Resource Orchestration in SDN-based Future Optical Data Centres

Salvatore Spadaro, Albert Pagès, Fernando Agraz, Rafael Montero, Jordi Perelló
Universitat Politècnica de Catalunya (UPC)
Advanced Broadband Communications Center (CCABA)
Barcelona, Spain
spadaro@tsc.upc.edu

***Abstract*—Data center (DC) infrastructures are a key element for nowadays Internet and cloud services. However, due to the increase of traffic that DCs have to manage, optical-based network architectures are being investigated to overcome the limitations of nowadays electronic-based intra-DC network (DCN) architectures. In this regard, it becomes imperative to coordinate the allocation of very heterogeneous resources (IT and optical network) for the efficient provisioning of services inside the DC. Such process is known as orchestration and allows the DC administrator to optimize the utilization of his physical infrastructure. In fact, resource orchestration has gained the interest of the industry and several software flavors have emerged to perform such operation. As a case of study, in this paper we focus on the provisioning of Virtual DC (VDC) instances belonging to different tenants over a shared DC infrastructure with optical DCN. To this end, an architecture for the orchestration and control of optically interconnected DCs in aims of efficient VDC provisioning is presented. The proposed solution is based on OpenStack for the orchestration and the concept of Software Defined Networking (SDN) for the control aspect of the network. To highlight the benefits of an orchestrated approach to DC resource management, several tests are performed, considering both static and dynamic VDC provisioning. The obtained results confirm the potential benefits of resource orchestration in DCs.**

Keywords—Data Center Networks; Virtualization; Orchestration

I. INTRODUCTION

Nowadays cloud platforms are experiencing a huge increase on the amounts of data that they have to handle, mainly driven by the emergence of bandwidth-hungry applications and paradigms such as Big Data and Internet of Things (IoT). This fact, coupled with the increasing need to handle to the end-user connectivity anywhere, anytime, with any device, has driven the telecom industry to push for the development of new infrastructures able to handle these requirements. In this scenario, the role of DC infrastructures arises of paramount importance. DCs allow for an efficient realization of complex cloud and Internet services through the collaborative efforts of the thousands of servers hosted in their premises. Nevertheless, due to the aforementioned increase on the traffic in the global cloud arena, DCs have experienced an enormous growth on the data that they have to handle, with around 75% of the traffic being handled inside DC infrastructures [1]. In such situation, current electronic-based intra-DCN network fabrics start to be a bottleneck in terms of scalability, latency and power consumption. For this reason, there is an increasing interest on bringing optical technologies inside the DCs due to their higher bandwidth and scalability as well as lower latencies and power consumption [2].

On the other hand, traditional telecom and cloud infrastructure owners have been focusing on offering services on top of the infrastructures that they own, with almost no control over the service from the user's part. This has led to a very rigid architecture incapable to adapt to dynamic traffic patterns, with limited capabilities in terms of infrastructure management and operation. In such a context, the concept of Infrastructure as a Service (IaaS) has been introduced [3], [4]. The rationale behind IaaS is to offer physical infrastructures as a service for exploitation by third party entities, giving the possibility to fully configure and manage the rented infrastructures, as if they were owned by those external entities. Additionally, IaaS is envisioned as a way to compose highly customizable infrastructures by merging physical resources from diverse natures, such as network resources, storage or computational capacities. The key enabling technology is the virtualization of the physical infrastructures. Thanks to virtualization it is possible to slice an underlying physical infrastructure into multiple virtual elements. Then, by federating and composing such virtual elements it is possible to create virtual infrastructures with very specific requirements in terms of resource characteristics, management and control.

Such a paradigm has resulted in the emergence of the so called VDC service (e.g. [5]). Essentially, a VDC is a case of IaaS where multiple external entities (hereafter referred as tenants) request to a DC physical provider to lease them a portion of his physical infrastructure in order to develop their own business models and offer services to end users on top of the rented infrastructure. A VDC is usually composed with computational resources, such as Virtual Machines (VMs), with a virtual network fabric interconnecting them with the desired capacity. Tenants execute their services and applications in this virtual infrastructure. The provisioning of a VDC instance entails basically two operations: node mapping and link mapping. The node mapping refers to the placement of the VMs onto physical servers while the link mapping consists on finding the necessary network resources that allow for the desired communication between VMs.

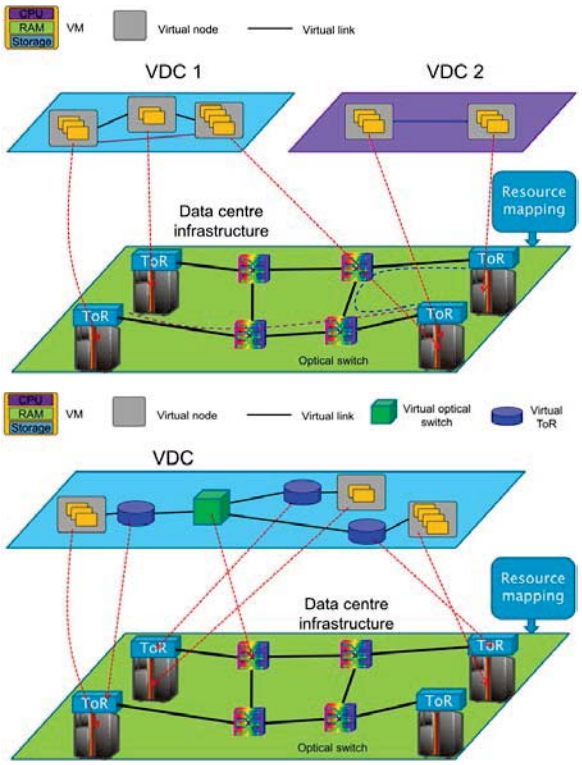

Fig. 1. Example of basic (top) and enhanced (bottom) VDC service scenarios [6].

Thanks to this, multiple VDC instances can coexist in the same physical infrastructure, each one of them having their dedicated resources without interfering with the other tenants. Fig. 1 (top) exemplifies this scenario, where multiple VDC instances are mapped on top of a DC infrastructure with optical DCN. Besides the presented VDC scenario, more sophisticated VDC services may be required in order to satisfy the needs of a broad range of tenants. For instance, finer control on the networking aspect of the VDC instance could be desirable as a way to optimize the utilization of the virtual slice. To achieve this purpose, it would be necessary to expose to the tenant the switching capabilities of the virtual slice, for instance, in the form of virtual switches [6]. By exposing these capabilities, the tenant can decide on how to route the traffic on his slice not only from the layer 3 perspective (e.g. IP), but also from the infrastructure point of view, achieving a higher customization and efficiency. Fig. 1 (bottom) depicts this scenario, considering the presence of virtual Top of the Rack (ToR) and optical switches. The exposure of the switching capabilities can be achieved by means of proper virtualization and slicing of the network resources and exposing the control of the obtained virtual resources towards the tenant by means of dedicated APIs.

Both presented scenarios require the provisioning of different types of resources (computational and network resources). In order to achieve an optimized VDC deployment, it is essential to coordinate both provisioning phases as well as to provide the necessary means to enforce programmatic control over the physical resources. This process, known as orchestration [7], is a primordial feature that modern DCs have to implement for efficient resource utilization. In fact, as reported in the last survey from Infonetics Research, during 2016 over 50% of the DC environments in production are planning to deploy orchestration solutions for resource management [8]. Following this trend, in the next section we present an SDN-based DC architecture which encompasses both a control and an orchestration layer in aims of supporting the provisioning of VDC instances.

II. SDN-BASED DC ARCHITECTURE

The proposed DC architecture is depicted in Fig. 2. Basically, it consists of three differentiated layers: data plane (bottom), control plane (middle) and orchestration plane (top). In more details, the data plane consists in several servers arranged in racks, with each rack having an opto-electronic Top of the Rack (ToR) switch that enables for the communication of the servers within the same rack. At its turn, the ToRs are interconnected thanks to an Optical Circuit Switching (OCS)-based network fabric. Thanks to the electronic capabilities of the ToRs, multiple communication flows may be groomed onto the same optical connection. Then, the optical connections are transparently switched by means of the intermediate OCS switches. At the destination ToR, an optical-to-electrical (O/E) conversion is performed and traffic is delivered to the corresponding destination server.

The intermediate layer corresponds to the SDN controller, which is, in brief, responsible for configuring the devices of the data plane. Finally, the orchestrator resides in the upper layer of the architecture and provides functionalities such as coordinated service provisioning according to the status of the physical infrastructure and the requests coming from external users. Next, more details about the architecture of the orchestrator and the control layers are provided.

A. *OpenStack-based Orchestrator Layer*

The orchestrator layer is based on the OpenStack software platform [9]. OpenStack is an open source DC management platform that allows for the orchestration of storage, computational and network resources. To achieve such purpose, OpenStack integrates several software modules and services, each one dedicated to a concrete task in the overall orchestration process (e.g. resource monitoring, service request, etc.). Thanks to the modularity of the OpenStack components, it is up to the DC administrator which ones to use and how they collaborate in order to achieve the provisioning of complex cloud services. In our proposed architecture, three OpenStack services are being employed, namely, Heat, Nova and Neutron.

Heat is the OpenStack orchestration service, which coordinates the overall provisioning process through communication with the different resource controllers (e.g. Nova, Neutron). Essentially, Heat is a template-driven service for describing and automating deployment of compute, storage and networking resources. Heat declarative mode takes form of a simple YAML template, a kind of markup language that allows specifying the parameters, resources and outputs of the request introduced in the template. When the template is deployed, a collection of infrastructure resources, known collectively as a stack, is instantiated, in the correct order and using the correctly inferred parameters. Depending on the type of resources Heat contacts the corresponding controller in order to instantiate them.

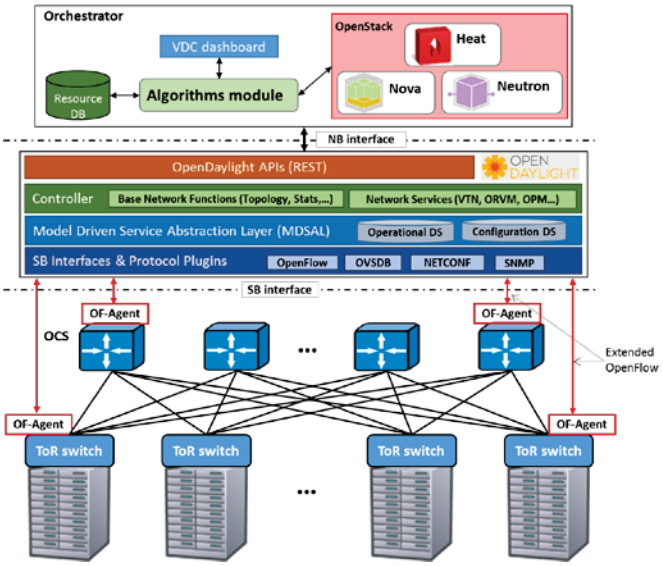

Fig. 2. SDN-enabled DC architecture.

For this, on the one hand, Nova is employed. Through Nova it is possible to specify the details of the VMs to be deployed (cores, memory, local storage, network interface capacity, operating system image, etc.) as well as the placement of the VMs in specific hosts or zones in the DC infrastructure and instantiate them. On the other hand, once the VMs have been deployed, Neutron is utilized to configure the network slice connecting them. Neutron is designed to manage the logical networks and to completely isolate the cloud user from the physical network considerations. For our purposes, it becomes essential that the optical capabilities of the underlying physical infrastructure are taken into account when provisioning the virtual networks. To this end, some extensions are developed in the Neutron service. These extensions consist of several APIs that allow the communication between Neutron and the network infrastructure controller, i.e. the SDN controller. In this way, the orchestrator can tap on the capabilities of the optical network, allowing for the provisioning of optical connections that will satisfy the needs of the VDC instance.

Apart from OpenStack, the orchestrator implements a dashboard module, which is the entry point for requests from tenants. Through the dashboard, a tenant can specify the desired VDC instance. Once specified, the details are sent to Heat for instantiation. Additionally, for a better optimization of the resource provisioning of VDC instances, we developed a new algorithms module, which seats between the dashboard service and the OpenStack Heat module. The main functionality of this module is to decide on the concrete mapping of the VDC resources onto physical ones. To this end, it fetches information from the infrastructure controllers (Nova and the SDN controller for computational and network resources, respectively). Such information is utilized to perform the mapping. In order to enable a more agile mapping process, some of the information may be stored into a resource database for further reference. Once the VDC mapping is decided, its details are fed to Heat in the form of a YAML template, which describes the characteristics of the VDC request and the desired placement for the requested resources. Some extensions on the template and on the Heat and Neutron modules are needed in order to cope with the provisioning of the virtual links of a VDC, since they are not a native OpenStack resource. For this, a new plug-in is designed for the virtual link resource. The purpose of the plug-in is to define the core characteristics of the resource as well as to provide the means to handle its whole lifecycle from the orchestrator perspective, that is, to perform Create, Read, Update and Delete (CRUD) operations.

B. SDN Controller Layer

The controller implementation is based on the OpenDaylight SDN platform in its Lithium release [10]. The OpenDaylight project is an initiative from the Linux Foundation that aims to develop an SDN-based control platform following the collaborative methods used by the open source community. From a functional perspective, the OpenDaylight controller is composed of a set of modules that cooperate to perform the set of tasks typically associated to the control layer (e.g. routing, signaling, etc.). More specifically, at the bottom of the controller, the Southbound (SB) interface implements the communication with the devices that compose the data plane. In a general case, this communication can be realized by means of different protocols such as OpenFlow (OF), NETCONF, SNMP, etc. Nonetheless, it is worth noting here that in our case all network devices can be configured through the OF protocol. The core of the controller relies on the SB interface to collect information about the data plane as well as to configure the requested network services over it. The basic functionalities implemented here are devoted to maintain the inventory of the network elements, the topology in which they are laid and interconnected, and the network services that are already deployed. The statistics collection service is also included here. The key entity in the core of the controller is the service abstraction layer (MD-SAL), which utilizes a set of data stores to keep a representation of the network elements at the control level. Hence, the functional modules of the controller perform their tasks by interacting with the data plane through the MD-SAL, which acts as a proxy. More advanced functionalities, such as network virtualization and monitoring, have been developed as well within the OpenDaylight project and can be dynamically installed in the controller. In the upper bound, OpenDaylight communicates with external applications and/or the orchestration layer through the Northbound (NB) interface, which is implemented by means of REST APIs.

For our purposes, an extended version of the OpenDaylight controller is required. First of all, extended versions of the OF protocol and plug-in able to cope with the specific requirements of the optical data plane are needed. In this way, the optical devices can be abstracted at the core of the controller (by means of the MD-SAL), so that the functional modules, such as the topology and inventory managers, can access them. In the same line, the topology and the inventory managers are extended to support the optical data plane infrastructure. Besides this, to implement the VDC allocation procedure, some network services have to be extended and new ones need to be implemented. First, an extended version of the Virtual Tenant Network (VTN) module is needed. In its current version, the VTN provides tools to virtualize packet-based electronic networks. However, due to the characteristics of the considered data plane scenario, the VTN needs to be extended to support

and virtualize pure optical transmission devices (i.e. OCS switches). To this end, the Optical Resource Virtualization Manager (ORVM) module has been developed and integrated with the VTN. The ORVM is responsible for creating and managing the virtual instances associated to the optical data plane devices. In particular, the ORVM keeps instances of virtual optical nodes and links. While a virtual optical node is defined as a subset of optical ports of the physical OCS switch, a virtual optical link is basically an optical path that connects two virtual nodes. In order to create a virtual link, the ORVM contacts the newly implemented Optical Provisioning Manager (OPM) module, which is responsible for creating optical paths over the physical data plane. Once created, the virtual optical resources are assigned to a single tenant who can further operate them.

C. VDC Provisioning Workflow

Having introduced the orchestrator and the control layer software architectures, we now detail the full workflow for the provisioning of VDC instances. The specific steps are as follows:

1. The tenant specifies the VDC request to be instantiated through the dashboard service of the orchestrator, giving the details about the desired resources: number of VMs, capacity of the VMs, virtual network topology and capacity of the virtual links. The VDC request is sent to the algorithms module (e.g., in a JSON file format).
2. Upon reception of the request, the algorithms module fetches information about the current status of the physical infrastructure. In concrete, it fetches server resource availability from Nova and network resource availability from the topology manager at the SDN controller.
3. With such information, the algorithms module executes an optimization algorithm in order to find the mapping of the VDC instance. Several algorithms may be available. The selection of the most proper one is performed by means of a set of policies, which are integrated in the orchestration platform.
4. Once the resource mapping is decided, the physical details of the virtual link mapping are stored in the resource database for further reference. The overall description of the VDC instance to be deployed is fed to Heat in the form of a YAML template. Heat contacts Nova and Neutron in order to configure the desired resources.
5. Nova creates the VMs as specified in the YAML template.
6. Neutron contacts the SDN controller which is the responsible to manage the creation and configuration of the virtual network infrastructure. This configuration is done in two phases. First, the ORVM is contacted to configure the virtual optical network. More precisely, the ORVM receives the optical path configuration associated to the virtual links to be created from Neutron. Afterwards, the ORVM contacts the OPM, which actually configures the optical paths in the data plane through the MD-SAL and the OF plug-in of the SB interface. Second, the VTN module is requested to create the virtual network infrastructure associated to the electrical network (i.e., the ToR switches).
7. When both phases are successfully finished, the controller updates the status of the network, which is kept in the inventory and topology managers. As a result, end-to-end connectivity between VMs is achieved and the whole virtual infrastructure is assigned to the tenant.

III. VDC PROVISIONING: A CASE STUDY

As a case of study, in this section we present an optimization algorithm to decide the optimal mapping of VDC requests. The proposed algorithm is executed within the algorithms module of the orchestrator layer. Thanks to it, it is possible to jointly map both the virtual nodes (VMs) and virtual links of the VDC instances, increasing the acceptance ratio of the requests.

Before detailing the algorithm, let us specify the problem under consideration. We consider a DC infrastructure with a set of servers $\mathcal{S}$ arranged in a set of racks $\mathcal{R}$. We assume that each server has a capacity in terms of VMs that can host equal to VM_s and a Network Interface Card (NIC) capacity equal to B_s, while each rack holds a total number of servers equal to $\mathcal{S}_r$. The racks are interconnected through an optical DCN represented by the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, with $\mathcal{N}$ the set of network nodes and $\mathcal{E}$ the set of fiber links. Specifically, $\mathcal{N}$ is composed by a set of ToRs $\mathcal{T}$, one per rack, and a set of OCS switches $\mathcal{O}$. We consider that each ToR is equipped with opto-electronic conversion capabilities, with electronic ports working at a bit-rate equal to B_s while optical ports work at a bit-rate equal to B_O. In this regard, we denote as L_{ToR} as the total number of ports of a ToR while we denote as L_{ToR}^{O} the number of optical ports of the same ToR. As for the OCS switches, we assume that each one of them has a port limit equal to L_{OCS}. Finally, we define as $\mathcal{W}$ as the set of wavelength channels per optical port at a ToR switch.

The optimization problem at hand consists on maximizing the number of VDCs to be mapped over a DC infrastructure with constrained capacity given a known set of VDC requests. For this, we define $\mathcal{D}$ as the request set. Each VDC is characterized by a virtual graph $\mathcal{G}_d = (\mathcal{N}_d, \mathcal{E}_d)$, with $\mathcal{N}_d$ the set of virtual nodes and $\mathcal{E}_d$ the set of virtual links. Each virtual node requests for a capacity in terms of VMs equal to VM_n^d while each virtual links requests for a bit-rate equal to B_e^d. In this regard, we follow a joint mapping approach to the VDC mapping, coordinating both node and link mapping phases. The benefits of a joint mapping approach have been demonstrated in the literature (e.g. [11], [12]), increasing the acceptance of the VDCs. In this regard, it is the role of the orchestrator not to only provide the joint mapping of the VDCs but to also coordinate all the involved software services to achieve the desired deployment. Next, we proceed on detailing the algorithm utilized for the mapping computation.

A. Optimization Algorithm for VDC Provisioning

The proposed algorithm takes as input the DC infrastructure, considering both server and network resources as well as the whole demand set and outputs the number of successfully deployed VDC instances as well as their corresponding mapping. Essentially, the algorithm applies an adaptive iterative procedure, trying to fit a VDC request per iteration. At the end, the obtained solution is returned. Let us note that we impose the restriction that the virtual nodes of a VDC request have to be mapped over different racks. This is done in order to achieve

some degree of robustness against server and rack failures. Nevertheless, the VMs of the same virtual node are mapped over servers belonging to the same rack. Additionally, we consider the possibility to aggregate several virtual links that have the same source and destination ToRs onto the same optical connection (lightpath) thanks to the electronic capabilities of the ToR switches.

With these, the steps of the algorithm are as follows:

1. Calculate all the candidate lightpaths from every pair of source and destination ToRs in the DCN, employing a K-shortest path strategy for the route computation.
2. Sort the demands in the demand set in descending order according to the following metric:

$$\frac{\sum_{n_d \in N_d} VM_n^d}{|N_d|} + \frac{\sum_{e_d \in E_d} B_n^d}{|E_d|}$$

   In this way, the algorithm tries to serve first the demands that request more resources respect their number of nodes and links.
3. Once sorted, it proceeds with the mapping of the VDCs in an iterative fashion. The procedure ends once every VDC has been tested for a successful mapping.
4. Start with the virtual node mapping. For this, the algorithm sorts the racks in the DC in descending order according to the Cartesian product between the normalized difference of the requested VMs and the server with the highest available capacity in the rack, and the normalized difference with the aggregated bit-rate incoming/outgoing from the virtual node and the available NIC capacity of the same server. If one of the two differences is equal to 0, it is set to 1 for the purpose of the metric. If both of them are negative, the resulting product will be set to negative. In this regard, the algorithm introduces network availability status awareness in the node mapping.
5. Select the first rack and map the VMs onto the servers having the highest previously described metric inside the rack. Update the capacities of the servers. Proceed with the next virtual node.
6. For the link mapping, the algorithm first checks if there is a currently established lightpath from the same source to the same destination ToR. If not, a new lightpath is created, selecting the first candidate lightpath that has enough room to support the virtual link. If yes, the lightpaths sharing the same source and destination are sorted in descending order according to their remaining capacity. The first lightpath with enough room to fit the virtual link is selected.
7. Update the capacity of the servers' NICs, network links and established lightpaths. Proceed with the next virtual link.
8. If the VDC has been fully served, its mapping is added to the full solution. Contrarily, the mapping is disregarded and the status of the DC resources is restored prior to the VDC mapping.
9. Proceed with the mapping of the next VDC starting from Step 4.

Note that the presented algorithm can be also applied for the dynamic provisioning of VDC instances, where requests arrive at the DC infrastructure one by one and no knowledge of the whole demand set can be achieved. For this, the algorithm is applied to a single demand, considering its characteristics and the actual status of the DC infrastructure. By coordinating both the node and the link mapping of the VDC requests, it is possible to minimize the blocking probability (BP) given a dynamic arrival and departure process.

IV. Results and Discussion

In this section we will evaluate the performance of the proposed algorithm and the benefits of a joint mapping of the VDC resources both from static and dynamic provisioning perspectives. For benchmark purposes, we also implemented a variation of the mapping algorithm where the node mapping is performed solely considering the status of the IT resources, that is, the server capacity, balancing the load of the servers and the aggregated capacity of the racks. Before discussing the obtained results, we detail the considered scenario for the tests.

A. Considered Scenario

We consider a DC scenario composed of 6 racks with each rack consisting of 48 servers. Each server has a VM capacity equal to 10 and a NIC capacity equal to 1 Gb/s. As for the network, we consider a spine-leaf topology, where each ToR is connected simultaneously to several OCS switches. Particularly, we consider that each ToR has a radix of 64x64, with 48 electronic ports at 1 Gb/s dedicated to the connection of the servers within the rack, and 6 optical ports with 12 wavelength channels each at 10 Gb/s dedicated to the interconnection with the OCS switches, accounting for a total capacity of 72 wavelength channels for the inter-rack communication. As for the OCS switches, we consider 2 OCS switches with a radix of 384x384, with each of the ToRs connected through 3 of their optical ports to each OCS switch, accounting for a total of 36 wavelength channels for the interconnection between a particular pair of ToR-OCS switch.

Focusing on the demand set, we generate the VDC requests following a 2-step procedure. First, between 2 and 5 virtual nodes are randomly generated per demand. For simplicity, we assume that each virtual node requests for a single VM, that is, $VM_n^d = 1$. Next, the virtual nodes are randomly connected, preventing the generation of non-connected graphs. Each virtual link requests for a bit-rate in the range [100, 1000] Mb/s, in steps of 100 Mb/s. In order to consider a realistic scenario, we generate the bit-rate of the virtual links considering that the traffic generated by a VM cannot not surpass the NIC capacity of a server (1 Gb/s). With such scenario and parameters, the next section presents the results obtained during the evaluation of the presented algorithm.

B. Results

First, we will evaluate the benefits of a joint resource orchestration when facing the allocation of a known set of VDC requests (static scenario). To this end, we evaluated the number of successfully allocated VDC requests as a function of the

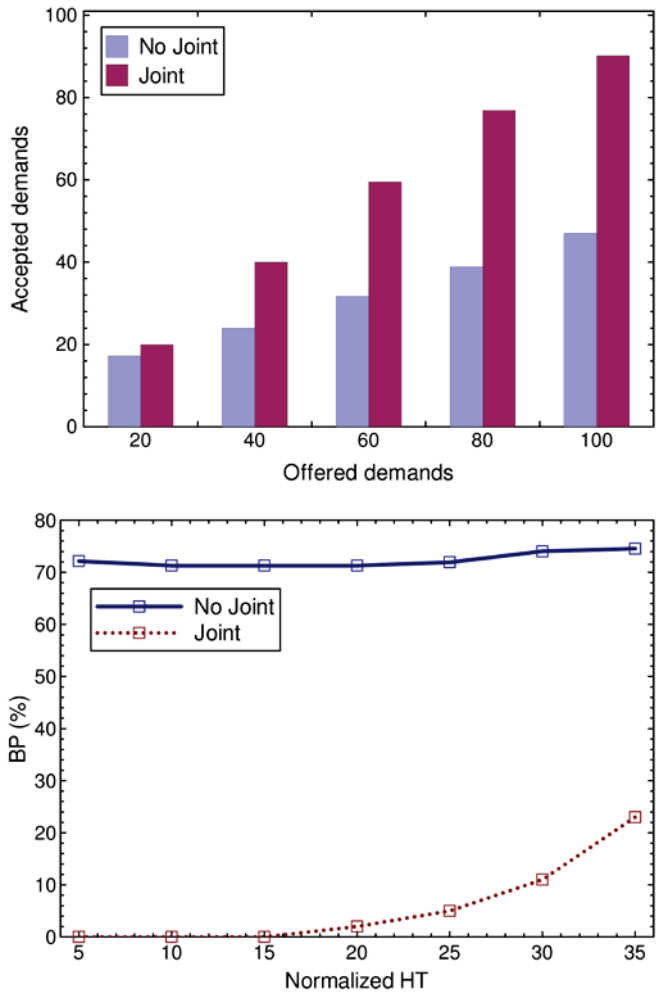

Fig. 3. Comparison between no joint and joint VDC mapping strategies for both static (top) and dynamic (bottom) scenarios.

number of offered VDCs. Fig. 3 (top) depicts the obtained results. All the data points have been obtained averaging 100 random problem instances. It can be appreciated how a joint mapping of the VDC instances allows for a higher number of accepted requests, up to around 47% more when compared with a mapping strategy that does not coordinate both mapping phases. This is due to the network awareness of the node mapping phase in the joint approach, which lowers the chances of blocking due to the lack of network resources during the link mapping phase.

To further evaluate the benefits of an orchestrated approach to VDC mapping, we also performed some tests considering a dynamic scenario, where VDC arrive at the DC following a random arrival and departure process. For this, we consider a Poisson arrival process, with exponentially distributed inter-arrival times (IATs) and holding times (HTs). We evaluated the blocking probability of the VDC requests considering increasing loads. For this, we have fixed the average IAT to one time unit and increased the average value of HT. Fig. 3 (bottom) depicts the obtained results. All the data points have been extracted considering $2 \cdot 10^5$ random VDC arrivals. The obtained results further confirm the benefits of a joint mapping, achieving blocking figures with at least 50% reductions when compared to the non-joint approach.

V. Conclusions

With the emergence of optical technologies inside DC infrastructures and the increasing interest of enabling them with the capacity to provision virtual slices, i.e. VDCs, for exploitation by external tenants, resource orchestration and virtualization has raised as a primordial feature that modern DCs have to implement. In this paper we presented a DC architecture that encompasses both an orchestrator platform, based on OpenStack, and a control layer, based on SDN, for efficient VDC provisioning and deployment.

To assess the benefits of resource orchestration in DC resource management, we also presented an algorithm for optimizing the resource mapping of VDC instances. The obtained results confirm that a joint orchestration is beneficial when deploying complex cloud services, such as VDCs, resulting in around a 50% increase on the accepted requests when compared to no orchestrated resource mapping strategies.

Acknowledgment

This work has been partially funded by the Spanish National project SUNSET (TEC2014-59583-C2-1-R) with FEDER contribution.

References

[1] Cisco Global Cloud Index: Forecast and Methodology, 2014, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html

[2] C. Kachris, I. Tomkos, "A Survey on Optical Interconnects for Data Centers", IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1021-1036, 2012.

[3] K.-K. Nguyen, M. Cheriet and M. Lemay, "Enabling infrastructure as a service (IaaS) on IP networks: from distributed to virtualized control plane", IEEE Communications Magazine, vol. 51, no. 1, pp. 136-144, January 2013.

[4] G. Kandiraju, H. Franke, M.D. Williams, M. Steinder and S.M. Black, "Software defined infrastructures", IBM Journal of Research and Development, vol. 58, no. 2, pp. 1-13, March 2014.

[5] Interoute VDC service, https://cloudstore.interoute.com/

[6] COSIGN Deliverable D4.2, "COSIGN orchestrator low level architecture and prototype design", http://www.fp7-cosign.eu/

[7] A. Mayoral et al., "Integrated IT and network orchestration using OpenStack, OpenDaylight and active stateful PCE for intra and inter data center connectivity", European Conference on Optical Communication (ECOC), September 2014.

[8] Infonetics Research, "Orchestration software is the new battleground in the data center", http://www.infonetics.com/pr/2014/Data-Center-Strategies-Enterprise-Survey-Highlights.asp

[9] OpenStack, https://www.openstack.org/

[10] OpenDaylight, https://www.opendaylight.org

[11] S. Peng et al., "Multi-Tenant Software-Defined Hybrid Optical Switched Data Centre", Journal of Lightwave Technology, vol. 33, no. 15, pp. 3224-3233, August 2015.

[12] A. Pagès et al., "Optimal virtual slice composition toward multi-tenancy over hybrid OCS/OPS data center networks", IEEE/OSA Journal of Optical Communications and Networking, vol.7, no.10, pp. 974-986, October 2015.