Orchestrating Virtual Slices in Data Centre Infrastructures with Optical DCN

Albert Pagès[1], Fernando Agraz[1], Rafael Montero[1], Giada Landi[2], Marco Capitani[2], Domenico Gallico[3], Matteo Biancani[3], Reza Nejabati[4], Dimitra Simeonidou[4] and Salvatore Spadaro[1]
1: Universitat Politècnica de Catalunya (UPC), Spain
2: Nextworks S.r.l, Italy
3: Interoute S.P.A, Italy
4: University of Bristol, UK
albertpages@tsc.upc.edu

Abstract—The emergence of new paradigms and services is pushing the limits of nowadays cloud infrastructures. It is a fact that current solutions lack in the flexibility and configurability to adapt to heterogeneous requirements coming from the applications/services to be supported over them. This results most of the time in severe underutilization of the underlying physical substrate. In light of this, newer approaches on resource provisioning and infrastructure management are needed. The Infrastructure as a Service (IaaS) paradigm is proposed as a solution to overcome these limitations. Thanks to IaaS, the physical infrastructure is partitioned onto virtual slices, encompassing heterogeneous resources (e.g. network, computing). Such a concept is expected to be harnessed by future data center (DC) infrastructures in order to cope efficiently with multi-tenancy as well as heterogeneous application requirements. However, current DC networks (DCNs) impose sever limitations onto traffic handling to fully exploit this vision. In light of this, optical technologies are seen as prime candidates for realizing the high performance network fabrics that future DC architectures will need. Under such an umbrella, it becomes primordial to develop specific provisioning solutions that account for the particularities of the optical medium, while providing the means to efficiently slice the DC infrastructure. With all of these in mind, in this article we present a solution for orchestrating and controlling virtual slices in a DC scenario with optical intra-DCN, with the scope of optimizing the underlying physical infrastructure utilization. The benefits of the presented solution are demonstrated against legacy architectures through exhaustive experiments and simulations.

Keywords—Data centers, optical networks, slicing, Software Defined Networking, resource orchestration, virtualization.

I. INTRODUCTION

Future cloud and telecom infrastructures are expected to meet very heterogeneous requirements in regards not only of resources to be provisioned, but also on the life cycle and dynamics of the applications and services that will have to be supported over them. Indeed, as we move towards future 5G mobile infrastructures there is an increasing interest on deeply integrating resource provisioning solutions with upper applications needs to deliver optimized services towards end users [1]. Such an approach raised the concept of vertical business, in which actors of the economic activity are identified (e.g. automotive industry). Then, such actors request to cloud and telecom infrastructure owners and service providers

for the necessary support to develop their own activities in the form of infrastructure resources, communication services and software functions. This substrate then becomes the foundation for service delivery towards the verticals' final users.

In this scenario, data center (DC) infrastructures are an essential part in the whole service delivery chain. Thanks to the collaborative efforts of the thousands of servers hosted inside their premises, rich applications can be realized (e.g. cloud storage, search engines). Additionally, novel paradigms like Network Function Virtualization (NFV) [2] require the deployment of complex software functions, usually in the form of Virtual Machines (VMs) or computing resources, for which DC infrastructures provide the necessary support to materialize them. However, current DC architectures are far from prepared to accommodate the plethora of customers that will need to be supported, having all of them heterogeneous requirements in terms of resources, Quality of Service (QoS), degree of control over the employed resources, and so on. Indeed, multi-tenancy and infrastructure customization are limited due to current service provisioning structures, with infrastructure owners focusing on the services offered on top of their infrastructures. To overcome such limitations, infrastructure slicing is seen as the next big revolutionizer [3] towards efficient infrastructure management. By means of virtualization techniques, the physical substrate is partitioned in logical instances, which are then delivered as isolated virtual infrastructures to meet the necessities of the tenant.

On another hand, DCs are also faced with increasing limitations on the traffic handling aspect, jeopardizing the performance of intra-DC networks (DCNs) for server-to-server communications and, as a consequence, the whole service delivery process. Indeed, global IP traffic handled by DCs has increased at a compound annual growth rate (CAGR) of around 25% during the last years and is expected to reach about 20.6 ZB of traffic handled per year by 2021, for which around 71.5%, 13.6% and 14.9% of the traffic relates to the intra-DC, inter-DC and DC-to-user share, respectively [4]. Such traffic growths are pushing the limits of current electronic-based DCN solutions, which cannot withstand the expected bandwidth, latency and scalability requirements of future DCN architectures, thus making it necessary to adopt other solutions. Optical network technologies are envisioned to overcome these limitations due to their superior scalability, bandwidth, latency and power consumption [5]. However, optical technologies require the presence of specialized control and management solutions to fully harness their capacities.

Nevertheless, aside from the challenges posed, the use of optical networking technologies in conjunction with infrastructure slicing is envisioned to enable flexible, cost-effective and tailored service provisioning in future DC infrastructures. The enabling technology for slicing is virtualization of the physical hardware. Through virtualization techniques, logical abstractions of the underlying physical hardware are obtained. Then, such abstractions are combined onto logical infrastructures, i.e. the slices, which will be exploited by external entities. Indeed, the concept of slicing leverages onto the Infrastructure as a Service (IaaS) paradigm, in which infrastructure owners offer part of their physical infrastructure for leasing by external entities as support to develop their own business model. However, the concept of slicing is far more ambitious. The idea of slice provisioning envisioned in future 5G cloud and telecom infrastructures does not only contemplate that physical resources in several segments (mobile, DC, core) are offered as elements to compose

the slice instances, but also the possibility to offer functions and applications that then will be integrated onto the virtual infrastructure, enriching the capacities of the slice. For example, the NFV paradigm is seen as an enabler of such concept. Moreover, slices are expected to have a dynamic life cycle, with the capacity to scale-up and down in both required resources and functions to meet the evolving needs of the applications and services that will run on top. Lastly, but not least, slices are expected to have their own control and management solutions, completely isolated from neighboring slices. In such a way, it would be possible to operate the slice as if it was owned by the external renting entity, opening up the possibility of dynamically on-boarding custom control and management solutions tailored to their specific needs.

Generally speaking, slices are composed of multiple types of resources, encompassing both computing and networking capacities. In the context of the present work, and without loss of generality, it is assumed that slices request for a virtual infrastructure composed of virtual nodes with computational capacities expressed in the form of VMs. Then, these VMs are connected through a virtual network graph composed of virtual links stating the desired bandwidth. This virtual graph must then be allocated over the physical infrastructure, guaranteeing the isolation across slice instances. Such allocation (or mapping) requires namely to compute the embedding of two types of resources: nodes and links. The node mapping involves finding the most suitable servers to deploy the VMs while the link mapping involves the route calculation and allocation of network resources interconnecting the servers in which VMs are deployed. Such problem is known as the Virtual Infrastructure Embedding (VIE) problem and should be tailored to the characteristics of the underlying physical substrate. For instance, in the presence of optical network technologies, the link mapping involves a Route and Wavelength Assignment (RWA) problem which then must be solved [6]. Figure 1 depicts an example of the assumed slice provisioning and mapping onto an optical DC infrastructure, exemplifying how multiple slices can coexists thanks to the isolation provided by the virtualization of the physical substrate, thus truly achieving multi-tenancy in a cloud environment.

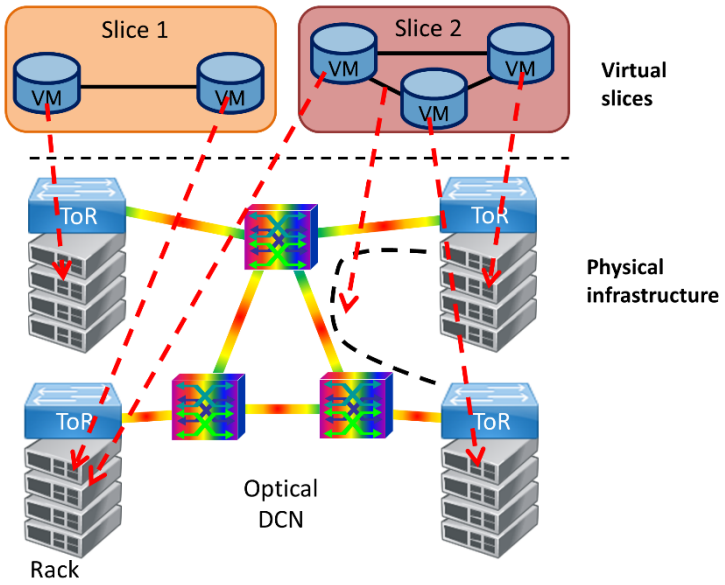


Fig. 1. Example of slicing over a DC infrastructure.

Traditional resource provisioning mechanisms in DCs are usually structured in two phases: a first phase where the decisions on which servers to deploy the VMs are taken and a second phase to decide the physical network resources to establish the connectivity between them. In this regard, resource provisioning is usually taken considering only the computing resources workloads (e.g. [7]-[9]), optimizing their load distribution [10]. However, such approach usually pays little to none regard to the status of the network resources interconnecting the servers, thus potentially leading to poor performance due to network congestion of the paths interconnecting them. In light of this, joint resource provisioning mechanisms, in which the server selection for the VMs also considers the network connectivity between servers, are a must in modern DC architectures. By means of a joint resource provisioning, it is possible for an overall optimization of the DC infrastructure, increasing the resource utilization and enhancing the service provisioning [11], [12]. Thus, resource orchestration becomes primordial in the context of virtual slice provisioning, in order to find the most optimized layout for the resources of the slice requests. Moreover, automatized resource orchestration can significantly improve the deployment time of slices, thus shortening the service time to market.

All of these challenges, coupled with more dynamic traffic patterns as well as a paradigm shift towards deep integration of network resource onto the cloud [13], require that new DC management solutions must seek for a joint orchestration of both network and computing resources to optimize the physical resource utilization. Moreover, special focus on solutions tailored to deliver the necessary means to slice DC resources is needed. With these requirements in mind, we present a solution which encompasses state-of-the-art optical technologies for the intra-DCN network fabric as well as novel orchestration and control software solutions that enable the dynamic, on-demand, automatic and joint provisioning of both networking and computing resources over the physical DC infrastructure to fulfil the requirements of complex cloud services. The rest of the paper is structured as follows: Section II further reviews work found in the literature in regards of the VIE problem and control and orchestration architectures for DC environments. Next, Section III presents the proposed DC architecture, detailing the solutions adopted for dynamic management and control in the overall architecture as well as the developed algorithm for computing the joint resource allocation of virtual slice requests. Section IV thoroughly evaluates the presented solution from different perspectives, highlighting the benefits to slice allocation against legacy solutions. Finally, Section V draws the conclusions of the work.

II. RELATED WORK

A. **Virtual Infrastructure Embedding (VIE) Problem**

Slice mapping, or the generic VIE problem, has been quite studied in the literature [14], [15] and several solutions have been proposed to provide a joint node and link mapping of slices, accounting for the requirements of the slice demands and the available resources at the network substrate. For instance, authors in [16] propose several exact and heuristic mechanisms for embedding virtual infrastructures in a generic electronic-based substrate

network, making use of graph extension techniques to decide the joint mapping of nodes and links. The scope of the optimization problem is to embed any incoming slice request with the minimal cost while balancing the load of the physical infrastructure. Other authors in [17] also propose several heuristic algorithms based on graph transformations to tackle the embedding of slices in generic electronic-based networks, with the aim of maximize the number of accepted slices and the operator's revenues over the time. However, in both works, the only resource that is accounted for both physical and virtual nodes is CPU, not contemplating memory and storage resources, which would be the case in a more realistic scenario and would require different approach to account for the multi-dimensionality of the node embedding phase.

With the rise of optical technologies at the DCN, new mapping mechanisms to fully harness their capabilities are a required. Additionally, when providing connectivity along an optical-based DCN, bandwidth allocation can greatly differ among different optical technologies (e.g. Wavelength Division Multiplexing (WDM), Elastic Optical Networks (EONs)). Thus, slice provisioning solutions must also account for the particularities of the underlying optical technology of the DCN. In this regard, several works have studied the VIE problem in optical networks and proposed slice embedding mechanisms accounting for different aspects related to optical networks in DC infrastructures. For instance, authors in [18] focus on the case of slice embedding in EONs, distinguishing the scenarios in which transparent or opaque connectivity services are required. A collection of exact and heuristic mechanisms is proposed, with the aim of minimizing the total number of frequency slots occupied on the network as well as the maximum number of frequency slots needed per link to satisfy the embedding of the slice requests.

Focusing in the same scenario of EONs, authors in [19] propose both exact and heuristic mechanisms for online and offline embedding scenarios, with the aim of minimizing the cost of the embedding solution. In this case, the authors account for CPU, storage and memory resources for the node embedding phase. A similar modelling of the slices demands is following in [20], where authors focus more in the planning stage of an inter-DC scenario, in which both the necessary IT resources (CPU cores, storage and memory) of DC sites and the wavelengths at the WDM-based inter-DC network need to be determined to accommodate an already known slice demand set.

Other aspects of the VIE problem not focusing purely on the resource utilization or efficiency of the embedding have been also studied, such as authors in [21], which tackled the energy consumption aspect when provisioning virtual networks in optical DCs. Energy consumption is becoming a very serious challenge in nowadays' DCs and many research efforts are devoted to provide solutions to reduce it. To this goal, the authors proposed an embedding mechanism that minimizes the number of electronic ports used when deploying a virtual network in a hybrid electrical/optical DC infrastructure. Having reviewed these several works surrounding the VIE topic, we provide a table summarizing their main traits, highlighting the aspects that have been studied, such as the network scenario and the scope of the optimization goal.

TABLE I. SUMMARY OF VIE-RELATED WORK

Work	Mechanism	Joint embedding	Network technology	Node resources	Scenario	Optimization goal	Multi-technology embedding depending on demand requirements
[16]	Exact; heuristic	Yes	Generic electronic-based	Only CPU	Online	Minimal cost; load balancing	No
[17]	Heuristic	Yes	Generic electronic-based	Only CPU	Online	Maximize demand acceptance; maximize revenues	No
[18]	Exact; heuristic	Yes	EONs	Only CPU	Online	Minimum use of frequency slots	No
[19]	Exact; heuristic	Yes	EONs	CPU; storage; memory	Online; offline	Minimum embedding cost	No
[20]	Exact; heuristic	Yes	WDM-based optical networks	CPU; storage; memory	Offline	Minimum resources to be equipped at substrate	No
[21]	Heuristic	No; only virtual link mapping is considered	Hybrid: WDM-based optical networks; Electronic packet switching	Not considered	Online	Minimum resource consumption	Yes

One important aspect that can be noted is that the majority of work related to VIE simplifies the aspect of the node embedding problem, that is, the multi-dimensional notion of resource requirements (CPU, storage, memory) is not properly tackled. Another important aspect is the selection of the resources in which slices are embedded in awareness of their needs. That is, in the presence of a data plane composed of hybrid networking technologies, it is of the most relevance to select the most suitable technology to satisfy the requirements of the slice, considering how the selection contributes to the optimization goal at hand. In this regard, a substantial number of studies found in the literature focus in scenarios in which a single network technology is present, thus, the embedding algorithm does not distinguish between networking resources based on their properties and the requirements of virtual links. With all of these in mind, in the present work, for the slice embedding calculation we propose an embedding algorithm that accounts for the multi-dimensionality of resource requirements at virtual nodes, tying the selection of the physical node according to the balanced usage of CPU, storage and memory resources inside servers. Moreover, for the virtual link mapping, our proposed heuristic allocates the most suitable network path according to their bandwidth requirements. In this regard, pure optical switching paths for the inter-rack route are selected for high bandwidth virtual links while paths with intermediate nodes in which opto-electronic

conversion capabilities are present are relegated to less bandwidth hungry virtual links, all in all, making the best use of the proposed DCN fabric. The details of the said fabric and of the embedding algorithm will be explained in latter sections.

B. **Orchestration and Control Architectures**

From an architectural perspective, there are several works that have proposed solutions to control and manage resources in optical networks. For instance, authors in [22] experimentally demonstrated a control architecture based in Software Defined Networking (SDN) to provide slices across EONs with the scope of implementing survivable services. In this regard, the SDN-based control layer is able to build on-demand QoS-aware restoration paths for the provisioned virtual slices. Similarly, and focusing also in generic optical networks, authors in [23] propose an SDN control schema to provide connectivity to virtual tenants across a shared EON with the goal of high availability in mind. Note that these works focus only on the control aspect of the network, hence, no orchestration solution is explored. More in the line of resource orchestration, authors in [24], presented an architecture composed of both an orchestration layer and an SDN control layer. The main goal of the presented architecture is to enable the provisioning and reconfiguration of virtual network slices across multi-technology transport networks. Thus, resource orchestration is essential to tailor the slice embedding to the particularities of each technological domain.

As for solutions providing control and/or orchestration architectures in optically interconnected DCs, both intra- and inter-DC, there exists several works that provided architectural designs and experimental validations. For instance, authors in [25] proposed a novel SDN-based Optical Packet Switching (OPS) DCN architecture that allowed for the automatic provisioning of connections aiming to improve the QoS of the overall system. Authors in [26] leverage on this work and opened the connectivity service delivery as an external service to be exploited by tenant applications, enabling the possibility to provide DC slices. More focused in the inter-DC scenarios, authors in [27] experimentally evaluated a control architecture to provide virtual slices across DCs connected through an inter-DCN based on EONs. To enable a more efficient utilization of the multi-layered IP/optical data plane, the authors integrate a cross-stratum embedding algorithm in the control schema to account for the particularities of the IP layer and the optical substrate. On the lines of this work, authors in [28] provide an architecture capable to orchestrate slices for the interconnection of DC spread over several network operators. To this end, the authors provide both control and orchestration solutions to efficiently deal with the multi-segment/operator scenario. Like in the previous sub-section, we provide a table summarizing the reviewed work, highlighting their main contributions in regards of control and orchestration solutions.

TABLE II. SUMMARY OF CONTROL AND ORCHESTRATION ARCHITECTURES-RELATED WORK

Work	Control layer	Orchestration layer	Network technology	Single-/Multi-segment	Node resources	Cross-stratum and cross-technology orchestration

[22]	Yes	No	Elastic optical networks	Single	Not present	No
[23]	Yes	No	Elastic optical networks	Single	Not present	No
[24]	Yes	Yes	Hybrid: EONs; WDM-based optical networks; OPS	Multi	Not present	Yes; cross-technology
[25]	Yes	No	Hybrid: WDM-based optical networks; OPS	Single	Not present	No
[26]	Yes	No	Hybrid: WDM-based optical networks; OPS	Single	Not present	No
[27]	Yes	Limited	Hybrid: EONs; IP networks	Multi	Present; not specified	Yes; cros-stratum
[28]	Yes	Yes	Hybrid: EONs; IP networks	Multi	Present; not specified	Yes; cross-segment

Although there are several solutions that tackle the control and orchestration of network and IT resources in optical-based intra-DCN scenarios, there are still several gaps that need to be covered. For instance, in the presence of both IT and network resources, it is important to provide solutions that not only orchestrate jointly the deployment of the whole virtual slice, but also automatize the workflow at all levels. Is not enough to configure and orchestrate the optical path that will support the physical connectivity of the slices, but also provide the means to automatically and transparently configure the electrical part of the network (the IP network), keeping track of the employed IP resources (e.g. network and sub-network addressing spaces and ports). Additionally, in the presence of hybrid data planes, the orchestration and control layers must be able to select and configure the most appropriate resources to be accommodated to virtual slice according to their needs, be them bandwidth or latency, to name some. Having all of these in mind, we provide a solution for slice deployment in intra-DCN scenarios, combining the capacities of specialized software layers to account for the characteristics of the underlying physical infrastructure with an optical DCN fabric for low latency, high throughput and energy efficient communication delivery across the whole DC. The proposed solution is intended to aid on the creation of slice instances in support of 5G services, for which DC integration is essential in the holistic vision of network and IT infrastructures convergence to deliver advanced and highly customizable services [29]. Indeed, as exemplified previously, advanced paradigms expected to be harnessed in future 5G infrastructures, such as NFV, require the support of DCs, as network processing and functions move from network nodes to general processing hardware (i.e. servers) and virtualization becomes more and more prominent. Thus, high performance DC slices and architectures for they delivery are a must. With all this in mind, the next section introduces the solution adopted, along with the rest of its architecture.

III. ARCHITECTURE FOR SLICE PROVISIONING

Slice provisioning is a daunting operation, which requires specialized solutions to fully exploit the capabilities of the physical infrastructure. This, in conjunction with the rising trend to bring optical network technologies inside DCs, pledges for new architecture designs able to provide virtual slices combining both computing and network resources in an automated and efficient way. For this, we introduce our proposed architecture for future DCs and the adopted approaches to realize a joint orchestration of DC resources when deploying cloud services (e.g. slice instances). Note, however, that the current work focuses on the control and orchestration layers, which are the main novelty along with the developed algorithm for joint slice mapping. Thus, subsequent sections will emphasize the functionalities of the developed software layers and how they collaborate to deliver virtual slice provisioning in optical DCs. Nevertheless, a brief description of the data plane along with performance evaluations including all layers are also supplied to provide an overall picture of the developed architecture. With this, Figure 2 depicts a schematic of the architecture, summarizing the main layers: data (bottom), control (middle) and orchestration (top).

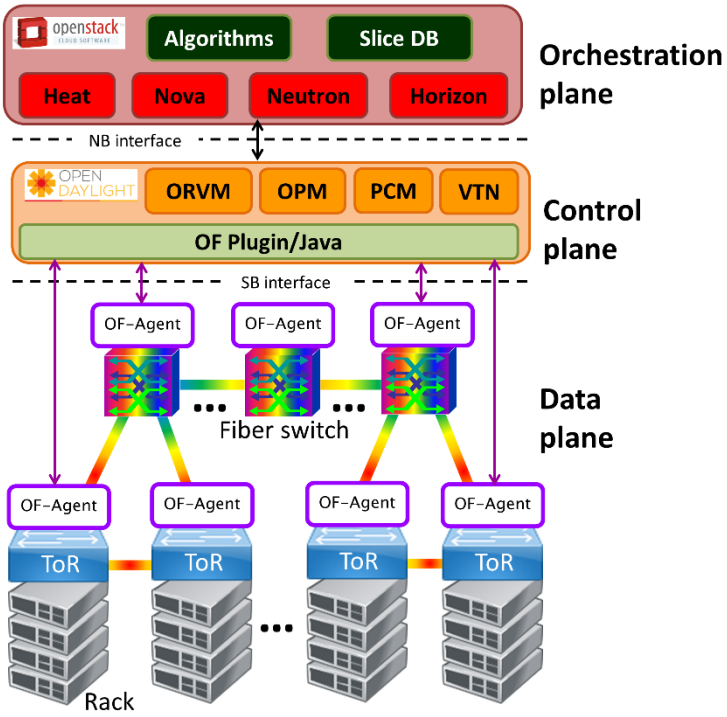


Fig. 2. Proposed DC architecture.

Starting from the bottom, a novel state-of-the-art optical data plane to realize the intra-DCN is employed. In order to cope with future DC traffic requirements in regards of improved bandwidth, scalability and latency, such DCN encompasses high-radix Top of the Rack (ToR) switches to enable the intra-rack communication [30] as well as large port count optical fiber switches for the inter-rack communication [31]. In summary, servers hosted in the racks are connected by means of Ethernet cables to a single ToR node, one per rack. Such node then processes the packets coming from the servers and aggregates them onto a single optical circuit connection to be transferred between a source and a destination pair of racks. To enable such connectivity, ToRs are connected by fiber links to neighboring ToR nodes

inside the cluster and to an optical fiber switch node, which enables bypassing unnecessary ToR switches whenever high throughput and low latency communications are needed. Then, fiber switches are connected in an arbitrary flat fiber network (only one level of switches is contemplated) that spans the whole DC, thus enabling the inter-rack/cluster communication between servers. Overall, such data plane provides a high performance flat optical network architecture, which overcomes the main challenges of nowadays electronic-based DCN solutions.

Second, to provide a dynamic configuration of the optical data plane, an SDN control layer is provided, which allows for the on-demand provisioning of end-to-end connectivity across the DCN in support of virtual slice creation. In this regard, all configuration decisions and operations are performed in a central controller while the communication with the data plane devices is achieved through means of the OpenFlow (OF) protocol. Hence, all control logic is removed from the data plane, which becomes a simple data forwarding layer, and condensed at the SDN controller. Several platforms offer the core functionalities for SDN-based control layer development and implementation (e.g. ONOS, FloodLight, Ryu, etc.). Among them, in our solution, the OpenDaylight (ODL) SDN platform is employed, based on its modularity and ease of extension [32]. In this regard, some core functionalities of ODL have been extended to cope with the presence of optical equipment while new ones have been developed. In essence, the extended modules are: the ODL DLUX web interface to cope with the optical DCN representation; the Topology Manager (TM) to maintain the topology of the physical DCN; the Virtual Tenant Network (VTN), responsible for the configuration of the Open Virtual Switch (OVS) instances at the physical servers, which act as a virtual bridge to interconnect the VMs to the physical DCN; and the OF plugin and protocol, along with OF-capable agents for the optical nodes, which augment the OF version 1.0 specification [33] using the optical extensions document [34].

As for the newly introduced modules, they are: the Optical Resource Virtualization Manager (ORVM), whose mission is to support the virtualization of the optical nodes and creating the optical segment of the virtual network slice requests; the Optical Provisioning Manager (OPM), which is responsible for the creation and destruction of optical connections at the data plane. In this regard, such functionalities are invoked by the ORVM according to the desired characteristics of the connections to be configured. Then, The OPM will send the appropriate OF rules to the physical devices in accordance with the characteristics of the connection; and the Path Computation Manager (PCM), which is responsible to determine the route to be followed by optical connections being configured through the OPM. To do so, it consumes the topological information managed by the TM and applies simple routing algorithms to determine the end-to-end route across the optical segment of the DCN. An additional operative mode for the PCM has been also developed to allow for the joint computing and network resource provisioning according to decisions taken at the orchestration layer (mainly the Algorithms module). In such a case, the PCM, instead of calculating the routes, requests them directly to the Algorithms module at the orchestrator through a dedicated interface, which responds with the route details according to the decided slice mapping.

Lastly, to guarantee an overall optimization of the DC resources, both computing and network, a novel orchestration layer in aims to coordinate the provisioning of resources for

virtual slices is provided. The main purpose of such layer is to act as a Virtual Infrastructure Manager (VIM), administrating the underlying control and data layer as well as enabling the cooperation among infrastructure controllers and resource configuration software to deliver complex services towards upper layers (i.e. the application/service layer). Several orchestrator and VIM platforms can be found, both commercially (e.g. Amazon AWS) or in open source/community formats (e.g. CloudStack). In the present architecture, OpenStack is used as the platform of choice [35]. Through a common Shared Services layer, the different OpenStack services can consume DC resources, easily leading to resource provisioning automation. Nevertheless, like in the control layer, to fully exploit the optical medium while providing the necessary means for dynamic and automatized slice provisioning, extensions to some OpenStack services have been developed, while introducing some new functionalities and modules.

Starting from the employed core OpenStack services they mainly are: Horizon, which is the canonical implementation of OpenStack web-based dashboard, allowing requesting any kind of virtual resources (e.g. VMs, networks, etc.). In the current implementation, Horizon has been extended to support the graphical definition of slices (resources and topology). A REST interface has been developed to support Create, Read, Update and Delete (CRUD) operations of slice instances, interconnecting the extended dashboard with the newly designed Algorithms module, which will be detailed later; Nova, which is the service responsible for the whole VM provisioning process; Neutron, which is employed for the instantiation of the Layer 2 logical network associated to the slice, interconnecting the VMs so they can exchange IP traffic; and Heat, which is OpenStack orchestration service and coordinates the communication among other subordinate services (such as Nova and Neutron) to correctly infer the slice resource dependency and instantiate them in order.

However, Heat by itself does not provide any capability to decide on the particular mapping of the virtual resources onto the physical ones. Additionally, Heat is completely unaware of the peculiarities of the underlying physical network infrastructure. For all these reasons, we introduced a newly designed Algorithms module, whose main purpose is to determine the optimal mapping of slices onto the physical infrastructure and interact with OpenStack and ODL for resource provisioning. At its core, the Algorithms module implements several algorithms that calculate the allocation of slice resources onto the DC infrastructure. Thanks to them, an overall optimization of the resources placement is achieved, completely adapted to the particularities of the underlying data plane, and hiding all the interactions with OpenStack and the control layer from the slice requester. A database is employed to give persistence to the successful slice requests as well as store details about the physical route selection for the virtual links, which will be later consumed by the control layer upon request.

With such software layers, it is possible to automate the provisioning of slice instances across the DC infrastructure. A schematic of the provisioning workflow is detailed in Figure 3. Such process starts from the orchestrator, particularly at the dashboard, which provides a graphical interface to configure the features of the desired slice (step 1). Then, the dashboard contacts the Algorithms module (step 2), which uses topological information previously collected from the TM of the Controller and Nova to compute an optimized slice layout. The Algorithms module then triggers the creation of the stack (i.e. the collection of

virtual resources, such as the IP network and the VMs) by contacting the Heat module of OpenStack, passing down a template containing the details of the stack (step 3). With this, Nova deploys the IT infrastructure, i.e. the VMs (step 5), and Neutron sends the network information to the VTN at the controller to create the IP network associated to the slice (step 4). At this point, however, the physical paths associated to the virtual links interconnecting the VMs have not been configured yet. Following the SDN paradigm, these paths are configured when traffic exchanges (e.g. IP) between VMs are required. Once the first traffic packet reaches the OVS instance, the packet is sent to the Controller where it is processed by the VTN (steps 6 and 7). The VTN contacts the ORVM, which triggers the configuration of the physical paths associated to the slice (step 8). The ORVM requests the path provisioning to the OPM, which in turn requests the path to the PCM (step 9). In our orchestrator-based approach, instead of computing the path, the PCM requests it to the Algorithms module (step 10). Finally, the OPM configures the flows in the physical network devices involved in the path by means of the OF plugin module (step 11b) and the VTN configures the OVS instances at the servers (step 11a).

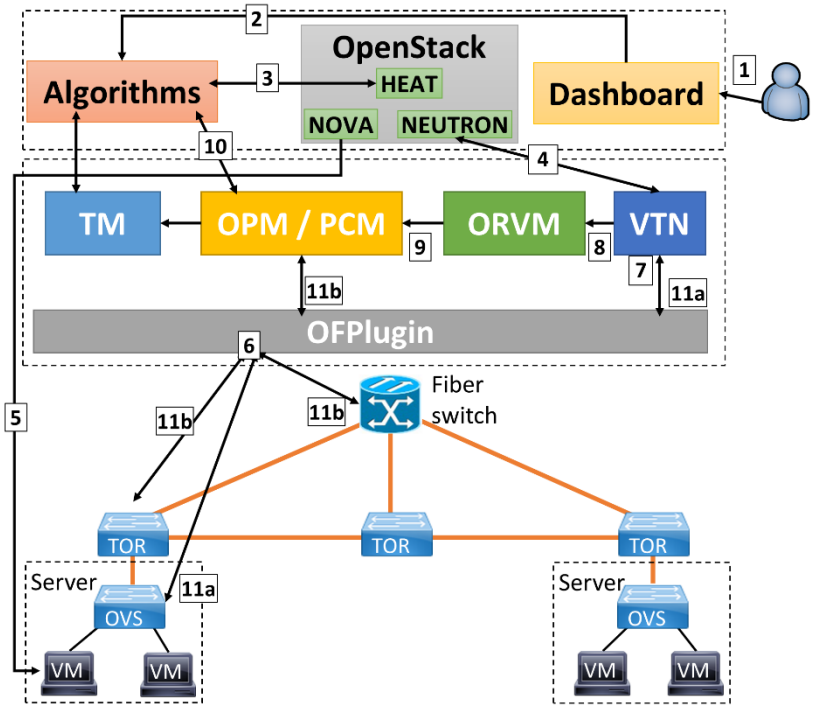


Fig. 3. Workflow for slice provisioning.

A. **Slice Embedding Optimization Algorithm**

Besides the automated slicing provisioning, one of the key points of the presented architecture is the inclusion of a novel Algorithm module, which allows for the optimal slice embedding onto the physical substrate (i.e. the resource selection). In order to overcome the limitations posed by traditional embedding mechanisms, where resource allocation for VMs is done without taking into account the status of the network, the developed algorithm incorporates network awareness onto the embedding decisions. Moreover, to tailor the mapping of the slices to the presented optical network and to the characteristics of the slice

(mainly virtual link bandwidth), the algorithm select the most suitable optical path according to the current status of the network resources. With these considerations in mind, in this section, we detail the mechanism employed to decide the mapping of the slices. Pseudo-code 1 summarizes the main actions taken by the algorithm. Before proceeding with the explanation of the pseudo-code, let us introduce the main employed notation, being $G = (N, E)$ the graph of the DCN, with N the set of network nodes (ToRs and fiber switches) and E the set of fiber links; d the slice request, represented by the virtual graph $G_v = (N_v, E_v)$, with N_v the set of VMs and E_v the set of virtual links. Each VM requests for a number of CPU cores, memory and storage equal to C_{n_v}, M_{n_v} and H_{n_v}, respectively, while each virtual link requests for a bandwidth equal to B_{e_v}; R as the set of racks in the DC, connected to the DCN through the ToRs; and S_r the set of servers in a rack, with each server being equipped with a total number of CPU cores, memory and storage equal to C_s, $M_s$and H_s, while C_s^a, M_s^a and H_s^a account for the available resources at a given time. Lastly, we define δ_n^+ and $\delta_{n_v}^+$ as the set of outgoing links from physical node n and virtual node n_v, respectively, and B_e^a the available bandwidth of physical link e at a given time.

Algorithm 1: Slice allocation algorithm pseudo-code

Input: d, G
Output: Slice mapping
Phase 1: Pre-processing
$G \leftarrow$ populate with status and topological information
$P \leftarrow$ end-to-end paths in G among all pairs of ToRs
Phase 2: Slice mapping
Phase 2.a: VM mapping
Sort VMs in descending order according to most restrictive resource requested
$R_d \leftarrow \emptyset$ *//Racks employed to map slice*
For each n_v in N_v **do**
  Sort racks in R in descending order according to $\Phi_{n_v}^r$
  mapped ← false *//Boolean to indicate if the VM has been mapped*
  For each r in R **do**
    If not mapped **and** r not in R_d **then**
      Find server s in S_r with available resources that minimizes $\Delta_{n_v}^s$
      If found **then**
        mapped ← true
        add r to R_d
        add n_v mapping to total slice mapping
        update server status
If all VMs are mapped **then**
  Phase 2.b: Virtual link mapping
  For each e_v in E_v **do**
    If B_{e_v} greater or equal than 50% port capacity **then**
      Find candidate path in P including only fiber switches
      If found **then**
        add e_v mapping to total slice mapping
        update network status
      Else
        Find candidate path in P including only ToR switches
        If found **then**
          add e_v mapping to total slice mapping
          update network status
    Else
      Find candidate path in P including only ToR switches
      If found **then**
        add e_v mapping to total slice mapping
        update network status
  If all virtual links are mapped **then**
    Return slice mapping
  Else

```
36:         Slice blocked
37:   Else
38:         Slice blocked
```

Essentially, the algorithm is based on an adaptive greedy procedure. In more detail, the algorithm firstly constructs the DCN graph, including the available servers connected at the ToRs, gathering information from Nova at OpenStack and the TM at ODL. Once constructed, the algorithm calculates the set of candidate paths in the DCN employing a K-Shortest Path (SP) routing mechanism, using the length of the paths in hops as the metric. A Depth First Search (DFS) procedure is employed to determine the routes. To deal efficiently with multi-fiber scenarios, a variation of the DFS is employed, in which the exploration is not done through individual links but instead through the fiber bundles between adjacent nodes. Then, the distinct routes are calculated as a combination of the sequence of bundles and the particular fiber within it. Once done, it proceeds with the slice mapping.

As a first step, the VMs requested by the slice are sorted in descending order considering the most restrictive resource in terms of server occupation, that is, the VMs that have the highest resource demand when compared to the nominal capacity of the servers precedes in the ordering. Once ordered, the mechanism iterates through them to find an appropriate mapping satisfying their resource requirements. As a first step, the average aggregated available capacity in terms of computing resources (CPU cores, storage and memory) per rack is calculated. Additionally, the outgoing available network capacity from the rack is also computed. The two values are then combined following a weighted product, accounting also for the resource requirements imposed by the VM under consideration. The exact way in which the metric is calculated is shown below:

$$\Phi_{n_v}^r = \alpha \cdot T_{n_v}^r + \beta \cdot W_{n_v}^r$$

with:

$$\mathrm{T}_{n_v}^r = \begin{cases} -\infty \; if \sum_{s \in S_r} C_s^a < C_{n_v} \; or \sum_{s \in S_r} M_s^a < M_{n_v} \; or \sum_{s \in S_r} H_s^a < H_{n_v} \\ 0 \; if \sum_{s \in S_r} C_s^a = C_{n_v} \; or \sum_{s \in S_r} M_s^a = M_{n_v} \; or \sum_{s \in S_r} H_s^a = H_{n_v} \\ \frac{1}{3}\left(\frac{\sum_{s \in S_r} C_s^a - C_{n_v}}{\sum_{s \in S_r} C_s^a} + \frac{\sum_{s \in S_r} M_s^a - M_{n_v}}{\sum_{s \in S_r} M_s^a} + \frac{\sum_{s \in S_r} H_s^a - H_{n_v}}{\sum_{s \in S_r} H_s^a} \right) \; otherwise \end{cases}$$

$$\mathrm{W}_{n_v}^{r} = \begin{cases} -\infty \; if \sum_{e \in \delta_n^+} B_e^a < \sum_{e \in \delta_{n_v}^+} B_{e_v}^a \\ 0 \; if \sum_{e \in \delta_n^+} B_e^a = \sum_{e \in \delta_{n_v}^+} B_{e_v}^a \\ \dfrac{\sum_{e \in \delta_n^+} B_e^a - \sum_{e \in \delta_{n_v}^+} B_{e_v}^a}{\sum_{e \in \delta_n^+} B_e^a} \; otherwise \end{cases}$$

With such metric, the racks are then sorted in descending order. In such a way, the algorithm prioritizes the most free racks while also accounting for the availability of network resources, which, if lacking, would lock the whole embedding procedure. The parameters α and β are employed to tune the sorting of the racks depending on the actual allocation policy in the DC. Once sorted, the algorithm iterates through them and selects the first available server within the rack that minimizes the following metric:

$$\Delta_{n_v}^{s} = \frac{1}{3}\left(\frac{C_s^a - C_{n_v}}{C_s} + \frac{M_s^a - M_{n_v}}{M_s} + \frac{H_s^a - H_{n_v}}{H_s}\right)$$

In this regard, the algorithm minimizes the number of occupied servers while searching for the server that provides the tightest fit for it. In this process, the algorithm avoids mapping two VMs belonging to the same slice onto servers of the same rack. In more detail, looking at the pseudo-code, the algorithm firstly initializes a set named R_d, whose purpose is to track the different racks employed to map the slice instance, thus, avoiding their repetition. This is mainly done to provide some degree of resilience against rack failures, thus rack diversity must be encouraged. If all the VMs are successfully mapped, it proceeds with the mapping of the virtual links.

For this, it iterates over all virtual links of the slice instance aiming to find enough lightpaths to satisfy their bandwidth requirements. For a particular virtual link, it seeks the candidate path set P for the physical paths that connect the racks onto which the endpoints of the virtual links have been mapped. For the path selection, the algorithm checks the requested bandwidth by the virtual link. If it is superior to the 50% of the maximum outgoing capacity at the ToR optical ports, a path incorporating only fiber switches as intermediate nodes will be prioritized. Otherwise, optical paths containing only ToRs as intermediate nodes will be selected. Note that, for the first case, if such available path is not found, a path containing only ToRs will be selected as an alternative, to avoid unnecessary virtual link blocking. With such considerations, the mechanism iterates through all virtual links and potential candidate paths and selects the path complaint with the preferred technology and the availability of end-to-end bandwidth. Once all virtual links are mapped, a satisfactory mapping of the slice has been found and its details are returned.

In what follows, we will provide a time complexity analysis of the proposed heuristic, considering the internal operations and mechanisms employed. As a first step, the algorithm computes all the candidate physical routes between rack pairs in the DC. This translates to having $\frac{|R|\cdot(|R|+1)}{2}$ different source-destination pairs for which route calculations are needed.

Considering that a DFS procedure is employed for the route calculation, for which the average time complexity is equal to $\mathcal{O}(|N| + |E|)$, the time complexity of the route calculation between rack pairs can be approximated to $\mathcal{O}\left(\frac{|R|\cdot(|R|-1)}{2} \cdot (|N| + |E|)\right)$.

Then, the algorithm proceeds with computation of the slice mapping, which is divided in two main phases: VM mapping and virtual link mapping. For the VM mapping, the algorithm first sorts the VMs according to the most restrictive resource that they are requesting, as explained before, employing the well-known Timsort procedure. Next, it essentially iterates among the different VM, rack and server combinations to find a suitable server to deploy the VM, sorting in each iteration the racks according to the previously described metric. Thus, the time complexity of the VM mapping can be approximated as $\mathcal{O}\big(|N_v| \cdot \log|N_v| + |N_v| \cdot (|R| \cdot \log|R| + |R| \cdot |S_r|)\big)$.

As for the virtual link mapping, the algorithm iterates through all virtual links to find a suitable candidate physical path to fulfil their bandwidth requirements. Defining as $\overline{P_{e_v}} \subseteq P$ the average set of candidate paths per virtual link, the algorithm has to repeat this operation a number of times equal to $|E_v| \cdot \left|\overline{P_{e_v}}\right|$. Then, for each of the candidate paths, the algorithm has to check for the available bandwidth at the path, which translates on having to check the available bandwidth for all physical links in the path, requiring at most $\overline{h_p}$ operations, where $\overline{h_p}$ denotes the average length in hops for a path between racks. Thus, the complexity of the virtual link mapping phase can be approximated to $\mathcal{O}\big(|E_v| \cdot \left|\overline{P_{e_v}}\right| \cdot \overline{h_p}\big)$.

Taking into account all the steps involved, the time complexity of the proposed heuristic can be approximated as $\mathcal{O}\left(\frac{|R|\cdot(|R|-1)}{2} \cdot (|N| + |E|) + |N_v| \cdot \log|N_v| + |N_v| \cdot (|R| \cdot \log|R| + |R| \cdot |S_r|) + |E_v| \cdot \left|\overline{P_{e_v}}\right| \cdot \overline{h_p}\right) \approx \mathcal{O}\left(\frac{|R|^2}{2} \cdot (|N| + |E|) + |N_v| \cdot \big(\log|N_v| + |R| \cdot (\log|R| + |S_r|)\big) + |E_v| \cdot \left|\overline{P_{e_v}}\right| \cdot \overline{h_p}\right)$. It can be observed that the performance of the proposed heuristic is polynomial and is tightly related to both the size of the physical infrastructure (with special emphasis on the number of racks) and the average size of the slice requests.

IV. EXPERIMENTAL VALIDATION AND DISCUSSION

As highlighted in previous sections, the main aim of the presented architecture is the automated provisioning of virtual slices, with joint configuration of the requested computing and network resources for enhanced utilization of the physical DC infrastructure. For this, in this section we present several experiments and simulations to demonstrate the correct operation of all involved layers (data, control and orchestration) as well as the benefits provided by the joint orchestrated approach. The evaluation is structured in three phases: a first one, where the performance in regards of connectivity between servers for the optical data plane is evaluated, employing a real small-scale physical infrastructure. Next, a second one, employing the same physical testbed, which focuses on the functional validation of the overall slice provisioning process. Finally, to conclude the study, a third phase of tests focuses onto the scalability of the orchestrated resource provisioning as well as on the

enhanced slice acceptance due to the joint resource configuration when compared to legacy solutions. To perform these last tests, due to the limitations on the available physical hardware of the small-scale DC testbed, bigger scenarios will be tested through emulated environments.

A. **DCN Performance Evaluation**

Before proceeding on detailing the performed tests, we introduce the employed physical small-scale DC testbed. It consists on an optical DCN constituted by three ToRs connected to a central fiber switch in a star fashion by optical fiber links. All neighboring ToRs are also connected among them through fiber links. Each ToR has a 64 radix with a maximum throughput per outgoing optical port of 10 Gb/s, while a 24x24 port fiber switch is employed to interconnect them. Then, a set of two servers is connected to such DCN through Ethernet links, one server connected to a single ToR. Servers are equipped with an instance of an OpenStack computing node, allowing for the deployment of VMs on top of them. Lastly, a third server is employed to run an instance of the control and orchestration layers software. To enable the configuration and orchestration of resources, this server is connected to the OF-agents at the DCN devices and the rest of servers through a dedicated Ethernet control network running in parallel. Figure 4 depicts a schematic of the set-up while Figure 5 shows a view of the physical hardware and the main components of the testbed.

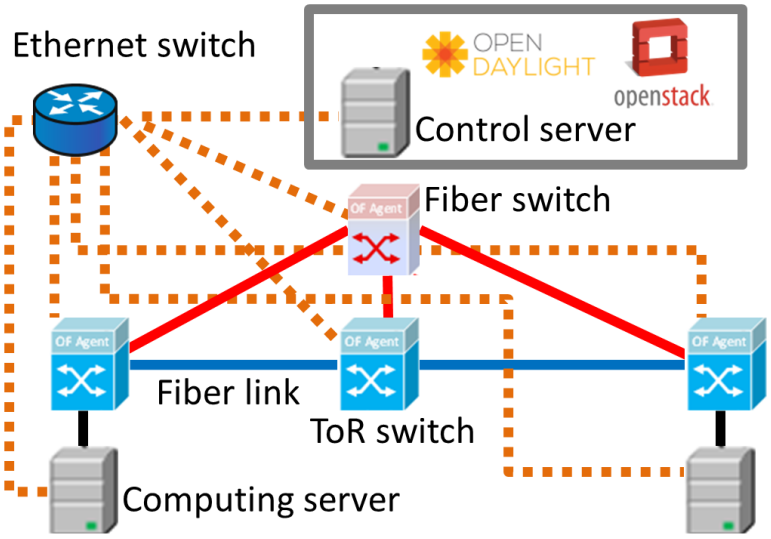


Fig. 4. Schematic of the experimental set-up.

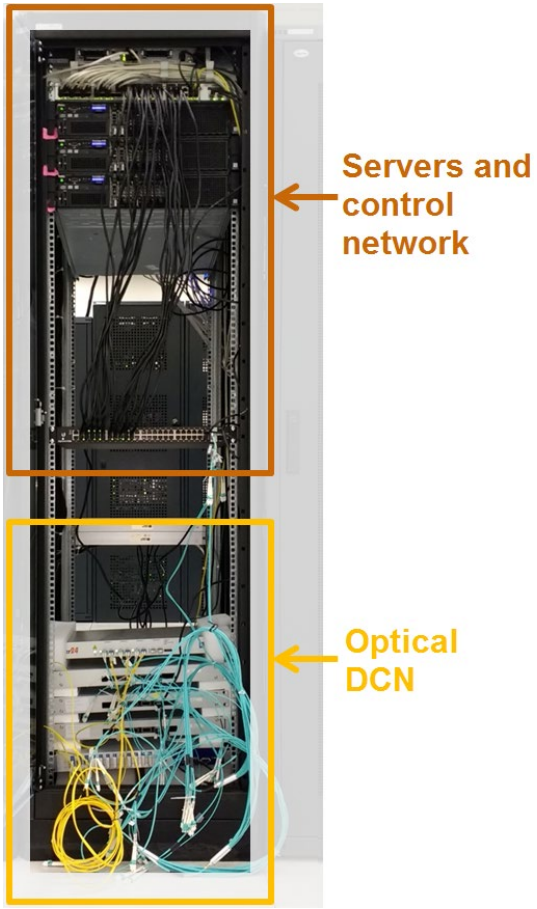


Fig. 5. View of the physical hardware.

With such small-scale testbed, the correct functional behavior of the presented architecture will be demonstrated. As mentioned, as a first step, the performance of the optical DCN is analyzed in regards of inter-server connectivity. More specifically, latency (Round Trip Time (RTT)) and throughput tests between servers have been performed. To this end, two sets of tests focusing on pre-configured paths in the network have been executed. For the first one, the servers communicate through a path containing only the three ToR switches, while for the second one, a path containing the optical fiber switch is configured, skipping the intermediate ToR switch (as depicted in Figure 4). To test the RTT, exchanges of ICMP messages from the servers have been performed, while for throughput tests the Iperf tool is employed, which enables sending traffic among servers in the form of TCP sessions. Ten different repetitions of the tests have been executed. For them, Figure 6 reports the maximum, average and minimum experimented RTT and throughput for both types of configured paths. For the sake of space, only the results in one sense of the communication among servers are reported, although tests in both directions have been performed. Nevertheless, let us note that results in both senses are similar, as to be expected due to the symmetry of the employed DCN topology.

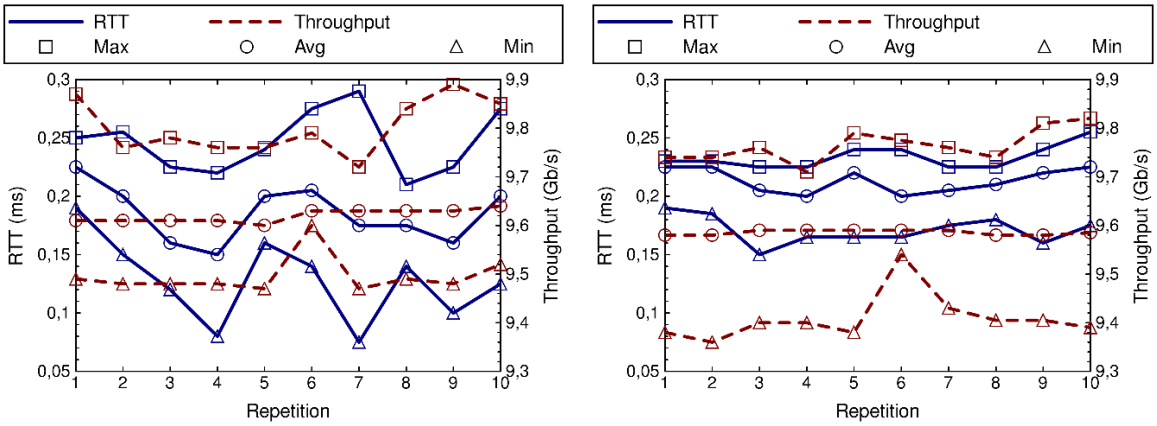


Fig. 6. Maximum, average and minimum experimented RTT (solid) and throughput (dashed) for pure ToR path between servers (left) and for path between servers through fiber switch (right).

It can be appreciated that the average for the RTT is in the order of 0.2 ms, while the maximum values are in the order of 0.25 ms for both types of paths while for the end-to-end throughput the average is in the order of 9.6 Gb/s, while the maximum values are in the order of 9.8 Gb/s, again for both types of configured paths. Note, however, that for the path involving the fiber switch the obtained values are fairly more stable among repetitions as well as differences between maximum, average and minimum values being smaller when compared to the pure ToR path. This is because the fiber switch is completely transparent to the traffic flowing through its ports, not requiring extra processing once the cross-connections have been configured. Nevertheless, for all types of switches it can be appreciated how the obtained latency is very small (less than 1 ms), while the throughput is close to the maximum theoretical possible throughput per port, highlighting the high performance of the DCN fabric for server-to-server communications.

B. **Architecture Functional Evaluation**

In the following round of tests, we will focus on demonstrating the automated slice provisioning on top of a shared physical hardware layer, enabled through the collaborative efforts of all the layers of the presented architecture. To this, the same testbed as before is employed. As a starting point, to demonstrate the correct connection of the OF-enabled optical devices to the control layer, their classification and the construction of the DCN topology, we depict the graphical view of the data plane as represented at the ODL DLUX web interface (Figure 7). It can be appreciated how all optical nodes (ToRs and fiber switch) are properly identified and classified. Additionally, all network adjacencies (i.e. the fiber links) are properly represented. Note that also the physical compute servers are represented in the network as host nodes, each one connected to the corresponding ToR. Finally, it can also be noted that ODL detects the OVS instances running at the compute servers, one per compute, since OVS nodes are also managed through the OF protocol.

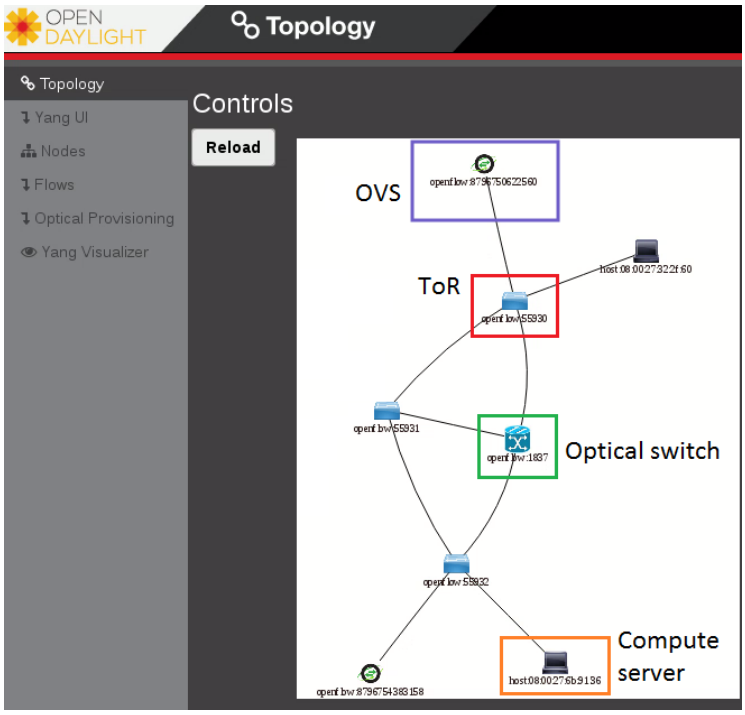


Fig. 7. Testbed as seen at the ODL DLUX graphical interface.

For the next step, a virtual slice request will be created and instantiated. As explained before, such process starts at the extended Horizon dashboard web interface, where it is possible to graphically specify the desired slice, both in terms of topology and resources. In particular, we have requested a slice consisting on two VMs interconnected through a virtual link requesting the maximum available bandwidth at the data plane, that is, 10 Gb/s. Once specified, the slice details are communicated to the orchestration plane, more particularly, to the Algorithms module, which will then compute the mapping of the slice onto the physical resources (servers and network paths). Since the whole provisioning process has already been explained before, we will focus the attention on the resulting network configuration to interconnect the deployed VMs of the slice, which are deployed one per server. To interconnect the servers, an optical path needs to be configured. The configuration of the paths is performed by the control layer according to the decisions taken at the Algorithms module. In the particular case at hand, only a single path needs to be configured, which will cross the fiber switch since maximum bandwidth is requested. To demonstrate the correct configuration of the necessary cross-connection at the fiber switch, Figure 8 depicts a capture of the CFlow Mod message stating the action and the ports to be connected. Once the path is configured, the slice is ready to be operated.

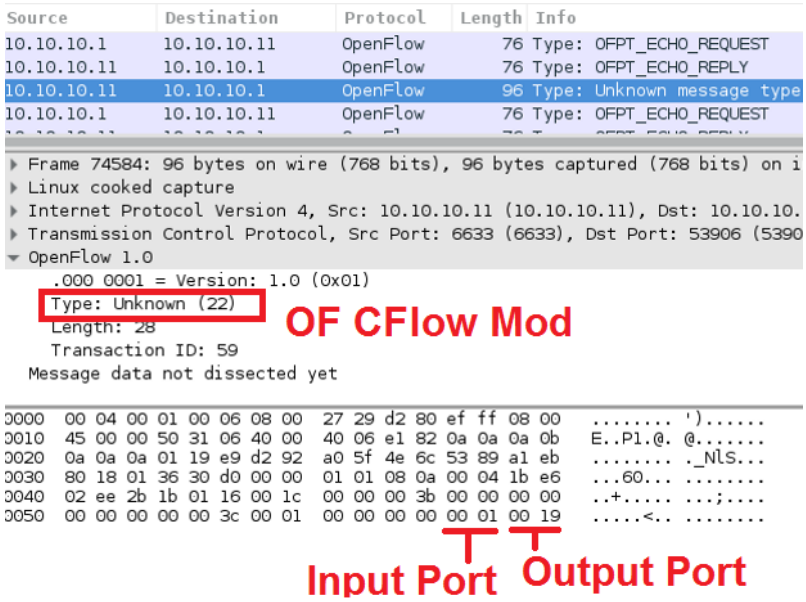


Fig. 8. Wireshark capture depicting a CFlow Mod message for the configuration of the optical fiber switch.

C. **Architecture Performance Evaluation**

To finalize the evaluation of the presented architecture and to highlight the benefits of an orchestrated approach for virtual slice provisioning, we compare our solution with legacy provisioning mechanisms where the allocation of computing and network resources is done in a non-joint fashion. To this end, and in order to obtain meaningful results, bigger DC scenarios are needed. For this, the Mininet emulation tool [36] is employed to construct them. Additionally, an external slice generator has been programmed in order to perform more agile tests. The generated slices are sent directly to the Algorithms module, bypassing the Horizon dashboard. Additionally, due to hardware limitations, servers at the DC will be also emulated. Thus, in the following set of experiments no VMs will be deployed. Note, however, that the mapping and allocation process still accounts for the server occupancy, hence, the overall slice provisioning process is essentially as explained before.

Given this setup, three data plane scenarios have been configured, in which several clusters of racks are present. In each cluster, each rack is connected to its corresponding ToR switch through Ethernet cables, allowing for the intra-rack communication. Then, ToRs in the cluster are connected to a central fiber switch, one per cluster, which, in turn, are connected in a ring fashion between them. Figure 9 depicts an example of the employed DCN distribution. Given this, the three configured scenarios are: 1) a DC composed of four clusters, with each cluster consisting in 8 racks, with all pairs of optical nodes at the topology connected through 19 fiber link pairs; 1) a DC composed of six clusters, with each cluster consisting in 16 racks, with all pairs of optical nodes at the topology connected through 10 fiber link pairs; and 3) a DC composed of nine clusters, with each cluster consisting in 16 racks, with all pairs of optical nodes at the topology connected through 10 fiber link pairs. The characteristics of the ToR switches are the same as in the previous section while 192x192 port optical fiber switches are configured for these scenarios. As for the servers, it is assumed

that each rack is equipped with 40 servers, with each server having 8 CPU cores, 16GB of memory and 1TB of disk available for VM instantiation.

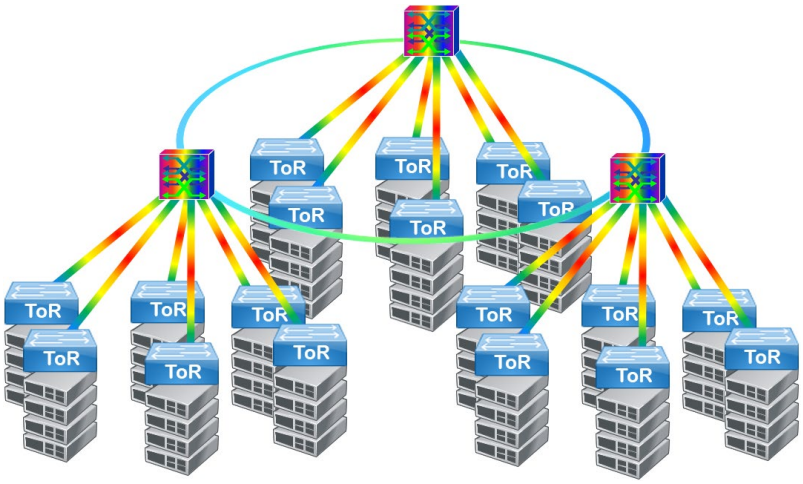
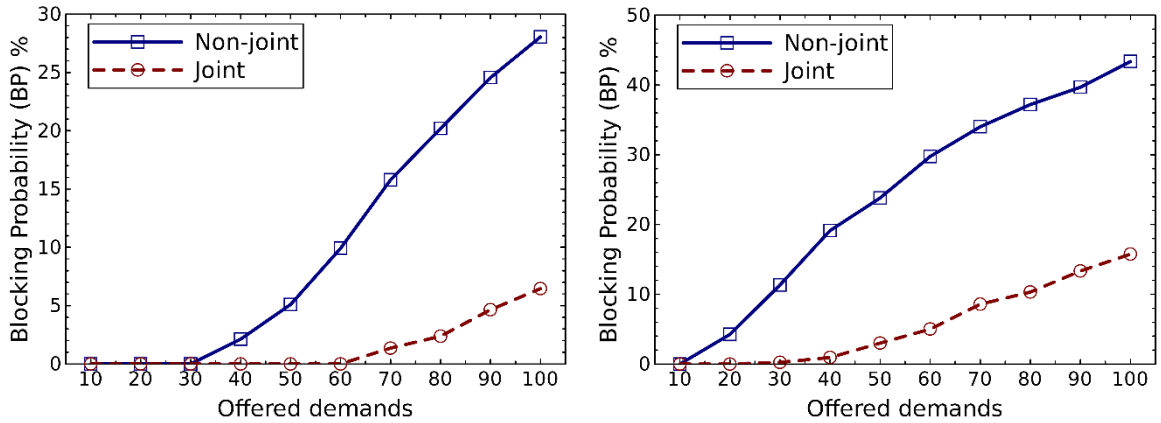


Fig. 9. Example of employed DCN distribution for simulation scenarios.

The following set of results assumes that a slice demand set has to be allocated concurrently at the DC physical infrastructure, with an arbitrary long life time, thus no slice deletion will be performed (a static scenario is considered). Thus, the Algorithms module will calculate sequentially the mapping of the slices in the set according to the current status of the available resources. For the two considered allocation scenarios (joint and non-joint allocation), we analyzed the differences on slice acceptance for increasing sizes of the offered demand set. In particularly, slices are assumed to consist on between 2 and 5 VMs interconnected with virtual links in a full mesh fashion. The characteristics of the VMs are chosen among the default OpenStack flavors [37]. These VMs are interconnected with virtual links with a requested bandwidth chosen among the set {10, 100, 1000} Mb/s. Figure 10 depicts the obtained results, with each data point being averaged over 100 different random instances of the slice set, having set $\alpha = \beta = 0.5$.



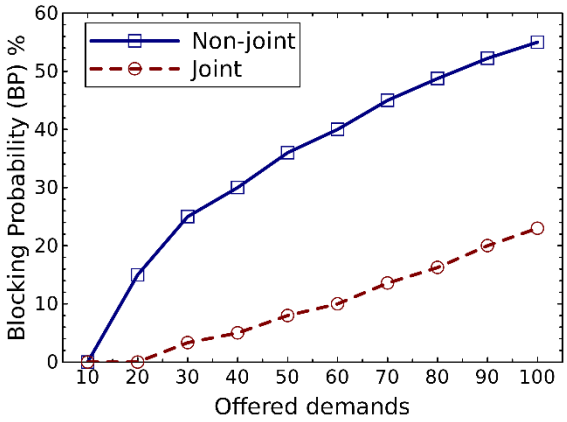


Fig. 10. Slice blocking probability as a function of the demand set size for DCN scenarios 1 (top-left), 2 (top-right) and 3 (bottom)

It can be appreciated how a joint provisioning allows for a higher number of accepted requests, reducing the slice BP up to around 60-80% in the worst cases when compared to legacy solutions. Additionally, we can appreciate how the BP is slightly higher for the larger configured DCN configured. This is mainly due to the larger hop count of end-to-end paths. In such situation, blocking due to lack of network resources increases. In fact, after having analyzed the reasons for the experienced slice blocking, it resulted that the totality of the blocking is due to the lack of network resources. More in particular, the blocking due to network resources happens at the optical switches when available cross-connections are exhausted, locking the potential combinations of input-output fiber ports. When such an event happens, several route options between racks are locked out, increasing the chances of blocking future slice demands, specially in scenarios in which demands require a high network connectivity.

In light of these results, it becomes evident that more taxing scenarios in regards of network resources to be deployed impose more limitations onto the acceptance of virtual slices. To analyze this, we have also extracted results particularizing the characteristics in terms of the number of VMs at the slices requests. Particularly, two scenarios are configured: a) each slice requests between 2 and 3 VMs; and b) each slice requests between 4 and 5 VMs. For each of them, we have particularized the size of the demand set to 60 requests. The obtained results are depicted in Figure 11, focusing on the smallest and largest tested DCNs (DCN1 and DCN3). Such results confirm the superior performance of the orchestrated joint resource provisioning. It can be appreciated that in situations with more meshed virtual slices (scenario b), the non-joint approach performs very poorly, mainly because a significant number of end-to-end paths cannot be realized due to not having enough optical network resources due to the selection of the servers, which does not account for the network status. On the other hand, in the joint approach this effect is mitigated, thanks to the network awareness of the mapping approach, which selects the placement of the VMs considering the optical resources status.

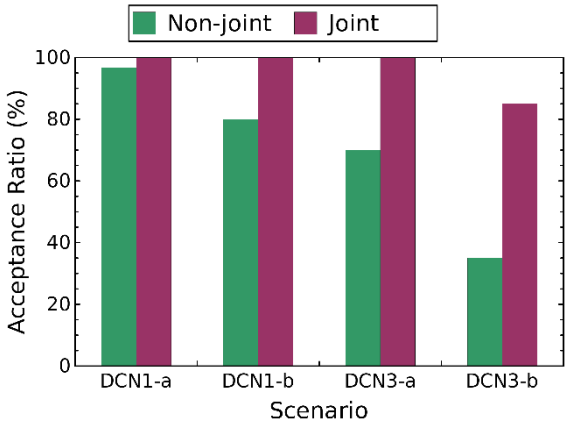


Fig. 11. Slice acceptance comparison as a function of the slice configuration.

To conclude our studies, we proceed on analyzing the slice provisioning times for both non-joint and joint approaches. To this end, we have focused on the scenarios employed for Figure 11, that is, considering the emulated optical data planes. Note, however, that aside from this fact, all orchestration and control software is still present (depending on the scenario), thus the provisioning times do account for the interactions across all involved software layers, their internal operations and the configuration of the emulated network nodes through the SDN controller. To better highlight the differences between both approaches, besides the total time, we also disclose the time components of both the control and orchestration layer for the joint approach. Figure 12 depicts the obtained results. The obtained times reflect first that the introduced overhead due to the joint resource provisioning (mainly due to the algorithms module) is fairly non-relevant (a maximum difference of around 1 s. is observed), considering the increased slice acceptance that can be achieved when compared to a legacy non-joint provisioning approach.

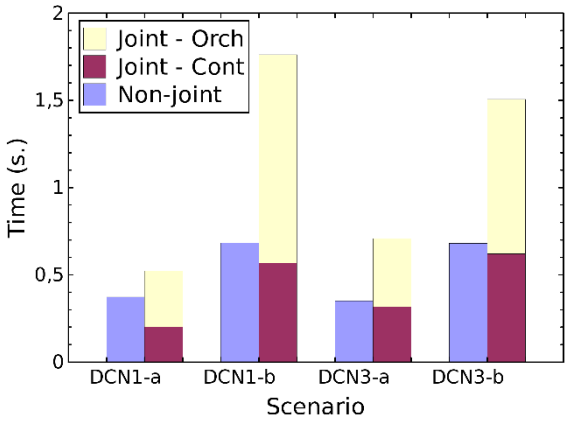


Fig. 12. Slice configuration times for both non-joint and joint approaches.

V. CONCLUSIONS

Slicing is seen as a promising solution towards future cloud and telecom infrastructures, allowing for the creation of complex virtual infrastructures fulfilling the specific needs of the customer and the applications/services that will run on top. Such a

concept is to be harnessed by future data centers (DCs). However, current technologies impose limitation on both traffic handling as well as on joint resource provisioning aspects. Thus, new architectural solutions with a vertical integration of novel control and management solutions as well as data plane technologies are a must to overcome such limitations. With this goal in mind, in the current paper we presented a DC architecture solution which encompasses state-of-the-art optical technologies for the realization of the intra-DC network (DCN) and both control and orchestration software layers to efficiently slice the underlying physical infrastructure.

The presented architecture has been properly tested in regards of the performance of the data plane technologies, highlighting the high bandwidth and low latencies that can be achieved thanks to optical technologies, as well as on the automated, dynamic and joint provisioning of computing and networking resources to satisfy the needs of virtual slices requests. In this regard, we showed how an orchestrated approach for resource provisioning allows for a higher number of virtual slices to co-exists when compared to non-joint legacy solutions, thanks to a better optimization of the underlying physical substrate. Indeed, we showed that in situations with more taxing network requirements for virtual slices, for example, more virtual links to be provisioned, non-joint resource provisioning severely limits the number of slices that can be deployed, due to the resulting poor resource efficiency. However, with the adopted orchestrated solution, such limitation is significantly overcame, without incurring in a significant time overhead in the overall provisioning time.

ACKNOWLEDGMENT

This work has been partially funded by the Spanish National project ALLIANCE (TEC2017-90034-C2-2-R) with FEDER contribution.

REFERENCES

[1] NGMN Alliance, “5G White Paper”, version 1, February 2015.
[2] ETSI, “Network Function Virtualization (NFV) White Paper”, October 2012.
[3] NGMN Alliance, “Description of Network Slicing Concept”, version 1, January 2016.
[4] Cisco White Paper, “Cisco Global Cloud Index: Forecast and Methodology, 2016-2021”, 2018.
[5] J. Perelló et al., “All-optical packet/circuit switching-based data center network for enhanced scalability, latency, and throughput”, IEEE Network, vol. 27, num. 6, pp. 14-22, December 2013.
[6] A. Pagès et al., “Strategies for Virtual Optical Network Allocation”, IEEE Communication Letters, vol. 16, num. 2, pp. 268-271, February 2012.
[7] D. Gmach et al., “Workload analysis and demand prediction of enterprise data center applications”, IEEE International Symposium on Workload Characterization (ISWC), Boston (USA), September 2007.
[8] N. Singh, S. Rao, “Energy optimization policies for server clusters”, IEEE International Conference on Automation Science and Engineering (CASE), Toronto (Canada), August 2010.

[9] X. Wang et al., “A resource management framework for multi-tier service delivery in autonomic virtualized environments”, IEEE Network Operations and Management Symposium (NOMS), Bahia (Brazil), April 2008.
[10] X. Sun at al., “Optimizing Resource Utilization of a Data Center”, IEEE Communications Surveys and Tutorials, vol. 18, num. 4, pp. 2822-2846, April 2016.
[11] B. Martini et al., “Design and evaluation of SDN-based orchestration system for cloud data centers”, IEEE International Conference on Communications (ICC), Kuala Lumpur (Malaysia), July 2016.
[12] S. Spadaro et al., “Resource orchestration in SDN-based future optical data centres”, International Conference on Optical Network Design and Modeling (ONDM), Cartagena (Spain), May 2016.
[13] M. Sekiya, at al., “Optical Network Softwarization: Virtualization and Software-Programmed Networking”, OSA Congress in Advanced Photonics 2016, Vancouver (Canada), July 2016.
[14] M. G. Rabbani et al., “On tackling virtual data center embedding problem”, IFIP/IEEE International Symposium on Integrated Network Management, Ghent (Belgium), May 2013.
[15] A. Wang et al., “Network virtualization: Technologies, perspectives, and frontiers,” IEEE Journal of Lightwave Technology, vol. 31, num. 4, pp. 523-537, February 2013.
[16] M. Chowdhury et al., “ViNEYard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping”, IEEE/ACM Transaction on Networking, vol. 20, num. 1, pp. 206-219, February 2012.
[17] L. Gong et al. “Novel Location-Constrained Virtual Network Embedding (LC-VNE) Algorithms Towards Integrated Node and Link Mapping”, IEEE/ACM Transactions on Networking, vol. 24, num. 6, pp. 3648-3661, December 2016.
[18] L. Gong et al., “Virtual Optical Network Embedding (VONE) over Elastic Optical Networks”, IEEE Journal of Lightwave Technology, vol. 32, num. 3, pp. 450-460, February 2014.
[19] Y. Wang et al., “Cost-Efficient Virtual Network Function Graph (vNFG) Provisioning in Multi-Domain Elastic Optical Networks”, IEEE Journal of Lightwave Technology, vol. 35, num. 13, pp. 2712-2723, July 2017.
[20] A. Pagès et al., “Planning of optical and IT resources for efficient virtual infrastructure embedding”, Proceedings of Photonics in Switching 2012 (PS 2012), Ajaccio (France), September 2012.
[21] Y. Tarutani et al., “Virtual network reconfiguration for reducing energy consumption in optical data centers”, IEEE/OSA Journal of Optical Communication and Networking, vol. 6, num. 10, pp. 925-942, October 2014.
[22] J. Yin et al., “Experimental Demonstration of Building and Operating QoS-aware Survivable vSD-EONs with Transparent Resiliency”, Optics Express, vol. 25, num. 13, pp. 15468-15480, 2017.
[23] Z. Zhu et al., “Build to Tenants' Requirements: On-Demand Application-Driven vSD-EON Slicing”, IEEE/OSA Journal of Optical Communication and Networking, vol. 10, num. 2, pp. A206-A215, February 2018.
[24] A. Aguado et al., “Dynamic Virtual Network Reconfiguration Over SDN Orchestrated Multitechnology Optical Transport Domains”, IEEE Journal of Lightwave Technology, vol. 34, num. 8, pp. 1933-1938, January 2016.

[25] W. Miao et al., “SDN-enabled OPS with QoS guarantee for reconfigurable virtual data center networks”, IEEE/OSA Journal of Optical Communication and Networking, vol. 7, num. 7, pp. 634-643, July 2015.
[26] P. Shuping et al., “Software-defined optical data centre networks”, China Communications, vol. 12, num. 8, pp. 1-9, August 2015.
[27] H. Yang et al., “Performance evaluation of multi-stratum resources integration based on network function virtualization in software defined elastic data center optical interconnect”, Optics Express, vol. 23, no. 24, pp. 31192-31205, November 2015.
[28] B. Kong et al., “Demonstration of Application-driven Network Slicing and Orchestration in Optical/Packet Domains: On-demand vDC Expansion for Hadoop MapReduce Optimization”, Optics Express, vol. 26, num. 11, pp. 14066-14085, 2018.
[29] R. Ruffini, “Multidimensional Convergence in Future 5G Networks”, IEEE Journal of Lightwave Technology, vol. 35, num. 3, pp. 535-549, February 2017.
[30] N. Calabretta et al., “System Performance Assessment of a Monolithically Integrated WDM Cross-Connect Switch for Optical Data Centre Networks”, Proceedings of 42nd European Conference and Exhibition on Optical Communications (ECOC 2016), Düsseldorf (Germany), September 2016.
[31] N. Parsons et al., “High Radix All-Optical Switches for Software-Defined Datacentre Networks”, Proceedings of 42nd European Conference and Exhibition on Optical Communications (ECOC 2016), Düsseldorf Germany), September 2016.
[32] OpenDaylight, https://www.opendaylight.org
[33] ONF, “OpenFlow swicth specification”, version 1.3.4, March 2014.
[34] ONF, “Extensions to the OpenFlow Protocol in support of Circuit Switching”, Addendum to OpenFlow Protocol Specification v1.0 - Circuit Switch Addendum v0.3, June 2010.
[35] OpenStack, https://www.openstack.org/
[36] Mininet, http://mininet.org/
[37] OpenStack, Manage flavours, https://docs.openstack.org/horizon/latest/admin/manage-flavors.html