Multi-Tenant Software-Defined Hybrid Optical Switched Data Centre

Shuping Peng, Bingli Guo, Chris Jackson, Reza Nejabati, Fernando Agraz, Salvatore Spadaro, Giacomo Bernini, Nicola Ciulli, and Dimitra Simeonidou

***Abstract*—We introduce a holistic solution for software-defined optical data centres (DC). Hybrid optical circuit/packet switching technologies are employed in the data plane, while a software-defined networking (SDN) controller based on OpenDaylight with significant extensions is adopted for the data centre network (DCN) control and management. Novel functional modules in the SDN controller together with its northbound (NBI) and southbound interfaces (SBI) are designed and developed. The OpenFlow protocol is extended at the SBI to support communication between the extended OpenDaylight SDN controller and the optical DCN devices. Over the NBIs, dc applications and the Cloud management system directly interact with the optical DCN. A virtual data centre (VDC) application is designed and developed that dynamically creates and provisions multiple coexisting but isolated VDCs. An optical network-aware virtual machine (VM) placement method is proposed and implemented for a single-step deployment of both network and IT (VM) resources to accommodate the VDC requests. The VDC deployment process is extensively simulated and experimentally demonstrated.**

***Index Terms*—Data centre virtualisation, multi-tenancy, optical data centre network (DCN), software-defined networking (SDN), virtual data centre (VDC).**

I. INTRODUCTION

THE role of data centres (DCs) is increasingly vital in the ICT eco-system [1], [2]. A huge amount of data is being maintained and processed in DCs, on the order of TeraBytes or even PetaBytes per DC. DC providers have observed over 70% annual increase in the DC traffic volume, of which about 50%–75% is east-to-west traffic across DC networks (DCNs) [3]. Current DCN infrastructures built with massive electronic switches in tree-like hierarchical topologies [4], [5] have already been stretched by the ever-increasing demand from new and emerging cloud applications. These infrastructures also suffer from inherent disadvantages such as limited bisectional bandwidth, limited scalability, and high power consumption [6]. In order to accommodate the enormous data volume and new traffic patterns with low power consumption, DCs are moving towards new architectures and solutions, exploring the use of optical technologies.

A plethora of solutions have been proposed to bring optical switching technologies into DCs in recent years, e.g. Helios [7] and C-through [8]. However, they are not able to deal with diverse applications using all-optical solutions. A flat optical DCN architecture empowered by Architecture-on-Demand (AoD) node [9] is proposed in [10], where a combination of optical packet switching (OPS) and optical circuit switching (OCS) technologies is adopted to dynamically accommodate the various traffic flows in DCs. The new optical DCN architecture is able to provide optical connectivity services with high throughput and low latency [11], [12].

Such an advanced optical DCN has also imposed great challenges in the DC control and management, requiring a novel framework that is able to manipulate the underlying heterogeneous optical devices and provide programmability, flexibility, and high availability of DCN connectivity services to meet the highly dynamic requirements imposed by new and emerging DC applications. The current DC control and management platforms are mostly based on rigid and slow human-driven resource provisioning procedures, resulting in a costly operation of DC infrastructure. In [13], a software-defined networking (SDN) [13] control platform is developed, which is based on the OpenDaylight (ODL) controller [15] but with some key functional modules (e.g., Forwarding Rules Manager) extended to support the control of optical network devices. OpenFlow (OF) [13] agents are built for the non-OpenFlow enabled devices, while the OpenFlow protocol is extended to enable communication between the SDN controller and DCN devices. This SDN control platform enables on-demand, flexible, and programmable optical DCN connectivity services [13].

Driven by a growing need to support cloud-based services and applications that must be provided to a wide plethora of business customers/users (e.g., 78% of all UK business have at least one cloud-based service [16]), *multi-tenancy* has become a key requirement of the next generation DCs [17]. With multi-tenancy, DC providers can efficiently multiplex customers across their physical infrastructure, while customers no longer need to buy, manage and maintain a physical DC infrastructure at their own premises. This reduces Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) of both customers and DC providers. Virtualisation is the key technical enabler for the multi-tenancy in cloud DCs [18], [19].

Today, thanks to *Server Virtualisation* [20], DCN infrastructure is surrounded by clusters of extensively virtualised servers. Virtual machines (VMs) can be easily instantiated (in minutes) and run on servers that in the past were dedicated to a single

---

Manuscript received February 2, 2015; revised May 20, 2015; accepted May 26, 2015. The work was supported by EU funded FP7 projects LIGHTNESS (no. 318606) and COSIGN (no. 619572).

S. Peng, B. Guo, C. Jackson, R. Nejabati, and D. Simeonidou are with the High Performance Networks Group, University of Bristol, Bristol BS8 1TH, U.K. (e-mail: shuping.peng@bristol.ac.uk; allen.guo@bristol.ac.uk; cscrj@bristol.ac.uk; reza.nejabati@bristol.ac.uk; Dimitra.Simeonidou@bristol.ac.uk).

F. Agraz and S. Spadaro are with the Universitat Politècnica de Catalunya, 08034 Barcelona, Spain (e-mail: agraz@tsc.upc.edu; spadaro@tsc.upc.edu).

G. Bernini and N. Ciulli are with the Nextworks, 56122 Pisa, Italy (e-mail: g.bernini@nextworks.it; n.ciulli@nextworks.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2015.2438398

application/tenant, therefore building a multi-tenant cloud environment. However, the VMs must wait hours or days for dedicated network connectivity services to be established between them due to the multiple levels of static and human-driven configurations that must be operated. In this context, the physical DCN that interconnects storage and compute resources (i.e., IT resources) must be operated in coordination with the applications running on the IT resources in order to deliver cloud-based applications more efficiently in the multi-tenant cloud environment.

In this paper, dynamic virtual data centre (VDC) deployment is investigated to enable the multi-tenancy in hybrid optical switched DCs. The proposed method, *Network-aware VM Placement*, takes into account both the optical DCN characteristics (e.g., wavelength continuity constraints) and VM dynamics. The method is seamlessly plugged into the OpenDaylight SDN control platform [15] as an advanced application, which also interacts with the OpenStack cloud management system [29] for coordinated and dynamic deployment of VDC requests. The virtual DCN Manager module is developed in the SDN controller to handle the deployment of the virtual DCN requests through interaction with other functional modules.

The rest of this paper is organized as follows. The software-defined optical DC architecture is described in Section II, including the hybrid optical switched DCN and the SDN solution, including the specification of the southbound and northbound interfaces (SBI/NBI). In Section III, the design and implementation of the dynamic VDC deployment using the network-aware VM placement method is introduced. The simulation studies and experimental demonstrations are discussed in Section IV and Section V, respectively. Finally, Section VI concludes the paper.

II. SOFTWARE-DEFINED OPTICAL DATA CENTRE

The proposed software-defined optical DC architecture, as shown in Fig. 1, aims at addressing the requirements resulting from new and emerging DC applications. These requirements are mainly high bandwidth, low latency, flexibility, scalability, and programmability.

The overall architecture includes an advanced optical data plane based on the AoD node as well as a comprehensive SDN control plane with northbound Application Programming Interfaces (APIs) and SBI (e.g., extended OF protocol), and OF agents for heterogeneous network devices, which will be described in details in the following sections, respectively.

A. Hybrid Optical Switched Data Centre Network

In DCs, applications that generate long-lived (elephant) data flows coexist with those that exchange short-lived (mice) flows with tight latency requirements [22]. To cope with the different type of data flows and satisfy their QoS requirements, the proposed optical DCN relies on a flat architecture that integrates OPS and OCS technologies. The adoption of OPS and OCS technologies enables us to avoid expensive Optical-Electrical-Optical conversions and optical transceivers, reducing the energy consumption and cost compared to contemporary electrical solutions. A modular OPS design [10], [31] with highly distributed control for port-count independent reconfiguration time is employed. This enables high load operation with *sub-μs* latency while having the potential to scale to over a thousand ports [31] if significant progress on cost-effective and robust OPS technology platform is made. Racks/servers are interconnected to the hybrid OPS/OCS DCN via Top-of-the-Rack (ToR) switches [10] that perform traffic aggregation and application-aware classification to either short- or long-lived data flows [23], [24]. The OCS technology can better accommodate long-lived high-capacity smooth data flows, while OPS can offer more flexible bandwidth capacity for each optical link when facing dynamic and unpredictable traffic demands with either short- or long-lived flows [25].

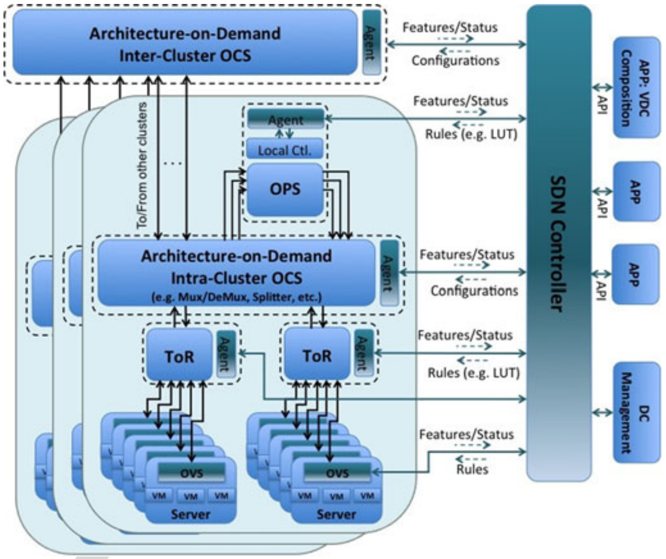


Fig. 1. Overall architecture of the software-defined optical DC (OVS: Open vSwitch [21], APP: Application).

Instead of hardwired interconnections of the main network elements (i.e., OCS, OPS, and ToR switches), the designed optical DCN adopts AoD nodes for constructing both intra- and inter-Cluster connectivity, as shown in Fig. 1. The AoD node consists of an optical backplane, i.e., an optical high-radix fibre switch (e.g., Polatis Series 6000, 192 × 192 ports [26]) with passive/active optical network elements (e.g., Mux/DeMux, Splitter, and Coupler) and switching modules (i.e., OPS and ToR switches) connected to it. With this AoD-based DCN architecture, various arrangements of inputs, elements/modules, and outputs can be adaptively constructed by dynamically provisioning appropriate cross-connections in the optical backplane to meet the requirements of different applications. The combination of hybrid optical switching technologies makes it possible to switch traffic in space, frequency, and time. This enhances the capability to efficiently handle both short- and long-lived traffic flows.

B. SDN Solution for the Hybrid Optical DC Network

For effective control and management of the underlying heterogeneous DCN devices, maintaining direct interaction with

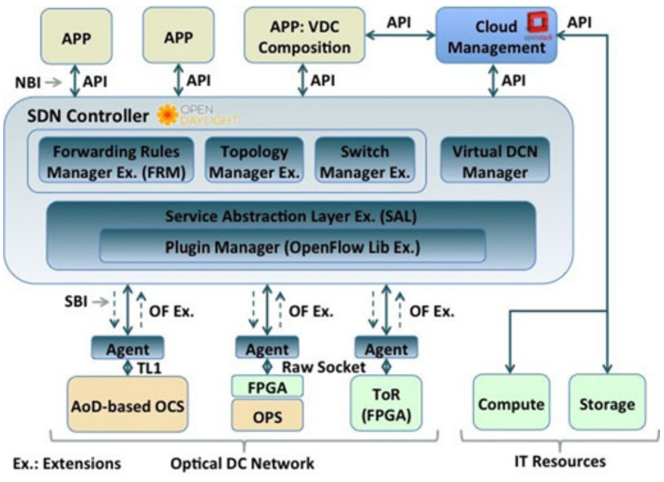


Fig. 2. SDN Solution for the Hybrid Optical DC Network.

DC applications, an SDN-based control plane equipped with both southbound and northbound interfaces is developed for the hybrid optical DCN, as shown in Fig. 2.

1) SDN Controller Architecture and Its Key Modules: The control intelligence is logically centralized in an SDN controller that maintains a full and detailed view of the DCN at both intra- and inter-cluster levels. The SDN controller functional architecture is based on ODL. In order to support the DCN devices, we have extended several key modules in the ODL SDN controller, including the Topology Manager, Switch Manager, Forwarding Rules Manager, Service Abstraction Layer, and Plugin Manager with an extended OpenFlow Library. The Virtual DCN Manager, responsible for the optical DCN virtualization, has also been developed, and is described in further detail in the next section.

The peer information of optical switch ports is parsed in the extended Topology Manager, which is used by ODL to build the optical DCN topology. The power level at the optical ports is collected by the ODL controller from optical switches via the OF agents, by exchanging the extended OF message *of_port_statistics*. Flow entry has been extended to specify optical flows [27]. The Forwarding Rules Manager is extended to construct optical flows and push them to the corresponding SBI. In the Service Abstraction Layer, new data structures have been added to represent the attributes of optical devices. With the OpenFlow Library, extended OF messages are exchanged between the controller and DCN devices. These extensions will be described in detail in the following subsection. The Plugin Manager is not limited to OF and can support multiple protocols as southbound interfaces.

2) Southbound and Northbound Interfaces: Unified communication between the SDN controller and the optical DCN fabric is enabled by the OF agents, which are built for each of the DCN devices, i.e., AoD backplane, OPS, and ToR switches, as shown in Fig. 2. This is to abstract the attributes of the heterogeneous DCN devices and convert the OF protocol to their own technology-specific protocols, such as TL1 and Raw Ethernet.

The OF protocol has been extended to support the optical devices. Fig. 3 shows the implemented key OF extensions in the fields of *ofp_capability*, *ofp_match*, and *ofp_action*. Specifically, the *ofp_match* is extended for OPS to support the use of wavelength and label of an optical packet, while the *ofp_action* is extended to enable the optical packet label configurations in the FPGAs for controlling OPS fabric [28].

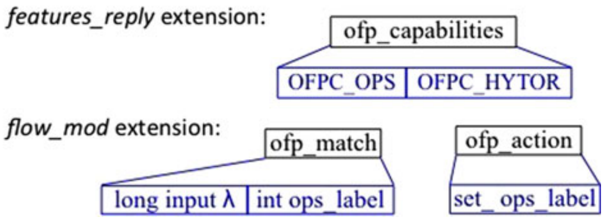


Fig. 3. OpenFlow Protocol Extensions.

Through the northbound interface (i.e., a set of open APIs), the SDN controller interacts with DC applications and the DC management system (e.g., OpenStack [29]). The northbound interface exposes the network functions implemented by the SDN controller to applications, so that it opens opportunities to implement advanced functionalities as DC applications running on top of the SDN controller, such as the VDC Composition that enables multi-tenancy in optical cloud environments. Moreover, using the northbound interface, the DC cloud management system is also able to perform a wide set of monitoring actions [29].

This SDN-based control provides a generalized and modular platform that can be extended to support future network technologies and devices. The adoption and support of open APIs at the northbound and standard interfaces at the southbound (with proper extensions where needed), allows DC operators (or customers) to implement network functions and applications according to their specific requirements and needs.

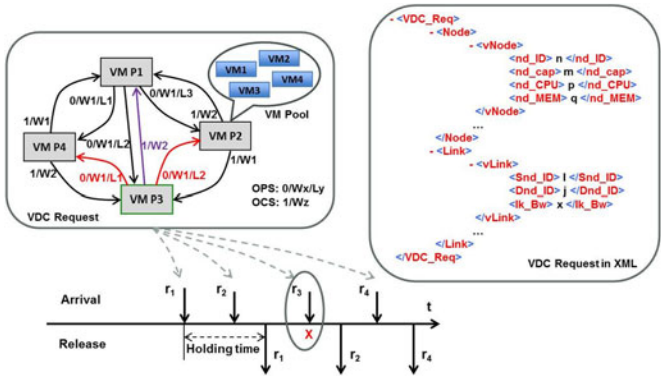
III. DYNAMIC VIRTUAL OPTICAL DATA CENTRE NETWORK DEPLOYMENT WITH NETWORK-AWARE VM PLACEMENT

This study deals with dynamic VDC deployment [30], that is, VDC requests arrive dynamically and are processed on demand for their provisioning. Multiple coexisting but isolated VDCs are deployed taking into account the characteristics and availability of both optical DCN and IT resources (mostly bandwidth and VMs).

A. *Virtual Data Centre Request*

A VDC request (see Fig. 4) comprises a set of virtual nodes interconnected by virtual links. Each virtual node is a VM pool with a number of VMs, each specifying a number of CPU cores and some memory. Each virtual link has constraints on bandwidth and/or latency, etc. The VM pools are interconnected in a certain network topology to accommodate dc application requirements such as distributed computing and file replication.

As shown in Fig. 4, when a DC is up and running, VDC requests arrive dynamically; here, we focus on processing VDC requests sequentially. However, the situation where multiple VDC requests arrive at the same time is not excluded. The whole SDN-based framework still supports and can be extended



```
- <VDC_Req>
      - <Node>
            - <vNode>
                  <nd_ID> n </nd_ID>
                  <nd_cap> m </nd_cap>
                  <nd_CPU> p </nd_CPU>
                  <nd_MEM> q </nd_MEM>
              </vNode>
              ...
        </Node>
      - <Link>
            - <vLink>
                  <Snd_ID> I </Snd_ID>
                  <Dnd_ID> j </Dnd_ID>
                  <lk_Bw> x </lk_Bw>
              </vLink>
              ...
        </Link>
  </VDC_Req>
```

Fig. 4. Dynamic VDC requests.

to handle concurrent VDC requests. Each VDC request has its own holding time (lifetime), and if the request is successfully deployed, when its holding time expires (see the request *r1* in Fig. 4), the corresponding resources (network and VMs) are released for future use. VDC requests may also be rejected (see the request *r3* in Fig. 4), for example due to a lack of available resources or optical layer constraints such as wavelength continuity and/or impairments in optical DCN.

B. Virtual Data Centre Deployment in Optical DCN

To deploy a requested VDC, the pool of VMs per virtual node needs to be embedded into a rack (of servers) that has enough available resources, while the virtual links will be accommodated by optical lightpaths provisioned in the optical DCN. In order to compose multiple coexisting but isolated VDCs to enable the multi-tenancy in optical DC, while taking advantage of the benefits brought by optical technologies, the characteristics of underlying heterogeneous optical devices need to be carefully considered in the virtualisation process.

1) Virtual Optical DCNs with Hybrid Granularities: Leveraging the hybrid optical switching technologies in the optical DC, we are able to achieve hybrid granularities of VDCs: Macro and Micro VDCs, which can be realised by a combination of different technologies.

The *Macro VDC* is a relatively stable VDC slice providing coarse granularity. In the optical DC with hybrid OPS and OCS technologies, the Macro VDCs can be composed by using pre-established optical circuits. This kind of VDC requires less frequent reconfiguration, which perfectly aligns with the relatively long switching time of optical circuit switches (e.g., Max 25 ms of the Polatis 6000).

The *Micro VDC*, providing finer granularities, is more flexible and complementary to the Macro VDC. Multiple Micro VDCs can be deployed on top of one Macro VDC. The statistical multiplexing capability provided by OPS can be exploited in composing the Micro VDCs.

The AoD-based hybrid optical DC design together with its SDN solution facilitates the flexibility of composing the Macro and Micro VDCs and achieving a rich set of granularities of VDCs.

Algorithm I. *VM Placement based on variance calculation*

```
for each VDC request
1     READ capacity requirement ($vC_i$) of virtual nodes ($VMP_i$);
2     READ available capacity ($C_j$) in the racks ($R_j$);
3     COMB = COMBNTNS({j|$R_j$}, VMP_num);
4     COMB_NUM = NCHOOSEK({R_num, VMP_num);
5         for k = 1 : COMB_NUM
6         COMPARE the capacity of VMP and COMB(k, :);
7         if the racks ∈ COMB(k, :) are able to support VMP
8             RECORD COMB_POSS = COMB(k, :);
9             CALCULATE
10                $resC_m = C_m - vC_m$ if $m \in$ COMB(k, :)
11                $resC_m = C_m$ otherwise
12                Mean = SUM{$resC_m$, all m};
13                VAR_VM = SUM{($resC_m$ - Mean)$^2$/R_num, all m};
14    SORT VAR_VM in a descending order;
15    GET the corresponding rack no. R_no of sorted VAR_VM;
```

2) Network-Aware VM placement: The VDC requests need to be properly embedded in the physical DC infrastructure. The actual location of the physical resources to support each VDC request is not the concern of the VDC requester or its users. However, the VDC composition process (including VM placement and virtual link embedding) and the final embedded locations will directly affect the overall DC infrastructure utilisation, VDC operations, and the isolation among VDCs. Energy consumption is another metric, but is outside of the scope of this study. In our work, we develop a coordinative method for the VM placement and virtual link embedding, which is called *network-aware VM placement*. In this subsection, we elaborate on the VM placement part (as shown in *Algorithm I*), while the overall VDC composition process will be explained in detail in the next section.

When a VDC request arrives, the requested VM capacity will be read (Line1 (L1)) as well as the current available capacity in the racks (L2). Here the capacity is defined as the number of VM images (each has an initial amount of CPU and Memory capacities). L3 is used to generate a matrix where each row represents a possible combination of the racks for embedding the requested virtual nodes. Since each virtual node represents a pool of VMs (*VMP*), we assume that one (but only one) virtual node is embedded into one rack. *COMB_NUM* is the number of all the possible combinations of racks (L4). For each possible combination of racks, we compare the available capacity in each rack and the requested capacity (L5–6). Here the bandwidth capacity of the ToR of each relevant rack is also checked. This is where the network-aware VM placement mechanism resides, which will interact with the VDC network embedding process (elaborated in the following subsection). If the requested virtual nodes can be supported (L7), the combination of racks *COMB_POSS* will be recorded (L8), and the variance *VAR_VM* of the residual VM capacity *resCm* of all the racks will be calculated (L9–13).

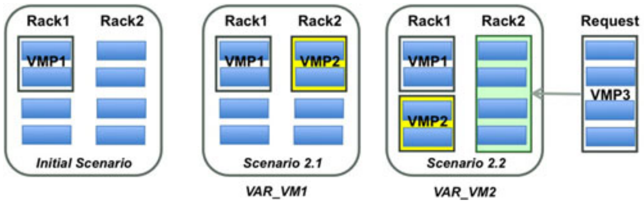


Fig. 5. The illustration of the VM placement with the variance of residual VM capacity calculated and sorted in a descending order.

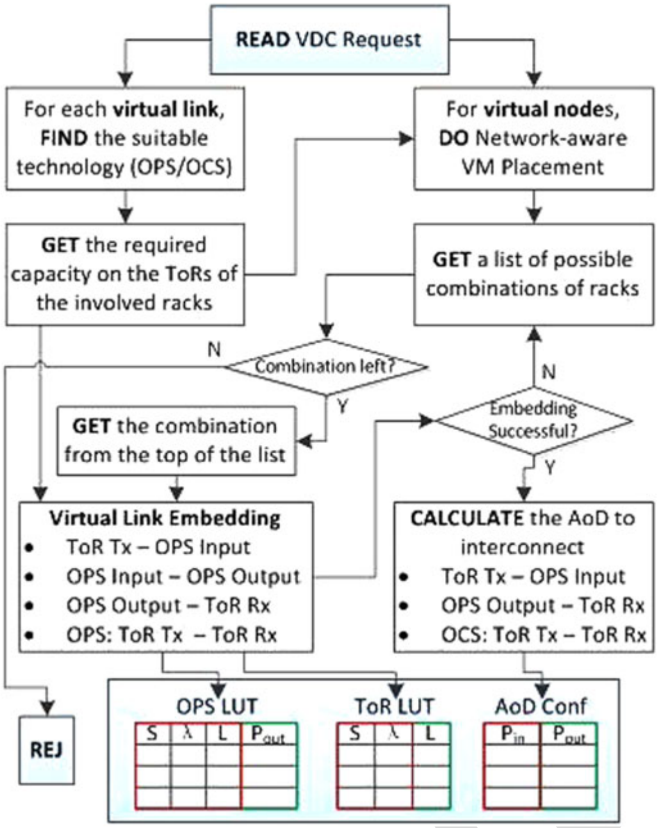


Fig. 6. The overall VDC composition workflow.

All the variances are sorted in descending order (L14), and the list of possible combinations of racks for embedding the virtual nodes is finally obtained (L15). The outputs feed into the overall VDC composition process.

The reason for sorting the variances in a descending order is to increase the VDC request acceptance ratio. To illustrate this, a very simple example is given in Fig. 5.

For the given situation with VMP1 already embedded (Initial Scenario), the two placement options of *VMP2* lead to two different scenarios with different variances (Scenario 2.1 *VAR_VM1* $(= 0) <$ Scenario 2.2 *VAR_VM2*). When the next request arrives (*VMP3*, which requests four VMs), it can only be deployed if the second option of the *VMP2* placement was adopted, that is, the one with higher variance.

3) Virtual Optical DC Composition Workflow: The overall VDC composition workflow is shown in Fig. 6, which integrates the network-aware VM placement as described above.

When a VDC request arrives, the information of its requested virtual nodes and virtual links (see Fig. 4) are read and analysed in parallel. For each virtual link, according to its requested bandwidth, the suitable optical switching technology (OPS or OCS) is selected. By taking advantage of the statistical multiplexing capability of OPS, multiple virtual links can be merged and share a single optical channel. In our work, we sort the bandwidth of the virtual links starting from the same virtual node in ascending order, and then add up the bandwidth of each virtual link till the sum exceeds the capacity of an OPS channel, which is defined as $x\%$ of the wavelength channel bitrate. The $x\%$ depends on the inherent overhead of OPS implementation and the blocking due to congestion. Through the checking of all the virtual links (bi-directional), the suitable optical technology for each virtual link can be determined as well as the requested number of channels starting from each virtual node. Since one but only one virtual node will be embedded in a rack, the requested number of channels is actually the requirement on the corresponding ToR switch of the rack. The *Network-aware VM Placement* module will take this output as an input together with the VM capacity requirements to finally give a list of possible combinations of racks to embed the virtual nodes. We always pick the top element of the list and input it into the *Virtual Link Embedding* module.

When embedding the virtual links through OPS channels across the optical DCN, we need to check against the OPS switching matrix as well as the optical layer constraints such as wavelength continuity if there is no wavelength converter employed. If the embedding is not successful, we will go back to the list and choose the next possible combination if there is still one.

If the deployment is successful, we will be able to obtain the pairs of *ToR Tx - ToR Rx* to be connected via OPS channels. More specifically, they are the sets of pairs of *ToR Tx - OPS Input*, *OPS Input - OPS Output*, and *OPS Output - ToR Rx*. We can determine the source/destination ToRs, the suitable Input/Output fibres (S), wavelengths (λ), and labels (L) for switching optical packets (see Fig. 6). Finally, the connections in the AoD are calculated in order to interconnect the pairs of *ToR Tx - OPS Input*, *OPS Output - ToR Rx*, and *ToR Tx - ToR Rx* via OCS. The final outputs will be the *Look Up Tables* (*LUTs*) for OPS and ToR switches, and configuration files for the AoD, which are pushed by the SDN controller to the devices (described in details in Section V, Experimental Demonstrations).

C. Implementation of the Dynamic VDC Deployment

The dynamic VDC deployment is implemented on a platform with the VDC Composition application interacting with the OpenDaylight SDN controller and the OpenStack cloud management system, as depicted in Fig. 7.

The user (VDC requester) generates a VDC request and calls the DC virtualisation application (APP: VDC Composition). The VDC request is represented in an XML file as shown in Fig. 4.

After a VDC request is received, the request for deploying the virtual DCN is forwarded to the Virtual DCN Manager in

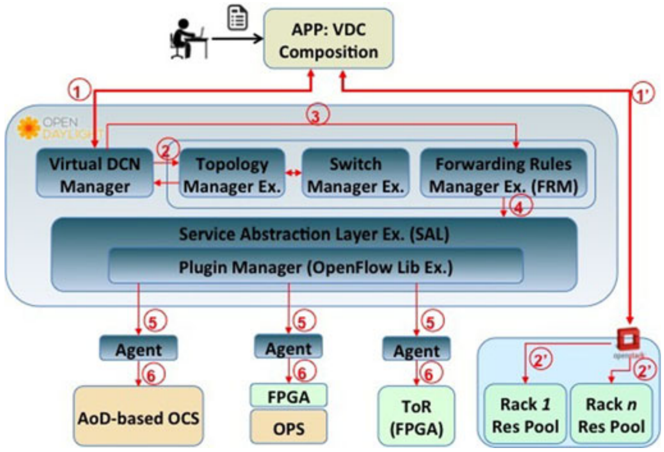


Fig. 7. The VDC deployment via OpenDaylight and OpenStack.

the SDN controller through the NBI (Step 1), while the request for placing VMs is sent to the cloud management system (Step 1′). First of all, the available network and IT resources are retrieved from the Topology Manager in the OpenDaylight controller (Step 2) and OpenStack via Compute API (Step 2'). The Topology Manager exchanges information with the Switch Manager. In Fig. 7, "Ex." indicates those modules in the original ODL controller extended to support optical devices. The retrieved information is presented in a JSON file that specifies the device capabilities and port features.

Based on the retrieved information, the VDC composition engine in the APP is called back to deploy the VDC request using the method presented in Section III-B. The VDC composition engine is implemented in Matlab R2012a.

If the VDC request can be successfully deployed, the configuration rules are specified and pushed to the Forwarding Rules Manager (Step 3) in the SDN controller and the OpenStack Computer Service NOVA. Going through the Service Abstraction Layer and OpenFlow Plugin (OpenFlow Lib) (Step 4), the configuration rules will be pushed to the OF agents of the devices (i.e., AoD backplane, OPS, and ToR switch) via extended OF protocol (see Fig. 3) (Step 5). The devices are then configured by their technology-specified instructions (AoD backplane: TL1, FPGA-based module for OPS and ToR switch: Ethernet frame), which are translated from the extended OF protocol by their own agents (Step 6).

To provision an OCS connection in the proposed hybrid optical DC scenario, the extended OF message *cflow_mod* is used to configure the Backplane cross-connect to interconnect the source and destination (s/d) ToR switches, and extended *flow_mod* is used to configure the flow entries in the s/d ToRs. Then incoming packets that are matched against specific VLANs and/or MACs are forwarded to corresponding output ports.

To provision an OPS connection, a flow entry with the rules for packet matching and label configuration (see Fig. 3) is sent to the source ToR, while the destination ToR will receive and forward the packets to the destination servers. Also, another flow entry for the traversed OPS node will be installed to indicate how the optical packet will be forwarded according to its attached label and assigned wavelength (specified in the extended *ofp_match*). More details can be found in Section V.

IV. SIMULATION STUDIES

In order to evaluate the VDC deployment method we built a simulation platform and conducted extensive experimental studies.

A. *Simulation Scenarios and Parameters*

The simulation platform is built in Matlab R2012a. The network and IT (VM) resources in the optical DC are first simulated as *DC resource database*. The VDC requests are randomly generated according to a set of predefined rules (explained in details below), and processed sequentially. If a VDC request can be successfully accommodated, it will be embedded into the optical DC and the DC resource database will be accordingly updated. When the holding time of a VDC expires, its occupied resources are released for future use.

In the hybrid optical DCN, a cluster includes 8 racks, each rack contains 20/40 servers, and each server has 10 VM images with initial CPU and Memory capacities. Each ToR has 8/16x10Gbps wavelength channels towards the AoD (Backplane). Each channel can support either OPS or OCS transmission. The OPS node supports 10 input and output ports (i.e., 10 × 10), each carrying 8 or 16 wavelengths. Due to the inherent overhead and congestion, the actual channel bitrate for OPS is set to 85%*10 Gb/s [31]. Optical labels are adopted to switch the packets being transmitted in the same wavelength to different output ports (i.e., destined ToRs). The number of optical labels needed will be less than or equal to the maximum node degree of the VDC request.

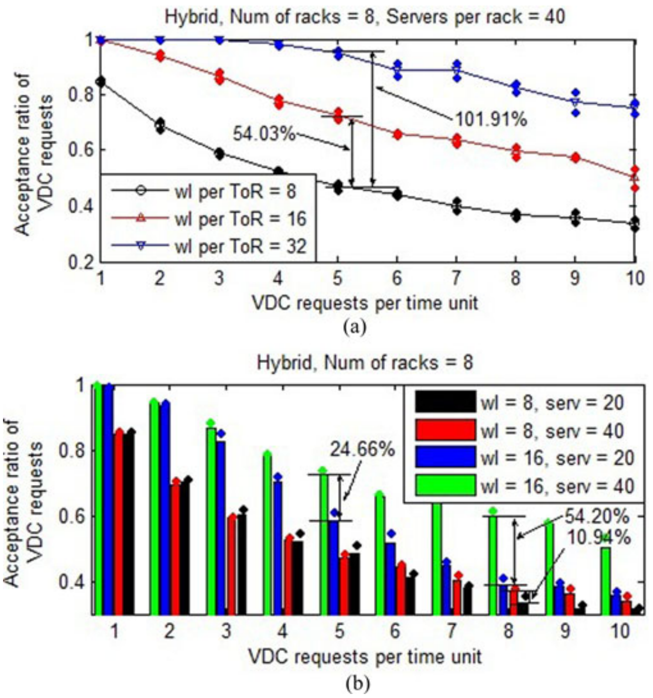
VDC requests arrive dynamically following a Poisson process, while the holding time (lifetime) of each VDC request follows an exponential distribution. The normalized number of VDC requests per time unit is varied from 1 to 10. The VDC topologies (500 in total) are randomly generated with controllable parameters: the number of virtual nodes (Min: 3, Max: 5), node degree (Min: 2, Max: 3), and the probability of interconnecting virtual nodes (0.5). For the capacity requirements, each VDC requests the number of VMs per node (Min: 5, Max: 50) and the bandwidth of each virtual link (Min: 1Gbps, Max: 6 Gb/s).

Based on the simulation platform, we evaluated the impact of available optical DCN and IT resources on the dynamic VDC composition in terms of the acceptance ratio of VDC requests, as well as the benefits of introducing hybrid optical switching technologies into the optical DC and the performance improvements brought by the proposed coordinate/joint mapping of optical DCN and IT resources.

Each result is presented as statistical values with the mean calculated by running the simulation four times. The 95% confidence interval is also shown in the result figures.

B. *The Impact of Available Optical DCN and IT Resources*

The results in Fig. 8 show the impact of available physical resources in the optical DC on the VDC embedding performance



(a)

(b)

Fig. 8. (a) The impact of number of wavelengths per ToR switch (wl per ToR) on the acceptance ratio of VDC requests in hybrid optical DC with 8 racks. (b) The impact of servers per rack (serv.) on the acceptance ratio of VDC requests in hybrid optical DC with 8 racks.

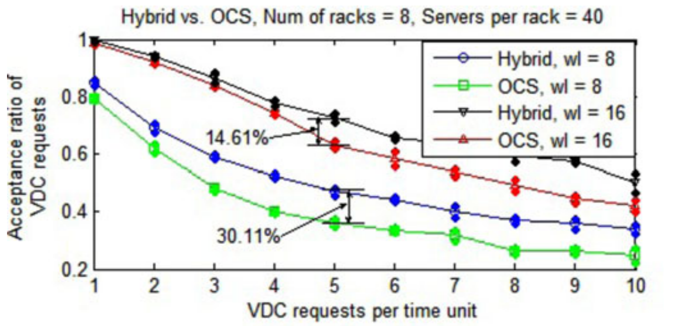


Fig. 9. The comparison of the hybrid optical switched DCN with OPS&OCS and the optical DCN with pure OCS technology.

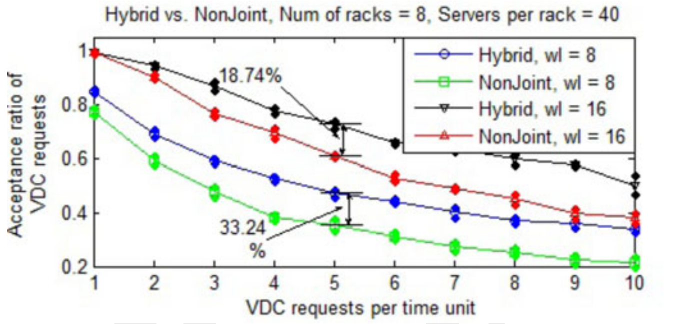


Fig. 10. Effectiveness of the joint optical DCN and IT resources mapping.

in terms of the acceptance ratio of VDC requests. In Fig. 8(a): the number of wavelengths per ToR (*wl per ToR*) [10] and in Fig. 8(b): the number of servers (*serv*) hosted in one rack.

In Fig. 8(a), compared to the scenario with 8 wavelengths per ToR switch, 16 wavelengths can improve the acceptance ratio of the VDC requests by 54.03% when the normalised number of VDC requests per time unit is 5, while 32 wavelengths can achieve 101.91% improvement.

In Fig. 8(b), we can see that when the number of wavelengths per ToR is limited (i.e., 8) the improvement brought by the increased number of servers (IT resources) is also limited. As the limitation on the network resources is released by increasing the number of wavelengths, we can see that growing improvements are introduced by the increasing IT resources. For example, when the normalised number of VDC requests per time unit is 8, the improvement of the acceptance ratio is 54.20% with 16 wavelengths per ToR, and 10.94% with 8 wavelengths per ToR. From Fig. 8(b), we also observe that the improvements brought by the increased IT resources are greater when more VDC requests arrived per time unit, e.g., 24.66% (5 VDC requests per time unit) versus 54.20% (8 VDC requests per time unit).

C. The Benefits of Hybrid Optical Switched DC

The advantages of bringing hybrid optical switching technologies (OPS and OCS) into DCN are reflected in Fig. 9. Compared to the optical DCN with pure OCS technology, with 40 servers per rack, the hybrid optical switched DCN can achieve 30.11% and 14.61% improvements when the number of wavelengths per ToR is 8 and 16, respectively. It shows that the improvement becomes bigger when the limitation arising from the availability of network resources (number of wavelengths per ToR) is stronger.

D. The Benefits of Joint Mapping of Network and IT Resources

In Fig. 10, the effectiveness of the proposed VDC composition with network-aware VM placement ("Hybrid") is evaluated and compared with the one that lacks the joint mapping method ("NonJoint").

Again, we observe that the improvement becomes bigger when the network resource limitation is stronger, that is, 33.24% with eight wavelengths per ToR while 18.74% with 16 wavelengths per ToR.

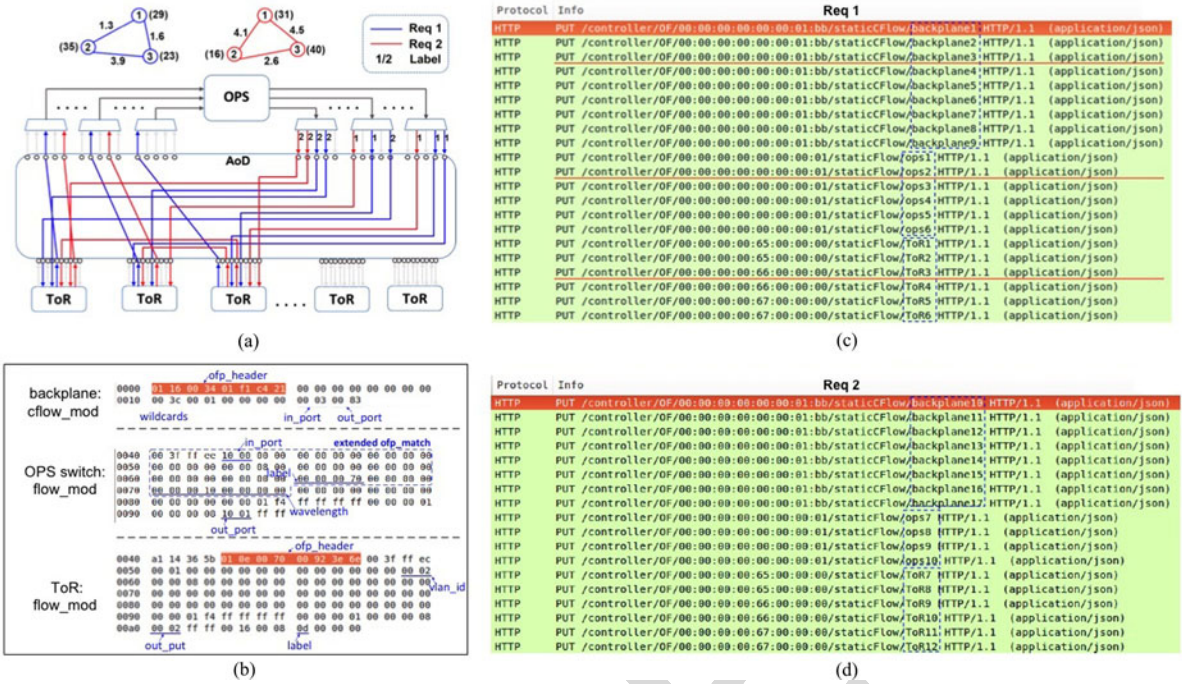
Finally, the execution time of the VDC deployment algorithm is also collected, which is around 35 ms.

V. EXPERIMENTAL DEMONSTRATIONS

We have also established an experimental setup to demonstrate the dynamic VDC composition in the optical DC, and evaluate its performance by measuring the time to successfully deploy a VDC request. The configurations provisioned by the SDN controller to each DCN device using the extended OF protocol (AoD backplane, OPS, and ToR switch) are also shown in this section.

A. VDC Composition in the Experimental Setup

In this experiment, a Polatis switch with 192 × 192 ports is used as the backplane of the AoD node. An OPS switch



(a)

```
0000  01 16 00 34 01 f1 c4 21  00 00 00 00 00 00 00 00
0010  00 3c 00 01 00 00 00 00  00 03 00 83
```

```
0040  00 3f ff ec 10 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 00 00 08 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 70 00 00 00 00
0070  00 00 00 10 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 01 f4  ff ff ff ff 00 00 00 01
0090  00 00 00 00 10 01 ff ff
```

```
0040  a1 14 36 5b 01 0e 00 70  00 92 3e 6e 00 3f ff ec
0050  00 01 00 00 00 00 00 00  00 00 00 00 00 00 00 02
0060  00 00 08 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  00 00 01 f4 ff ff ff ff  00 00 00 01 00 00 00 08
00a0  00 02 ff ff 00 16 00 08  0d 00 00 00
```

(b)

Req 1

```
Protocol  Info
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane1 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane2 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane3 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane4 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane5 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane6 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane7 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane8 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane9 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops1 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops2 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops3 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops4 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops5 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops6 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:65:00:00:00/staticFlow/ToR1 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:65:00:00:00/staticFlow/ToR2 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:66:00:00:00/staticFlow/ToR3 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:66:00:00:00/staticFlow/ToR4 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:67:00:00:00/staticFlow/ToR5 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:67:00:00:00/staticFlow/ToR6 HTTP/1.1  (application/json)
```

(c)

Req 2

```
Protocol  Info
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane10 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane11 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane12 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane13 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane14 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane15 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane16 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:01:bb/staticCFlow/backplane17 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops7 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops8 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops9 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:00:00:00:01/staticFlow/ops10 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:65:00:00:00/staticFlow/ToR7 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:65:00:00:00/staticFlow/ToR8 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:66:00:00:00/staticFlow/ToR9 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:66:00:00:00/staticFlow/ToR10 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:67:00:00:00/staticFlow/ToR11 HTTP/1.1  (application/json)
HTTP      PUT /controller/OF/00:00:00:00:67:00:00:00/staticFlow/ToR12 HTTP/1.1  (application/json)
```

(d)

Fig. 11. Virtual optical DCs deployment and configurations

and ToR switches are plugged into the backplane. Servers are connected to ToR switches via 10GE SFP+. The ToR switches are implemented with FPGA optoelectronics with 12 × 10GE ports, which can be configured by the SDN controller via the corresponding OF agent. The OPS switch is implemented with SOA based fast switch and RF tones labeling technique, as well as an FPGA-based local controller [31]. More implementation details of each individual device can be found in [10].

Two VDC requests are generated randomly and processed one by one. In Fig. 11(a), they are marked blue (Req1) and red (Req 2), respectively. They both have a three-node topology with capacity requirements on their virtual nodes and virtual links, which are specified in the figure. Bidirectional communications are considered while deploying the VDC requests, that is, one virtual link is allocated with two bidirectional physical lightpaths. Before the VDC deployment, the VDC requester is not concerned and is unaware of the physical location of nodes and links allocated to the VDC. The VDC requester will call the VDC composition APP to process the request following the procedure described in Section III. In order to clearly show the VDC deployment, i.e., the end-to-end interconnections between different devices in the wavelength level, the necessary DeMux and Mux are eliminated from the figure. Labels (1/2) are used to indicate the switching of optical packets transmitted over the same input port/wavelength to different output ports.

B. VDC Configurations

After processing a VDC request, the SDN controller will push the devices' configurations via the extended OF protocol. These are then translated into technology specific control messages by the OF agents of these devices. As an example, to map the two VDC requests in Fig. 11(a), the network configuration items are shown in Fig. 11(c) and (d), and one selected configuration item for each device is expanded and the extended OpenFlow messages are shown in Fig. 11(b). The command lines for configurations are listed in the Appendix.

TABLE I
BREAKDOWN OF THE VDC DEPLOYMENT TIME

Total (ms)	Application Execution	Process in OpenDaylight	Process in OpenStack	Device Config.
255.12	35.12 ms	Running in Parallel		25 ms
		195 ms	158 ms	

Backplane: *cflow_mod* indicates the input and output ports of the requested optical cross connection, which is extended from the OpenFlow 1.0 with addendum draft v0.3.

OPS: *flow_mod* is extended by adding optical label (4 bytes) and wavelength (8 bytes) information in the matching fields. The corresponding input and output ports are also indicated.

ToR switch: *flow_mod* indicates the VLAN ID for packet matching and extended with OPS label setting action if the matched packet is going to be delivered via an OPS connection.

In order to allocate resources (i.e., vCPU and Memory) for VMs, the flavor list and OS (Operating System) type in the image list need to be predefined.

C. VDC Deployment Time

The breakdown of the total time for successfully deploying a VDC request is given in Table I, which comprises of 1) VDC application execution (to call the VDC deployment algorithm engine) (35.12 ms), 2) information processing and message exchanges in the OpenDaylight SDN controller (195 ms) and OpenStack cloud management system (158 ms), and 3) device configurations (25 ms). The processes in the SDN controller execute in parallel with those in OpenStack. The total deployment time is 255.12 ms.

VI. CONCLUSION

In this paper, we introduced a flat hybrid optical-switched DCN architecture equipped with an advanced optical data plane and a comprehensive SDN control platform, which is able to provision dynamic and flexible optical connectivity with high throughput and low latency. Using this optical DC, we designed and developed a network-aware VDC deployment method as a DC application for enabling multi-tenancy in the optical cloud environment. The performance evaluation shows the benefits of bringing hybrid optical switching technologies (OPS and OCS) into the DCN, and the effectiveness of the coordinate deployment of virtual DCN resources and VMs. Moreover, we demonstrated deployment of dynamic VDCs, the extended OpenDaylight SDN control platform and the OpenStack cloud management system. The detailed configurations of DCN devices exchanged via extended OF messages and OF agents were described. Finally, the time to successfully deploy a VDC request was measured and analysed to give insight into the implementation process.

APPENDIX

Backplane:

```
curl -u admin:admin -H ‘Content-type: application/json’
-X PUT -d ‘{“installInHw”:”true”, “name”:”backplane”,
“node”:
{“id”:”00:00:00:00:00:00:01:bb”, “type”:”OF”}, “ingressPort”:”x”,
“outgressPort”:”y”, “inWavelength”:”1”, “outWavelength”:
”1”, “wildcards”:”OFPCW_FIBER_PORT”, “actions”:
[“OFPAT_CKT_OUTPUT”]}’ ‘http://localhost:8080/
controller/nb/v2/flowprogrammer/default/node/OF/00:00:00:
00:00:00:01:bb/staticCFlow/backplane’
```

OPS:

```
curl -u admin:admin -H ‘Content-type: application/json’ -X
PUT -d ‘{“installInHw”:”true”, “name”:”ops”, “node”:
{“id”:”00:00:00:00:00:00:00:01”, “type”:”OF”}, “ingressPort”:”x”,”priority”:”500”,”etherType”:”0x800”,
”opsLabel”:”y”, “opsInWavelength”:”z”, “actions”:
[“OUTPUT = n”]}’ ‘http://localhost:8080/controller/nb/v2/
flowprogrammer/default/node/OF/00:00:00:00:00:00:00:01/
staticFlow/ops’
```

ToR switch:

```
curl -u admin:admin -H ‘Content-type: application/json’ -X
PUT -d ‘{“installInHw”:”true”, “name”:”ToR”, “node”:
{“id”:”00:00:00:00:00:00:00:0x”, “type”:”OF”}, “ingressPort”:”y”,”priority”:”500”,”etherType”:”0x800”,”vlanId”:”
z”, “actions”:[“OUTPUT = n”, “SET _LABEL = m”,
“SET_TIMESLOT = p”]}’ ‘http://localhost:8080/controller/
nb/v2/flowprogrammer/default/node/OF/00:00:00:00:65:00:
00:00/staticFlow/ToR
```

VM:

```
curl -i http://nova_controller_ip:8774/v2/tenant_id/servers -
X POST -H “X-Auth-Project-Id: admin” -H “Content-Type:
application/json” -H “Accept: application/json”-H “X-Auth-
Token: token_id”-d’{“server”: {“name”: “instance_name”,
“imageRef”: “image_id”, “key_name”: “repleced_with_key_
name”, “flavorRef”: “flavor_id”, “max_count”: 1, “min_
count”: 1}}’
```

REFERENCES

[1] R. Villars, J. Koppy, and K. Quinn, “From IDC Predictions 2013: The new data center dynamic,” Dec. 2012. Q1

[2] A. Von Lehmen, R. Doverspike, G. Clapp, D. M. Freimuth, J. Gannett, K. Kim, H. Kobrinski, E. Mavrogiorgis, J. Pastor, M. Rauch, K. K. Ramakrishnan, R. Skoog, B. Wilson, and S. L. Woodward, “CORONET: Testbeds, cloud computing, and lessons learned,” presented at the Opt. Fiber Commun. Conf. Exhib., San Francisco, CA, USA, 2014.

[3] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in *Proc. 10th ACM SIGCOMM Conf. Internet Measurement*, Nov. 2010, pp. 267–280.

[4] L. A. Barroso and U. Hölzle, “The data centre as a computer: An introduction to the design of warehouse-scale machines,” in *Synthesis Lectures on Computer Architecture*, San Rafael, CA, USA: Morgan & Claypool, 2009.

[5] F. Abel, C. Minkenberg, I. Iliadis, T. Engbersen, M. Gusat, F. Gramsamer, and R. P. Luijten, “Design issues in next-generation merchant switch fabrics,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1603–1615, 2007.

[6] S. J. B. Yoo, “Energy efficiency in the future internet: The role of optical packet switching and optical-label switching,” *IEEE J. Sel. Topics Quantum Electron.*, vol. 17, no. 2, pp. 406–418,, Mar./Apr. 2011.

[7] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, “Helios: A hybrid electrical/optical switch architecture for modular data centres,” in *Proc. ACM SIGCOMM*, 2010, pp. 339–350.

[8] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, “c-Through: Part-time optics in Data centres,” in *Proc. ACM SIGCOMM*, 2010, pp. 327–338.

[9] M. Garrich, N. Amaya, G. Zervas, P. Giaccone, D. Simeonidou, “Architecture on demand: Synthesis and scalability,” presented at the 16th Int. Conf. Opt. Netw. Design Modeling, Colchester, U.K., 2012.

[10] S. Peng, D. Simeonidou, G. Zervas, R. Nejabati, Y. Yan, Y. Shu, S. Spadaro, J. Perelló, F. Agraz, D. Careglio, H. Dorren, W. Miao, N. Calabretta, G. Bernini, N. Ciulli, J. Carlos Sanchov, S. Iordache, Y. Becerra, M. Farrera, M. Biancani, A. Predieri, R. Proietti, Z. Cao, L. Liu, and S. J. B. Yoo, “A novel SDN enabled hybrid optical packet/circuit switched data centre network: The Lightness approach,” presented at the IEEE Eur. Conf. Netw. Commun., Bologna, Italy, Jun. 2014.

[11] [Online]. Available: http://www.ict-lightness.eu/ Q2

[12] J. Perello, S. Spadaro, S. Ricciardi, D. Careglio, S. Peng, R. Nejabati, G. Zervas, D. Simeonidou, A. Predieri, M. Biancani, H. J. S. Dorren, S. Di Lucente, J. Luo, N. Calabretta, G. Bernini, N. Ciulli, J. C. Sancho, S. Iordache, M. Farreras, Y. Becerra, C. Liou, I. Hussain, Y. Yin, L. Liu, and R. Proietti, “All-optical packet/circuit switching- based data center network for enhanced scalability, latency and throughput,” *IEEE Netw. Mag.*, vol. 27, no. 6, pp. 14–22, Nov./Dec. 2013.

[13] B. Guo, S. Peng, C. Jackson, Y. Yan, Y. Shu, W. Miao, H. Dorren, N. Calabretta, F. Agraz, J. Perelló, S. Spadaro, G. Bernini, R. Monno, N. Ciulli, R. Nejabati, G. Zervas, and D. Simeonidou, “SDN-enabled programmable optical packet/circuit switched intra data centre network,” presented at the Opt. Fiber Commun. Conf., Los Angeles, CA, USA, Mar. 2015, Paper Th4G.5.

[14] [Online]. Available: https://www.opennetworking.org/ Q3

[15] [Online]. Available: http://www.opendaylight.org/

[16] Cloud Industry Forum. [Online]. Available: http://cloudindustryforum.org

[17] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, “NetLord: A scalable multi-tenant network architecture for virtualized datacenters,” in *Proc. ACM SIGCOMM*, 2011, pp. 62–73.

[18] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, “Data center network

virtualization: A Survey," *IEEE Commun. Surveys Tut.*, vol. 15, no. 2, pp. 909–928, 2013.

[19] S. Peng, R. Nejabati, and D. Simeonidou, "The role of optical network virtualization in cloud computing," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 10, pp. A162–A170, Oct. 2013.

[20] J. Daniels, "Server virtualization architecture and implementation," *ACM Crossroads*, vol. 16, no. 1, pp. 8–12, Sep. 2009.

[21] [Online]. Available: http://openvswitch.org/

[22] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proc. ACM SIGCOMM Workshop: Res. Enterprise Netw.*, 2009. Q4

[23] P. Peter, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," *Network Working Group*, RFC 3176, 2001.

[24] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Design Implementation*, vol. 10, 2010, p. 19.

[25] Y. Shu, G. Zervas, Y. Yan, S. Peng, S. Yan, E. Hugues-Salas, and D. Simeonidou, "Programmable optical packet/circuit switched data centre interconnects: Traffic modeling and evaluation," presented at the Eur. Conf. Opt. Commun., Cannes, France, Sep. 2014.

[26] [Online]. Available: http://www.polatis.com/press-releases/polatis-introduces-192-fibre-single-sided-all-optical-switch-for-datacentre-and-telecom-networks.asp

[27] M. Channegowda, R. Nejabati, M. Rashidi Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martínez, R. Muñoz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J. P. Elbers, P. Kostecki, and P. Kaczmarek, "Experimental demonstration of an OpenFlow based software defined optical network employing packet, fixed and flexible DWDM grid technologies on an international multi-domain testbed," *OSA Opt. Exp.*, vol. 21, no. 5, pp. 5487–5498, 2013.

[28] F. Agraz, W. Miao, A. Ferrer, G. Bernini, H. J. S. Dorren, N. Calabretta, N. Ciulli, J. Perelló, S. Peng, G. Zervas, D. Simeonidou, G. Junyent, and S. Spadaro, "Experimental assessment of SDN-based control of OPS switching nodes for intra-data centers interconnect," presented at the Eur. Conf. Opt. Commun., Cannes, France, Sep. 2014.

[29] [Online]. Available: http://www.openstack.org

[30] S. Peng, R. Nejabati, B. Guo, Y. Shu, G. Zervas, S. Spadaro, A. Pages, and D. Simeonidou, "Enabling multi-tenancy in hybrid optical packet/circuit switched data center networks," presented at the Eur. Conf. Opt. Commun., Cannes, France, Sep. 2014.

[31] W. Miao, S. Di Lucente, J. Luo, H. J. S. Dorren, and N. Calabretta, "Low latency and efficient optical flow control for intra data center networks," *Opt. Exp.*, vol. 22, pp. 427–434, 2014.

Shuping Peng received the B.S. degree in physics, and the Ph.D. degree in communications and information system from Peking University, Beijing, China. She is currently working as a Research Fellow in High Performance Networks Group, University of Bristol, Bristol, U.K. She is involved in multiple international, EU, U.K. projects. Her research interests include optical network virtualization, software-defined optical data centers, network modeling, and algorithm design. She is author and coauthor of more than 60 papers, and has served as a TPC member and the session chair of several IEEE/ACM conferences.

Bingli Guo received the B.S. degree in telecommunication engineering from Jilin University, Jilin, China, in 2005, and the Ph.D. degree in electromagnetic field and microwave technology from the Beijing University of Post and Telecommunication, Beijing, China, in 2011. He is currently a Research Associate at the High Performance Network Group, University of Bristol, Bristol, U.K. From 2011 to October 2013, he was a Researcher in Peking University, Beijing, China. His research interests include routing algorithm in converged IP and optical network, survivability strategies in integrated network and cloud computing system, and software defined optical network.

Chris Jackson received the M.Eng. and Ph.D. degrees from the University of York, York, U.K., and Bristol in 2007 and 2013. He is currently a Research Associate with the High Performance Networks Group, University of Bristol. He has worked previously in Business Intelligence and Analytics. He has authored and coauthored papers on energy-efficient on-chip interconnection networks, including routing algorithms and adaptive architectures. His current research interests include optical networks, software-defined networking, abstractions for network virtualization, disaggregated data center architectures and programming language development.

Reza Nejabati is a Lecturer at the University of Bristol, Bristol, U.K. His research interest include the application of high-speed network technologies, design and control of software-defined, service-oriented and programmable network, cross-layer network design, network architecture and technologies for e-science and cloud computing. He is author and coauthor of more than 150 papers and three standardization documents. He is involved in several national/international projects. He has served as a TPC member, the chair and a organizer of several IEEE conferences and workshops.

Fernando Agraz received the M.Sc. degree in computer science in 2005 and the Ph.D. degree in signal theory and communications in 2012 from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. He is currently working as a Research Engineer in the Optical Communications Group of the Signal Theory and Communications Department of the UPC where he is the responsible of several software developments. He has participated in various FP6 and FP7 European research projects such as LIGHTNESS, EULER, STRONGEST, DICONET, etc. His expertise includes software engineering, communication protocols, and systems administration. His research interests focus on all-optical networks routing, control and management, as well as their application to data center networks and virtualization.

Salvatore Spadaro received the M.Sc. degree in 2000 and the Ph.D. degree in telecommunications engineering from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 2005. He also received the Dr.Ing.degree in electrical engineering from Politecnico di Torino, Torino, Italy, in 2000. He is currently an Associate Professor in the Optical Communications Group of the Signal Theory and Communications Department of UPC. He has participated in various FP6 and FP7 European research projects. He is currently the UPC scientific coordinator for the running EU-funded LIGHTNESS and COSIGN projects. He has published more than 150 articles in international journals and conference proceedings. His research interests include the fields of all-optical networks for data center interconnects, with emphasis on network control and management, resilience and network resources virtualization.

Giacomo Bernini received the Italian Laurea degree in telecommunication engineering from the University of Pisa, Pisa, Italy, in 2006. He is currently the R&D Project Manager at Nextworks, and his research interests include SDN, NFV, cloud computing, NSI, ASON/GMPLS control plane, and PCE frameworks. He recently participated to design, development, and demonstration activities in several FP6 and FP7 projects as well as to industrial projects. He is currently active in the FP7 LIGHTNESS, TRILOGY 2, PACE, CONTENT, and H2020 ORCHESTRA projects.

Nicola Ciulli is a Co-founder and the Head of Research and Development activities at Nextworks, an innovative and dynamic SME, based in Tuscany, Italy, with a lot of investment in R&D activities. He holds a degree in telecommunication engineering from the University of Pisa and SSSUP S. Anna (1997). He has coauthored and participated in several (20+) EU FP4-FP7 and H2020 projects in collaboration with a wide range of EU actors, in National and Regional research initiatives, and in industrial projects. He is a frequent external expert in EC evaluations, project reviews, and call observations, with 10+ years of experience (FP6, FP7, and H2020).

Dimitra Simeonidou is a Professor of High Performance Networks Group, University of Bristol, Bristol, U.K. He is a leading Academic in optical networks, future internet research and experimentation (fire), grid and cloud computing, and a Founder of transport software defined networking. She has chaired a number of international conferences and committees across these technical fields. She is the author and co-author of more than 350 publications of which many have received best paper awards, 11 patents, and standards.