm, at higher loads, is able to better provision service requests on already-established lightpaths. Table 5 shows instead that IP links are always almost fully utilized (more than 90% of utilization), and the average link utilization is only slightly higher for the baseline algorithm with respect to the application-aware algorithm (up to 1.7% more). This means that both algorithms are able to effectively groom traffic on existing IP links. Both Tables 4 and 5, show that our application-aware algorithm brings benefits in terms of blocking and violation probabilities at no considerable additional costs in terms of used optical and IP/MPLS resources.

Experiments shown in Figs. 5, 6a, 6b and Tables 4, 5 were repeated over a different network topology, i.e., Internet2 [50], considering uniform traffic. Very similar results were obtained, with up to 8% violation probabilities for the baseline algorithm. We do not report a detailed evaluation on the Internet2 topology for the sake of conciseness.

We focused then on a scenario where two different service request classes exist, whose application requirements are shown in the table in Fig. 7. The first class, called *constrained*, has low latency and high availability requirements, and well suits for future 5G applications. The second class, called *unconstrained*, does not consider latency and availability as relevant requirements. We set the traffic load to 1750 Erlang and we considered TID traffic distribution. Figure

Service request class	Application requirements
Constrained	$b = \{1, 10, 100\}$ Gb/s (uniform) $l = 10$ ms $a = 99.75\%$
Unconstrained	$b = \{1, 10, 100\}$ Gb/s (uniform) $l = \infty$ $a = 0\%$

(a)



(b)

Figure 7: Definition of *constrained* and *unconstrained* service request classes (a) and violation/blocking probability comparison of *application-aware* (AA) and *baseline* service provisioning algorithms when the percentage of constrained service requests increases (b). Load = 1750 Erlang, TID traffic matrix.

7b shows the blocking and violation probabilities, for both application-aware and baseline algorithms, as a function of percentage of service requests belonging to each class: (0%,100%) means that all the service requests belong to the unconstrained class, while (100%,0%) means that all the service requests are constrained. The figure shows that, as the percentage of constrained service requests increases, *(i)* blocking probability slightly increases for the application-aware algorithm, since it becomes harder to find an application-aware path for all of the constrained service requests, while *(ii)* it is constant for the baseline algorithm, since constrained and unconstrained service requests are always provisioned in the same way. However, while the violation probability is guaranteed to be zero for the application-aware algorithm, it consistently increases for the baseline algorithm, reaching 11% when all the service requests belong to the constrained class. In this specific case, our application-aware algorithm performs best, mishandling 10% service requests less than the baseline algorithm. This means that our approach will show its full potential in the future 5G era, where most of the traffic is expected to be generated by applications with very stringent requirements.

Parameter	**Value**
Link failure	
Failure generation	Poisson process ($\frac{1}{\lambda} = 400$ time units)
Failure holding time	Exponential ($\frac{1}{\mu} = MTTR_{link} = 1$ time unit)
DWDM optical and IP/MPLS layer	
$MTTR_{link}$	1 time unit
$MTTF_{link}$	$\frac{1}{\lambda} \cdot N_{links} = 22400$ time units
A_{link}	99.996%
A_{node}	100%
Service requests / application requirements	
Traffic matrix	TID
Requested bandwidth b	$\{10, 50\}$ Gb/s (uniform)
Maximum latency l	$\{10, \infty\}$ ms (uniform)
Minimum availability a	$\{99.75, 0\}$ % (uniform)
Restoration r	`fast` or `slow` (depends on simulation)

Table 6: List of parameters for service restoration simulations.

6.3. Evaluation of application-aware service restoration

In this subsection, we focus on the evaluation of the application-aware service restoration algorithm described in Section 5. We consider a scenario where bi-directional fiber links can fail. Link failures are generated according to a Poisson process, with exponential repair time (i.e., M/M/∞ queue) governed by the parameter $MTTR_{link} = 1$ time unit. We assume that the inter-arrival time between link failures is around $\frac{1}{\lambda} = 400$ time units, and the failed optical link is uniformly chosen among the whole set of links. This implies that $MTTF_{link} = \frac{1}{\lambda} \cdot N_{links} = 22400$ time units and consequently, according to Eq. 3, $A_{link} = 99.996\%$ for all the bidirectional links. We do not consider node failures in this experiment, meaning that $A_{node} = 100\%$ for all the ROADMs and IP/MPLS routers. For each service request, the b, l, a application requirements are uniformly chosen from the following sets: $b = \{10, 50\}$ Gb/s, $l = \{10, \infty\}$ ms, $a = \{99.75, 0\}$ %. We focus on the evaluation of each proposed application-aware restoration strategy (i.e., IP/MPLS restoration and multi-layer restoration) independently, meaning that we choose for all the service requests of a simulation run either $r =$ `fast` (IP/MPLS restoration) or $r =$ `slow` (multi-layer restoration), depending on the restoration strategy we want to evaluate. We always consider as traffic distribution the TID traffic matrix. The list of parameters considered in the described scenario are reported in Tab. 6. In the following experiments we compare *(i)* application-aware (*AA*) IP/MPLS and *(ii)* application-aware (*AA*) multi-layer (*ML*) restoration strategies with the three benchmark strategies reported in Tab. 1: *(iii)* baseline IP/MPLS, *(iv)* baseline multi-layer (*ML*) and *(v)* optical restoration. We focus on higher traffic loads than in the previous subsection, since for low traffic load conditions all the restoration strategies have similar performance.

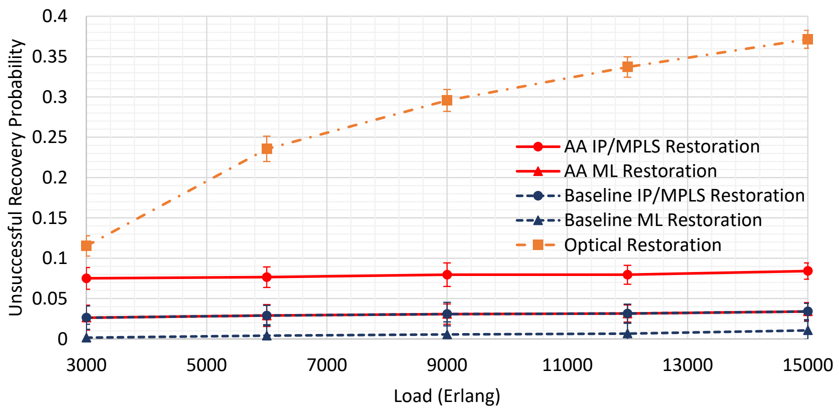
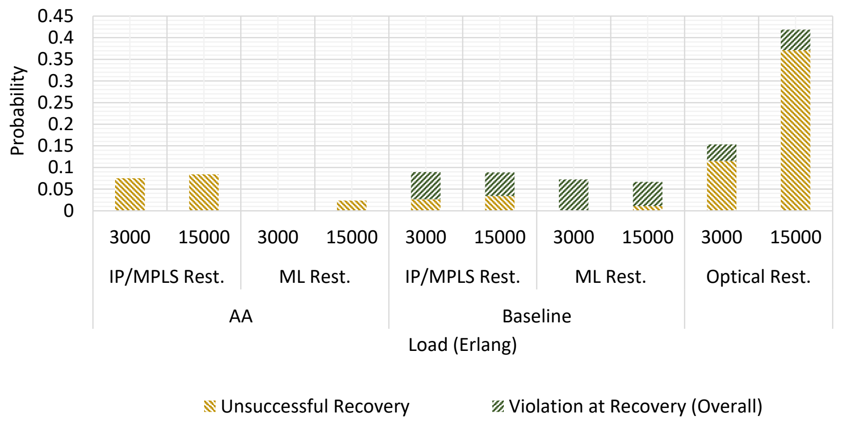


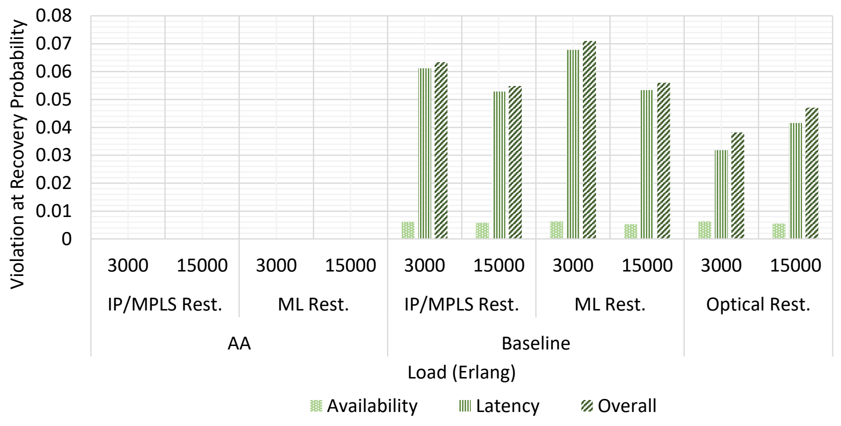
Figure 8: Unsuccessful recovery probability comparison between the different service restoration strategies: AA IP/MPLS, AA ML, baseline IP/MPLS, baseline ML and optical restoration. Results are reported along with 95% confidence interval.

Figure 8 shows the *unsuccessful recovery probability* of service requests when the restoration process is triggered, i.e., every time a failure occurs, as a function of network load. The unsuccessful recovery probability metric measures the probability that the restoration process fails for a service request. The graph shows that the fine-grained strategies (both application-aware and baseline) always outperform optical restoration. Specifically, optical restoration unsuccessful recovery probability significantly increases as the load increases (up to 40% for a load of 15000 Erlang). The reason is that with higher load it is harder to find spare optical resources to set up an alternative lightpath to circumvent a link failure. If not possible to find, the optical restoration process fails and the recovery, for all the affected service requests crossing the tiered-down lightpath, fails simultaneously. Conversely, all the fine-grained strategies can reroute each affected request individually and in a tailored way, by reusing and/or establishing IP/MPLS and optical resources more effectively. In fact, the chosen restored path for each affected service request can be different and both depends on the considered application requirements (b for baseline and b, l, a for application-aware strategies) and on the current network status. Unsuccessful recovery probability is thus kept low (always less than 10%) and is only slightly load-dependent. Moreover, the IP/MPLS strategies (both application-aware and baseline) have a higher unsuccessful recovery probability than the corresponding multi-layer ones, since they cannot establish new optical resources. Furthermore, the application-aware strategies (both IP/MPLS and multi-layer) have a higher unsuccessful recovery probability than the corresponding baseline ones, since it is harder to find a path meeting all of b, l and a application requirements (application-aware) other than only b (baseline).

Figures 9a and 9b investigate the *violation at recovery probability* metric, which evaluates the probability that a restored path for a service request violates one or more application requirements. We focus on a traffic load of 3000 and



(a)



(b)

Figure 9: Violation at recovery and unsuccessful recovery probability comparison (a) and violation at recovery probability breakdown (b) for the different service restoration strategies: AA IP/MPLS, AA ML, baseline IP/MPLS, baseline ML and optical restoration.

15000 Erlang. Both figures show that violation at recovery probability is always zero for the application-aware strategies, since they always avoid the violation of application requirements and fail the recovery if this is not possible. Conversely, it ranges from around 4% to 7% for the other strategies (baseline and optical restoration). Figure 9a shows the sum of unsuccessful recovery and violation at recovery probabilities for all the restoration strategies, thus measuring the probability of mishandling a service request in the restoration phase. Results show that such probability is always lower for the application-aware strategies than for the corresponding baseline ones. However, application-aware IP/MPLS restoration performs only slightly better than baseline IP/MPLS restoration (around 1% better). This is expected, since application-aware IP/MPLS restoration can only reuse IP/MPLS resources to meet application requirements for an affected

	ML Rest.		**Optical Rest.**
Load	**AA**	**Baseline**	
3000	60.4 ± 1.1	63.3 ± 1.4	61.7 ± 2.0
15000	88.9 ± 3.4	90.2 ± 2.3	90.3 ± 1.1

Table 7: Average number of established ligthpaths when restoration is triggered by a failure. Results are reported along with 95% confidence interval.

service request. Instead, application-aware multi-layer restoration consistently outperforms the baseline multi-layer restoration strategy (up to 7% better), meaning that application-aware multi-layer restoration is able to set up new optical resources in a more effective way to meet the requested application requirements. It is also important to note that both application-aware and baseline fine-grained strategies always significantly outperform optical restoration, especially at high traffic loads and network utilization. Figure 9b reports the breakdown of violation recovery probability into the different requirements. It shows that the violation at recovery probability for baseline and optical restoration strategies is mainly due to l violations (up to 6.5%), with only marginal a violations (around 1%), with a small fraction of service requests violating both.

Finally, Tab. 7 reports the average number of established lightpaths when restoration is triggered for network loads of 3000 and 15000 Erlang and for application-aware multi-layer restoration, baseline multi-layer restoration and optical restoration (note that IP/MPLS restoration, both application-aware and baseline, never adds lightpaths to the network). The table shows that the higher is the load, the more lightpaths need to be established in the network, when restoration is triggered, to recover the affected service requests. Moreover, application-aware multi-layer restoration requires the establishment of a slightly less number of lightpaths than the other two strategies, meaning that the benefits in terms of violation at recovery probability of our multi-layer strategy (shown in Figs. 9a and 9b) come with no additional costs in terms of established optical resources.

7. Conclusion

In this work we proposed and evaluated two dynamic algorithms for application-aware service provisioning and restoration in multi-layer transport networks. The algorithms have been designed to be part of the application-aware path computation and resource allocation modules of a multi-layer SDN orchestrator, and have been integrated in an orchestrator implementation [10].

We first formalized the problems of application-aware service provisioning and restoration, according to which a service request must be provisioned or restored in such a way that all the application-specific requirements are met. We identified an auxiliary-graph-based approach as the most suitable strategy to solve such problems, and introduced the auxiliary graph model. We then described our application-aware service provisioning and restoration algorithms, which properly create and manipulate auxiliary graphs. The service provisioning algorithm provides, to each service request, an application-aware path meeting all of bandwidth, latency and availability requirements; however, since it is designed in a modular way, it could be extended to consider additional ones. Furthermore, the service restoration algorithm allows to individually restore each failure-affected service while meeting bandwidth, latency, availability and service disruption tolerance.

We evaluated our algorithms by means of simulations. We showed that they outperform some benchmark and state-of-the-art service provisioning and restoration strategies, since they always prevent application requirement violations with only a marginal increase in blocking/unsuccessful recovery probabilities, especially when network load is unbalanced. Moreover, this comes at no considerable additional costs in terms of used optical and IP/MPLS resources. We also showed that our algorithms perform best when most of the applications have strict latency and availability requirements, as it is foreseen to happen in the future 5G era.

As future work, we plan to improve our algorithms with respect to the auxiliary graph construction phase (e.g. by identifying and evaluating new potential lightpath edge addition policies) and the path selection phase (e.g. by considering more complex weights for the auxiliary graph edges, not only related to the physical length of lightpaths but also to their residual bandwidth) to achieve further traffic engineering objectives other than, as done in this paper, maximizing the usage of IP/MPLS network resources.

Acknowledgment

The research leading to these results has received funding from the European Commission within the H2020 Research and Innovation programme, ACINO project, Grant Number 645127, www.acino.eu.

The authors would like to thank Jose Luis Izquierdo Zaragoza (from Universitat Politecnica de Catalunya) for the help in early implementation stages with Net2Plan, and Victor Lopez (from Telefonica) for providing network data.

References

[1] R. Vannithamby, S. Talwar, Towards 5G: Applications, Requirements and Candidate Technologies, John Wiley & Sons, 2017.

[2] G. P. Fettweis, The tactile internet: Applications and challenges, IEEE Vehicular Technology Magazine, vol. 9, no. 1, pp. 64–70, 2014.

[3] M. Maurer, J. C. Gerdes, B. Lenz, H. Winner, Autonomous driving: technical, legal and social aspects, Springer, 2016.

[4] R. Munoz, A. Mayoral, R. Vilalta, R. Casellas, R. Martinez, V. Lopez, The need for a transport api in 5g networks: The control orchestration protocol,

in: 2016 Optical Fiber Communications Conference and Exhibition (OFC), 2016.

[5] P. Skoldstrom, S. Junique, Application-centric networks and the future 5G transport, in: International Conference on Transparent Optical Networks (ICTON), 2015.

[6] V. Lopez, D. Konidis, D. Siracusa, C. Rozic, I. Tomkos, J. P. Fernandez-Palacios, On the benefits of multilayer optimization and application awareness, IEEE/OSA Journal of Lightwave Technology, vol. 35, no. 6, pp. 1274-1279, 2017.

[7] M. Pham, D. B. Hoang, SDN applications - the intent-based northbound interface realisation for extended applications, in: 2016 IEEE NetSoft Conference and Workshops (NetSoft), 2016.

[8] O. Gerstel, V. Lopez, D. Siracusa, Multi-layer orchestration for application-centric networking, in: International Conference on Photonics in Switching (PS), 2015.

[9] F. Pederzolli, D. Siracusa, P. Skoldstrom, S. Junique, C. Rozic, D. Klonidis, T. Szyrkowiec, M. Chamania, V. Uceda, V. Lopez, Y. Shikhmanter, O. Gerstel, SDN application-centric orchestration for multi-layer transport networks, in: International Conference on Transparent Optical Networks (ICTON), 2016.

[10] Final implementation of the ACINO orchestrator, in: ACINO Deliverable D4.3, 2017.

[11] I. Chlamtac, A. Farago, T. Zhang, Lightpath (wavelength) routing in large WDM networks, IEEE Journal on Selected Areas in Communications, vol. 14, no. 1, pp. 909–913, 1996.

[12] J. Zhang, Y. Ji, M. Song, H. Li, R. Gu, Y. Zhao, J. Zhang, Dynamic virtual network embedding over multilayer optical networks, OSA Journal of Optical Communication and Networking, vol. 7, no. 9, pp. 918–927, 2015.

[13] M. Gjoka, V. Ram, X. Yang, Evaluation of IP fast reroute proposals, in: International Conference on Communication Systems Software and Middleware, 2007.

[14] R. Doverspike, J. Yates, Challenges for MPLS in optical network restoration, IEEE Communications Magazine, vol. 39, no. 2, pp. 89–96, 2001.

[15] S. Das, Y. Yiakoumis, G. Parulkar, N. McKeown, P. Singh, D. Getachew, P. D. Desai, Application-aware aggregation and traffic engineering in a converged packet-circuit network, in: OSA Optical Fiber Communication Conference (OFC), 2011.

[16] J. S. Dantas, D. Careglio, R. M. Silveira, W. V. Ruggiero, J. Sol-Pareta, PCE algorithm for traffic grooming and QoS in multi-layer/multi-domain IP over WDM networks, in: International Conference on Transparent Optical Networks (ICTON), 2011.

[17] J. Liu, L. Zhu, W. Sun, W. Hu, Scalable application-aware resource management in software defined networking, in: International Conference on Transparent Optical Networks (ICTON), 2015.

[18] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, G. Noubir, Application-awareness in SDN, in: ACM SIGCOMM Conference, 2013.

[19] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, P. Tran-Gia, SDN-based application-aware networking on the example of YouTube video streaming, in: 2013 European Workshop on Software Defined Networks (EWSDN), 2013.

[20] A. Khalil, A. Hadjiantonis, M. Ali, N. Abdellatif, J. Moghaddasi, Dynamic routing of sub-wavelength connections in IP/MPLS over WDM networks, in: Optical Communications Systems and Networks Conference (OCSN), 2004.

[21] A. Bukva, R. Casellas, R. Martinez, R. Munoz, A dynamic path-computation algorithm for a GMPLS-enabled multi-layer network, IEEE/OSA Journal of Optical Communications and Networking, vol. 4, no. 6, pp. 436–448, 2012.

[22] M. Alhowaidi, P. Yi, B. Ramamurthy, Dynamic provisioning in virtualized cloud infrastructure in IP/MPLS-over-WDM networks, in: International Conference on Computing, Networking and Communications (ICNC), 2017.

[23] H. T. Nguyen, A. V. Vu, D. L. Nguyen, V. H. Nguyen, M. N. Tran, Q. T. Ngo, T. H. Truong, T. H. Nguyen, T. Magedanz, A generalized resource allocation framework in support of multi-layer virtual network embedding based on SDN, Elsevier Computer Networks, vol. 92, part 2, pp. 251–269, 2015.

[24] H. Alshaer, J. M. H. Elmirghani, Multilayer dynamic traffic grooming with constrained differentiated resilience in IP/MPLS-over-WDM networks, IEEE Transactions on Network and Service Management, vol. 9, no. 1, pp. 60–72, 2012.

[25] S. C. Lin, I. F. Akyildiz, P. Wang, M. Luo, QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach, in: IEEE International Conference on Services Computing (SCC), 2016.

[26] F. S. Tegueu, S. Abdellatif, T. Villemur, P. Berthou, T. Plesse, Towards application driven networking, in: 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2016.

[27] G. Shen, H. Guo, S. K. Bose, Survivable elastic optical networks: survey and perspective (invited), Springer Photonic Network Communications, vol. 31, no. 1, pp. 71–87, 2016.

[28] J. Foo, A survey of service restoration techniques in MPLS networks, in: Australian Telecommunications, Networks and Applications Conference, 2003.

[29] B. J. Ambika, N. Naga Maruthi Kumari, M. K. Banga, A Survey on the path restoration in MPLS Networks, 2016 International Conference on Cognition and Recognition (ICCR), Springer Singapore, 2016.

[30] O. Gerstel, C. Filsfils, T. Telkamp, M. Gunkel, M. Horneffer, V. Lopez, A. Mayoral, Multi-layer capacity planning for IP-optical networks, IEEE Communications Magazine, vol. 52, no. 1, pp. 44–51, 2014.

[31] O. Gerstel, C. Filsfils, W. Wakim, IP-optical interaction during traffic restoration, in: OSA Optical Fiber Communication Conference (OFC), 2013.

[32] A. Castro, L. Velasco, J. Comellas, G. Junyent, Dynamic restoration in multi-layer IP/MPLS-over-flexgrid networks, in: International Conference on the Design of Reliable Communication Networks (DRCN), 2013.

[33] A. L. Chiu, G. Choudhury, M. D. Feuer, J. L. Strand, S. L. Woodward, Integrated restoration for next-generation IP-over-optical networks, IEEE/OSA Journal of Lightwave Technology vol. 29, no. 6, pp. 916–924, 2011.

[34] M. Gunkel, Multi-layer restoration - the impact on the optical layer, in: OSA Optical Fiber Communication Conference (OFC), 2014.

[35] L. Liu, D. Zhang, T. Tsuritani, R. Vilalta, R. Casellas, L. Hong, I. Morita, H. Guo, J. Wu, R. Martinez, R. Munoz, First field trial of an openflow-based unified control plane for multi-layer multi-granularity optical networks, in: OSA Optical Fiber Communication Conference (OFC), 2012.

[36] Z. Lu, Y. Jayabal, M. Razo, M. Tacca, A. Fumagalli, G. M. Galimberti, G. Martinelli, G. Swallow, Optical layer-driven network restoration and redesign for improved fast reroute reliability, in: International Conference on Transparent Optical Networks (ICTON), 2016.

[37] Y. Liu, H. Li, C. Lyu, Z. Duan, Y. Ji, Experiment demonstration of multi-layer restoration in service-oriented software defined optical network, Electronics Letters vol. 53, no. 14, pp. 935–937, 2017.

[38] I. S. Hwang, Z. D. Shyu, C. Z. Yang, K. P. Chen, QoS-constraint multicast restoration with 2-tuple domination core nodes selection in DWDM mesh networks, Telecommunication Systems, vol. 49, no. 3, pp. 287–298, 2012.

[39] A. M. Ghaleb, T. Khalifa, K. B. Shaban, Enhancing the performance of post-failure restoration schemes in multi-tenant networks, in: International Conference on Network and Service Management (CNSM), 2016.

[40] D. Amar, E. L. Rouzic, N. Brochier, C. Lepers, Class-of-service-based multilayer architecture for traffic restoration in elastic optical networks, OSA Journal of Optical Communications and Networking, vol. 8, no. 7, pp. A34–A44, 2016.

[41] M. Santuari, T. Szyrkowiec, M. Chamania, R. Doriguzzi-Corin, V. Lopez, D. Siracusa, Policy-based restoration in IP/optical transport networks, in: IEEE NetSoft Conference and Workshops (NetSoft), 2016.

[42] M. Savi, F. Pederzolli, D. Siracusa, An application-aware multi-layer service provisioning algorithm based on auxiliary graphs, in: Optical Fiber Communications Conference (OFC), 2017.

[43] J. Zhang, K. Zhu, H. Zang, B. Mukherjee, A new provisioning framework to provide availability-guaranteed service in WDM mesh networks, in: IEEE International Conference on Communications (ICC), 2003.

[44] S. Verbrugge, D. Colle, P. Demeester, R. Huelsermann, M. Jaeger, General availability model for multilayer transport networks, in: International Workshop on Design of Reliable Communication Networks (DRCN), 2005.

[45] A. W. Brander, M. C. Sinclair, A Comparative Study of k-Shortest Path Algorithms, Springer London, London, 1996, pp. 370–379.

[46] C. Rozic, M. Savi, C. Matrakidis, D. Klonidis, D. Siracusa, I. Tomkos, Application-centric dynamic multi-layer resource allocation in availability-aware SDN-orchestrated networks, in: 43rd European Conference on Optical Communication (ECOC), 2017.

[47] P. Pavon-Marino, J. L. Izquierdo-Zaragoza, Net2plan: an open source network planning tool for bridging the gap between academia and industry, IEEE Network, vol. 29, no. 5, pp. 90–96, 2015.

[48] F. Rambach, B. Konrad, L. Dembeck, U. Gebhard, M. Gunkel, M. Quagliotti, L. Serra, V. Lopez, A multilayer cost model for metro/core networks, OSA Journal of Optical Communications and Networking, vol. 5, no. 3, pp. 210–225, 2013.

[49] J. J. Pedreno-Manresa, J. L. Izquierdo-Zaragoza, P. Pavon-Marino, Guaranteeing traffic survivability and latency awareness in multilayer network design, Journal of Optical Communications and Networking, vol. 9, no. 3, pp. B53-B63, Mar. 2017.

[50] Internet2 network infrastructure topology (Oct. 2014).
URL `https://www.internet2.edu/media/medialibrary/2017/09/25/I2-Network-Infrastructure-Topology-Alllogos-201705_hr8gwSg.pdf`