Demonstration of application-driven network slicing and orchestration in optical/packet domains: on-demand vDC expansion for Hadoop MapReduce optimization

Bingxin Kong[1], Siqi Liu[1], Jie Yin[1], Shengru Li[1], and Zuqing Zhu[1,*]

[1] *University of Science and Technology of China, Hefei, Anhui, 230027, China*
[*] *zqzhu@ieee.org*

Abstract: Nowadays, it is common for service providers (SPs) to leverage hybrid clouds to improve the quality-of-service (QoS) of their Big Data applications. However, for achieving guaranteed latency and/or bandwidth in its hybrid cloud, an SP might desire to have a virtual datacenter (vDC) network, in which it can manage and manipulate the network connections freely. To address this requirement, we design and implement a network slicing and orchestration (NSO) system that can create and expand vDCs across optical/packet domains on-demand. Considering Hadoop MapReduce (M/R) as the use-case, we describe the proposed architectures of the system's data, control and management planes, and present the operation procedures for creating, expanding, monitoring and managing a vDC for M/R optimization. The proposed NSO system is then realized in a small-scale network testbed that includes four optical/packet domains, and we conduct experiments in it to demonstrate the whole operations of the data, control and management planes. Our experimental results verify that application-driven on-demand vDC expansion across optical/packet domains can be achieved for M/R optimization, and after being provisioned with a vDC, the SP using the NSO system can fully control the vDC network and further optimize the M/R jobs in it with network orchestration.

OCIS codes: (060.4250) Networks; (060.4510) Optical communications.

1. Introduction

Recently, applications related to Big Data analytics have been increasing exponentially and thus imposed new challenges to Internet infrastructure [1]. To adapt to the rapid growth of such applications, service providers (SPs) are relying on the cloud infrastructure based on geographically distributed (geo-distributed) datacenters (DCs) to achieve timely responses to demands [2, 3]. However, due to the high cost of owning and maintaining multi-DC cloud systems, it would not be feasible for each SP to build its own geo-distributed DCs. Therefore, it is common for SPs to leverage hybrid clouds for ensuring high quality-of-service (QoS) and staying profitable simultaneously [4]. Specifically, in addition to its private DC(s), an SP can rent IT and bandwidth resources from a few public DCs to expand its services dynamically. Then, by creating virtual machines (VMs) through server virtualization [5] in public DCs, the SP can easily deploy its services with reduced latency and increased capacity.

Note that, server virtualization is not the whole story of hybrid cloud management. This is because to successfully deploy Big Data applications with stringent requirements on latency and/or throughput, an SP may require explicit control of the DC network that interconnects VMs and servers in public and private DCs, respectively [6]. For example, it is known that the response time of an analytics request is important for gaining data insight [7], while in Hadoop MapReduce (M/R) [8], which is a programming model for Big Data analytics and implemented based on the famous Hadoop framework, the poorly managed communication-heavy phase without considering the network condition can prolong the response time significantly [7]. Hence,

the SP should be able to configure the traffic routing and bandwidth allocation in the DC network to deliver its services timely and wisely. This, however, is only partially realized in practice, since the SP usually does not have the authority to control the hybrid cloud's DC network and can only set up end-to-end connections among VMs and servers [6, 9]. The issue can be addressed by leveraging the ideas of network slicing [10] and [11]. Here, network slicing means to run multiple logical networks as virtually independent business operations on a common physical infrastructure in an efficient and economical way [10], while network orchestration refers to the scheme to manage the ecosystem of computing, storage, and networking elements in a multi-DC cloud system (*i.e.*, including both intra- and inter-DC networks) in concert for ensuring the QoS of applications [11]. More specifically, the DC network should be virtualized such that the SP can get a virtual DC network to manage and manipulate the network connections in it for guaranteed latency and/or bandwidth [12, 13]. Specifically, the SP can request virtual network (VNT) slices and VMs from one or more infrastructure providers (InPs), and stitch them and its private DC(s) together to form a virtual DC (vDC). Here, the topology of each VNT should be customized such that the virtual switches (VS') and virtual links (VLs) in it are assigned with sufficient switching capacity and bandwidth, respectively, and can connect the VMs and servers of the SP like in a real DC. This actually motivates us to study the network slicing and orchestration system (NSO) for creating and managing vDCs.

The rapid growth of data traffic has been stressing the transport capacities of inter-DC networks, which are currently built based on fixed-grid wavelength-division multiplexing (WDM) transmission systems. However, such an inter-DC network only has a rigid optical layer for bandwidth allocation, and thus can hardly adapt to the heterogeneous bandwidth demands among DCs. Recently, by leveraging the flexible WDM grids [14], elastic optical networks (EONs) have been proposed for achieving flexible bandwidth allocation in the optical layer [15, 16]. Hence, with an EON-based inter-DC network, the operator can utilize optical bandwidth resources more efficiently by setting up lightpaths adaptively according to the actual bandwidth demands [17–20]. Moreover, industry has already supplied the key enabling components for EONs, *e.g.*, bandwidth-variable wavelength-selective switches (BV-WSS') and bandwidth-variable transponders (BV-Ts), as commercial products. Therefore, in this work, we will consider an EON as the inter-DC optical domain.

Previous studies have suggested that by leveraging software-defined networking (SDN), the network control and management (NC&M) of vDCs becomes more programmable for their SPs [9, 21]. To combine network virtualization with SDN, the key system is the network virtualization hypervisor (NVH) [22], with which an InP can slice VNTs for SPs. Note that, in order to adapt to the bursty M/R workloads, an SP might need to expand its vDC dynamically across heterogeneous optical/packet domains by leveraging multiple NVHs. This is because a hybrid cloud usually consists of several packet domains as intra-DC clusters and an inter-DC optical domain to interconnect the clusters [23, 24]. The network domains are usually managed by different InPs, each of which would own and operate an NVH. Moreover, the application-driven vDC expansion would not be possible without the support from the IT management, *i.e.*, monitoring IT resource usages on VMs/servers and adjusting their workloads accordingly.

To this end, we can see that to enable the "pay-as-you-use" scenario for the SPs to achieve application-driven on-demand vDC expansion, we need a multi-NVH based NSO system. Here, the on-demand vDC expansion refers to the mechanism that when the IT and/or bandwidth resources in an SP's private DC or vDC become insufficient for its applications, it requests for customized VNTs and VMs from the InPs through an NSO system, which will create the required VNTs and VMs and stitch them with the SP's original DC to form a new vDC. More specifically, for Hadoop M/R optimization, the NSO system should be able to realize dynamic network slicing in optical/packet domains and manage VMs/servers in DCs effectively. The design and implementation of an NSO system with such comprehensive functionality have not

been fully explored before. Furthermore, to verify the system is truly operational, one needs to incorporate experimental demonstrations that can evaluate it with real M/R jobs running. Hence, the performance of the control plane operation for expanding a vDC on-demand, the data plane operation for routing real M/R traffic in the vDC, and the IT management operation for distributing the jobs among the VMs/servers, can all be confirmed. Nevertheless, to the best of our knowledge, such experimental demonstrations have not been achieved before.

In this work, we design and implement an NSO system that can create and expand vDCs across optical/packet domains on-demand for M/R optimization. Specifically, we present both the architectures of the data, control and management planes in the system and the operation procedures for creating, expanding, monitoring and managing a vDC for M/R optimization in details. The proposed NSO system is realized in a small-scale network testbed that includes four optical/packet domains, and we conduct experiments with it for proof-of-concept demonstrations on the whole operations of the data, control and management planes. The experimental results verify that application-driven on-demand vDC expansion across optical/packet domains can be achieved for M/R optimization, and after being provisioned with a vDC, the SP can fully control the vDC network and further optimize the M/R jobs in it with network orchestration.

The rest of the paper is organized as follows. Section 2 provides a brief survey on the related work. We describe the architecture of the proposed NSO system in Section 3, and the operation procedures designed to achieve Hadoop M/R optimization are discussed in Section 4. We present the experimental demonstrations in Section 5. Finally, Section 6 summarizes the paper.

2. Related work

To realize virtual SDNs (vSDNs), NVH abstracts the resources in the InP's substrate network, virtualizes substrate switches for network slicing, and bridges the communications between the substrate switches and the controllers of vSDNs. Therefore, from the view point of each vSDN controller, it directly manages the VS' in its VNT slice, while all the substrate switches view the NVH as the centralized controller for NC&M. Previously, a few NVH systems for packet domains have been demonstrated, such as FlowVisor [25], OpenVirtex [26], and SR-PVX [27]. However, they cannot achieve vSDN slicing in optical domains since the characteristics of the optical layer have not been addressed. By extending the famous OpenFlow protocol to include optical-related extensions, researchers have studied the system architecture and operation procedure for realizing VNT slicing across multiple heterogeneous domains, and they have demonstrated the control plane operations for virtual optical network deployment and virtual infrastructure composition over multi-domain multi-technology transport networks in [28] and [29], respectively. Nevertheless, these demonstrations did not include data plane operations, and they did not try to deploy real applications in the created virtual network/infrastructure either. Recently, with the EON as the background, we have designed and implemented a network system to build and operate virtual software-defined EONs (vSD-EONs) in [30], and demonstrated control and data plane operations for survivable and application-driven vSD-EON slicing in [31] and [32], respectively. However, the network system does not work for heterogeneous domains or have the capability of IT management, and thus it cannot achieve application-driven NSO.

In addition to generic network virtualization, attentions have also been paid on vDC formulation. Specifically, people have designed network systems, such as Oktopus [9] and Diverter [21], to create vDCs in the packet domain within a physical DC. Besides the drawback that they did not address heterogeneous domains, the proposed systems have other limitations, *e.g.*, Oktopus works only for tree-like substrate topologies and Diverter does not provide QoS guarantee on bandwidth or latency. Chen *et al.* [23] demonstrated the control plane operations for building vDCs over multi-domain optical networks. However, data plane operations were not included and the authors only considered the optical domain by abstracting each DC as an edge node. Meanwhile, there are standardization efforts on network slicing and orchestration too. Cross

stratum optimization (CSO) discusses the cooperation between the application stratum and network stratum to efficiently utilize cloud and network resources for providing various QoS levels to applications [33]. However, the architecture of CSO does not include the virtualization layers discussed in this work, and thus an SP might not have the explicit and fine-grained control on its vDC network that interconnects the VMs and/or servers. The architecture of abstraction and control of traffic engineered networks (ACTN) leverages physical network controllers to manage administrative domains, uses a multi-domain service coordinator to abstract underlay transport resources and provisions VNTs to tenants, and allocates each tenant a customer network controller to manage its [34]. Nevertheless, ACTN only focuses on how to realize the virtualization of network resources across multiple domains, but it does not consider the virtualization and management of IT resources. In the meantime, we hope to point out that the vDC creation/expansion discussed in this paper covers a more generic aspect in terms of NC&M than the network function virtualization (NFV) [35]. Specifically, in a use-case of NFV, both the types of virtual network functions (VNFs) and the traffic routing among them are predetermined, while for the vDC creation/expansion, the VNTs are undirected and the VMs can carry various types of VNFs. In other words, with a vDC, the SP can implement many NFV use-cases.

Hadoop M/R is a scalable framework for realizing Big Data analytics with parallel and distributed algorithms in a cluster [8]. The Hadoop cluster gets supports from the Hadoop distributed file system (HDFS) to run on commodity servers, and consists of a master node and a few slave nodes. The master node is in charge of handling job requests, assigning tasks to slave nodes, and monitoring the cluster's performance, while all the tasks are actually executed by the slave nodes. Hence, typical M/R applications such as WordCount (*i.e.*, to count the number of a word's occurrences in a given set) and Teragen (*i.e.*, to generate random data) can run in a Hadoop cluster. Known as the next generation Hadoop, Hadoop yet another resource negotiator (YARN) has been designed to conceal the heterogeneousness among slave nodes. Recently, intensive studies have been conducted to optimize M/R from the perspectives of resource scheduling [36] and job [37]. Nevertheless, how to optimize Hadoop M/R with application-aware network design has just started to attract research interests since recently [7]. In [7], the authors leveraged SDN to design a framework that can conduct adaptive traffic engineering on M/R workloads to optimize their overall completion time. However, the proposed approach neither considered network virtualization nor tried to scale network resources dynamically.

3. System architecture

3.1. Overall network architecture

Figure 1 shows the overall architecture of the proposed NSO system that can create and expand vDCs across optical/packet domains dynamically for M/R optimization. The NSO system consists of three planes, *i.e.*, the data, control and management plane, all of which will be implemented and demonstrated in this work. Then, by checking the network elements in the three planes, we can see that there are five entities plotted with different colors in it, *i.e.*, four autonomous systems (AS') and a third-party resource broker.

For the AS', the SP owns the private DC, two InPs own two public DCs, respectively, and the third InP owns the EON that interconnects the DCs. The DCs are packet domains built with OpenFlow (OF) switches (*i.e.*, commercial hardware switches or software-based OpenvSwitch (OVS) [38]). Each public DC takes the normal three-layer intra-DC network architecture based on fat-tree [6], *i.e.*, the VMs/servers are connected by top-of-rack (ToR) switches, the aggregation switches groom the traffics from ToR switches, and the core switches facilitate backbone communications. Each core switch is equipped with several optical ports (*i.e.*, 10GbE SFP modules), which can be used to set up lightpaths in the EON. Note that, we use transparent optical networking in the EON, which means that each optical port is directly connected to a BV-WSS in the EON without going through an optical line system for optical-electrical-optical (O/E/O)

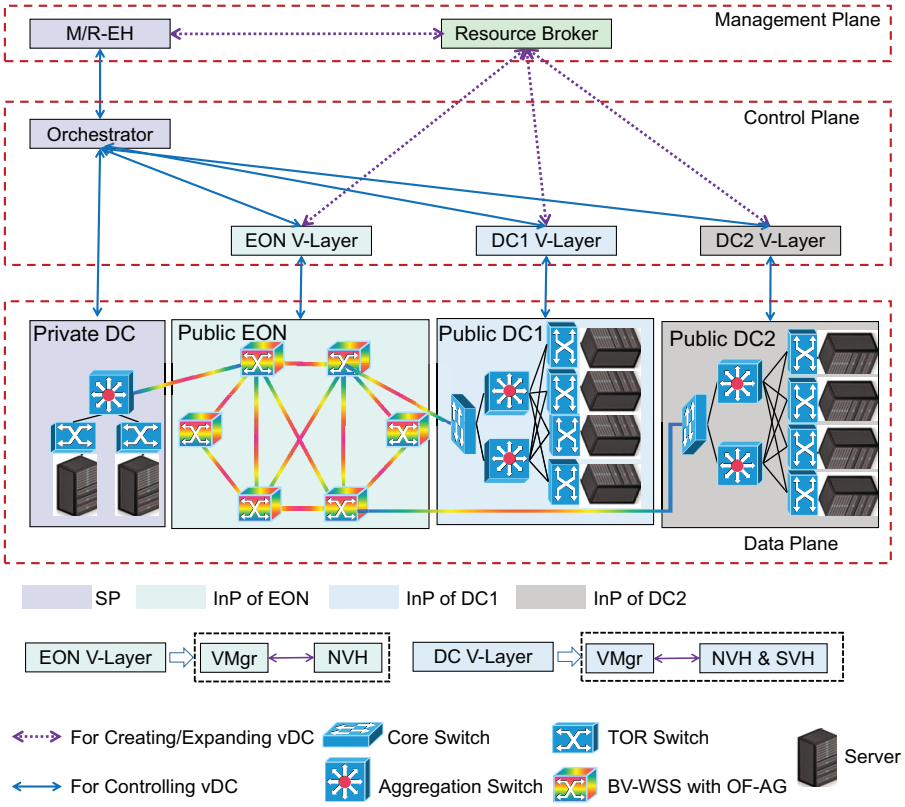

Fig. 1. System architecture for creating and expanding vDCs across heterogeneous optical/packet domains, ToR switch: top-of-rack switch, BV-WSS: bandwidth-variable wavelength-selective switch, OF-AG: OpenFlow agent, V-Layer: virtualization layer, VMgr: virtualization manager, NVH: network virtualization hypervisor, SVH: server virtualization hypervisor, M/R-EH: MapReduce event handler.

conversion and traffic grooming. This is because we only have very limited funding budget and needs the flexibility provided by the transparent scheme to virtualize the EON and gain full control over the end-to-end data transmission between any VM/sever pair. The EON consists of several BV-WSS', each of which is controlled by an OF agent (OF-AG) [39, 40] that is home-made and programmed based on OVS. To coordinate the AS' for application-driven NSO, we design a resource broker (Broker) and place it in the management plane [41, 42]. We assume that the Broker is owned and operated by a third-party other than the SP and InPs, and thus the autonomy of each InP is protected. In other words, the Broker only analyzes the resource request from the SP, disassembles it into sub-requests, and forwards them to proper InPs, while the actual NSO is done by the InPs.

3.2. Interactions of management, control and data planes

Each AS owns and operates a network element in the control plane in Fig. 1. The orchestrator of the SP achieves IT/bandwidth resource orchestration in its vDC for M/R optimization, and thus it not only talks with the data plane of the private DC but also communicates with those of the public DCs and EON through their virtualization layers (V-Layer) when necessary. Moreover, the orchestrator may send an alert message to the M/R event handler (M/R-EH) in the management plane, when it sees a mismatch between the resources in its vDC and the requirement

from the M/R workloads. The V-Layer of each InP realizes NSO in its AS for vDC creation and expansion. Hence, each V-Layer consists of a virtualization manager (VMgr) and a resource hypervisor. Specifically, upon receiving a sub-request from the Broker, the VMgr calculates the NSO scheme in its AS, which is then realized by the resource hypervisor. Here, the NSO scheme in an AS includes at least the virtual network embedding (VNE) scheme [13] (*i.e.*, how to map the VS' and VLs of the VNT in the sub-request onto the AS' substrate network), and if it is for a public DC, the NSO scheme also includes the VM placement (*i.e.*, how to deploy the required VMs in the sub-request on the DC's servers). Since the EON only contains bandwidth resources, the resource hypervisor in EON V-Layer only consists of an NVH to realize VNE schemes. We design the NVH based on OpenVirteX and extend it to support OpenFlow 1.3 with optical transport protocol extensions (OTPE).

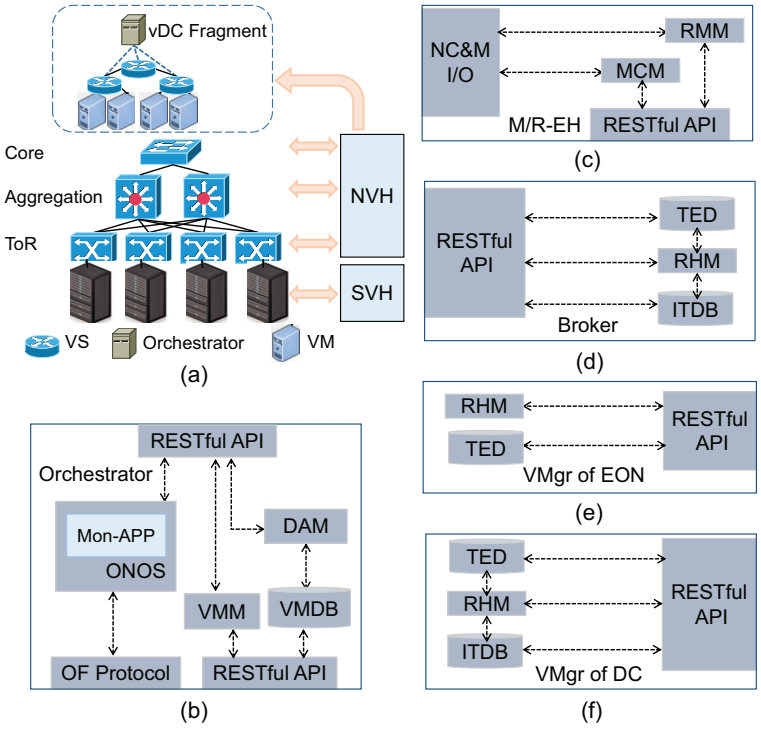

Fig. 2. (a) Creation of a vDC fragment in a public DC, (b) design of the SP's orchestrator, Mon-APP: monitoring application, VMM: VM manager, VMDB: VM monitoring database, DAM: data analytics module, (c) design of the SP's M/R-EH, RMM: resource management module, MCM: monitoring configuration module, (d) design of Broker, TED: traffic engineering database, ITDB: IT resource database, RHM: request handler module, and (e) design of EON VMgr, and (f) design of DC VMgr.

On the other hand, for each public DC, the resource hypervisor in DC V-Layer consists of both an NVH and a server virtualization hypervisor (SVH), to orchestrate the IT/bandwidth resources in the DC for realizing NSO schemes. Figure 2(a) explains how to orchestrate the IT/bandwidth resources in a public DC to create a vDC fragment according to an NSO scheme for vDC expansion. Here, we assume that the resource hypervisor already received the NSO scheme from its VMgr. The VM placement in the NSO scheme is realized by the SVH, which is programmed based on the Nova module in OpenStack [43]. Specifically, the SVH assigns IT resources to the vDC fragment in the form of VMs, which are configured as kernel VMs

(KVMs) by Nova. The VNE in the NSO scheme is accomplished by the NVH, which is also implemented based on OpenVirteX. The NVH can create VS' over ToR, aggregation and core switches and connect them and the VMs from the SVH with arbitrary VLs. After realizing the NSO scheme, the resource hypervisor will connect to the SP's orchestrator to stitch the created vDC fragments with the private DC for vDC expansion.

3.3. Network elements of SP

The design of the orchestrator is shown in Fig. 2(b). In the orchestrator, we implement the S-DN controller of the vDC based on ONOS [44], which manages the physical/virtual switches in the vDC with OF. Note that, to achieve application-driven NSO, we program a monitoring application (Mon-APP) in ONOS to monitor the operation of the vDC's network part. Specifically, Mon-APP sends *Port_Stats_Request* messages to the physical/virtual switches in the vDC periodically to collect their working status. The switches will then reply with *Port_Stats_Reply* messages to report the bandwidth utilization and packet loss rate on each physical/virtual link in the packet domains and the optical power on each VL in the EON. Based on the collected information, Mon-APP determines the working status of each link in the vDC (*i.e.*, in the normal or abnormal state), and will send an alert message to the M/R-EH through RESTful API if necessary. The VM/server part of the vDC is controlled by the VM manager (VMM) in the orchestrator, which is implemented based on the virt-manager [45] to start, migrate and stop the M/R jobs on VMs/servers. Similar to its network part, the operation of the vDC's VM/server part should also be monitored in real-time for application-driven NSO. Therefore, we place an IT resource monitoring agent implemented based on libvirt [46] in each of the VM/server to collect the utilizations of CPU, memory and storage on it. The monitoring agents communicate with the VM monitoring database (VMDB) in the orchestrator, which is realized based on InfluxDB [47] and can store the collected metrics as customizable time-series. Then, the data analytics module (DAM) will analyze the data in the VMDB, determines the working status of each VM/server in the vDC, or sends an alert message to the M/R-EH if necessary.

Figure 2(c) illustrates the design of the M/R-EH, which sits on top of the orchestrator for high-level NC&M. Here, the resource management module (RMM) collects the alert messages from the orchestrator, analyzes their contents, and determines whether an adjustment on the vDC is necessary and how to adjust the vDC based on the automatic management strategies pre-defined by the SP through the NC&M I/O. For instance, if the RMM finds that the utilization of VMs/servers in the vDC is unbalanced, it would invoke VM migrations to re-balance the workloads, and if it determines that the resources in the vDC will become insufficient for M/R workloads, it would send a resource request to the Broker for vDC expansion. The monitoring configuration module (MCM) is designed for configuring the strategies of network and VM/server monitoring in the orchestrator. Specifically, the SP can define the frequency of monitoring and the trigger conditions and parameters of alerts through the MCM.

3.4. Third-party resource broker

Upon receiving a resource request from the M/R-EH, the Broker coordinates the InPs to realize vDC expansion. To achieve this, the Broker is designed as in Fig. 2(d), which includes four software modules. The RESTful-API handles the communications to/from the M/R-EH and the InPs' V-Layers, respectively. For instance, it accepts the resource requests from the M/R-EH, each of which includes the requirements on virtual topology, bandwidth and VMs for vDC expansion. The traffic engineering database (TED) is responsible for gathering global topology of the substrate multi-domain networks. Since each InP might not want to disclose the complete topology of its domain due to security considerations, it may only provide an abstracted topology to the Broker. The Broker then aggregates all the abstracted topologies from the InPs to get the global topology and stores it in the TED. Similarly, the IT resource database (ITDB)

is used to stores the abstracted IT resources in the public DCs. When there comes a resource request from the M/R-EH, the RESTful-API forwards it to the request handler module (RHM), which will analyze the request, communicate with the TED and ITDB to obtain necessary information, and then divide it into sub-requests (*i.e.*, vDC fragments). This can be achieved by modifying the algorithms developed by us in [48] to consider more than two domains. Each sub-request is for a specific optical/packet domain (*i.e.*, an InP), and contains a virtual intra-domain topology and the requirements on IT and bandwidth resources in it. Then, the Broker sends the sub-requests to the V-Layers of the corresponding InPs through the RESTful API.

3.5. Network elements of InPs

In each V-Layer, the VMgr calculates the NSO scheme in its domain for embedding a sub-request. Note that, the designs of the VMgrs for the InPs of a public DC and the EON are different, since the EON only contains bandwidth resources. The design of the VMgr of the EON is shown in Fig. 2(e). The RHM receives the sub-request from the Broker, gets the EON's current status from the TED, calculates a vSD-EON slicing scheme for the virtual intra-domain topology to satisfy its bandwidth requirement, and then forwards the vSD-EON slicing scheme to the NVH in its V-Layer for implementation. Note that, the network slicing in the EON is different from that in the packet-based public DCs, and we realize it by leveraging the approach developed in our previous studies [31, 32]. The details in the VMgr of a public DC is illustrated in Fig. 2(f), and compared with the VMgr in Fig. 2(e), we add two more modules to realize IT management for embedding a sub-request in the public DC. Here, the RHM analyzes the sub-request from the Broker, which contains a virtual intra-DC topology with both IT and bandwidth requirements, and calculates the NSO scheme to embed it based on the network status stored in TED and ITDB. Specifically, the NSO scheme should include not only the network slicing scheme for embedding the virtual intra-DC topology but also the VM placement scheme for instantiating the required VMs on servers. Note that, algorithms should be implemented in the VMgrs to determine the VNE and VM placement schemes in their AS'. Since this work mainly focuses on the proof-of-concept demonstration of application-driven NSO in optical/packet domains and similar algorithms have already been studied by us before [48], we would not emphasize on the algorithm part. Instead, we just implement simple algorithms in the VMgrs to ensure that the experiments will run as designed. This will not limit the effectiveness of our NSO system, since the VMgrs are software modules and can be modified without any restrictions.

4. Operation procedures

With the network system designed in the previous section, we can realize two major functionalities, which are: 1) creating/expanding a vDC across heterogeneous optical/packet domains on-demand and provisioning the vDC to an SP, and 2) monitoring and managing the vDC with IT/bandwidth resource orchestration for Hadoop M/R optimization. The operation procedures to achieve the functionalities are explained as follows.

4.1. vDC creation

The NC&M tasks on a vDC all happen in two phases, *i.e.*, the planning and provisioning phases. Specifically, the tasks in the planning phase are responsible for scaling the IT and network resources in the vDC according to its resource requirement, while those in the provisioning phase are for allocating the resources obtained in the planning phase to actual applications. Hence, vDC creation and expansion both happen in the planning phase. Meanwhile, there are two ways to scale a VM, *i.e.*, the horizontal and vertical scaling schemes [49]. Here, vertical scaling means to add more IT resources in the VM in runtime, which, however, is not supported in OpenStack [49]. Therefore, our NSO system uses the horizontal scaling scheme that instantiates new

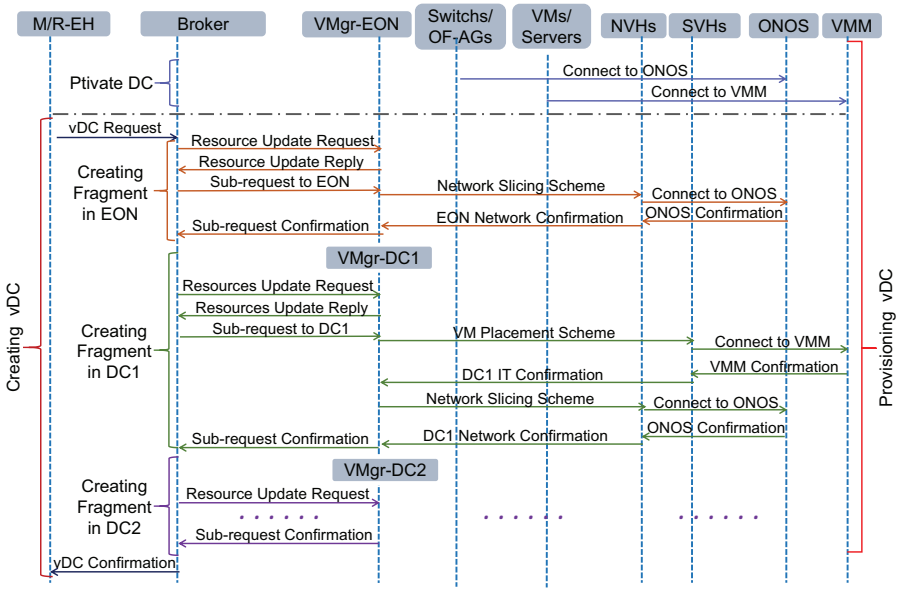

Fig. 3. Operation procedure for creating a vDC and provision it to an SP.

VMs to share the existing VM's workloads, which makes the operations of creating and expanding a vDC in the planning phase essentially the same. To this end, we only discuss how to create a vDC across optical/packet domains and provision it to an SP with the procedure in Fig. 3.

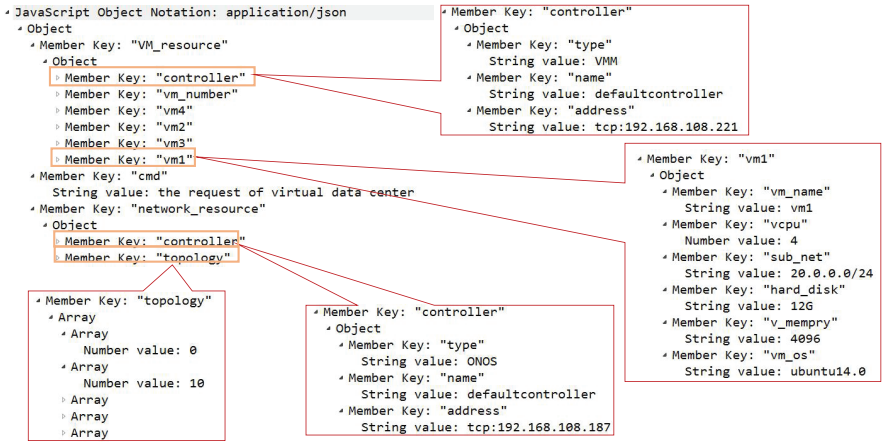

Fig. 4. Wireshark capture of a vDC request.

Initially, for the private DC, the servers and SDN switches in it should be connected to the ONOS and VMM in the SP's orchestrator, respectively. Then, when necessary, the M/R-EH will send a vDC request to the Broker and request for more IT and bandwidth resources. Since the IT resource requirements will be fulfilled with VMs, the vDC request should contain the number of new VMs and the properties of each VM, *e.g.*, its image name and required IT resources in CPU cycles, memory, and disk size. An example on the Wireshark capture of a vDC request is shown in Fig. 4. The "VM_resource" portion indicates the IT resource demand, where the

"controller" field specifies the information of the VMM that manages the VMs in the vDC and the remaining fields are used to customize the required VMs. In the "network_resource" portion, the "controller" field specifies the information of the ONOS (*i.e.*, the SDN controller of the vDC), and the field of "topology" describes the topology and bandwidth requirements of the vDC. Upon receiving the request, the Broker will first communicate with the VMgr of each InP to update its information on the abstracted IT/bandwidth resources in the InP's domain (*i.e.*, the EON or a public DC). Then, based on the latest network status, the Broker divides the vDC request into sub-requests (*i.e.*, vDC fragments), each of which will be sent to an domain for implementation. In the EON, the VMgr first analyzes the sub-request and then calculates a vSD-EON slicing scheme, which will be realized by the NVH. Next, the NVH connects to the ONOS in the SP's orchestrator to hand over the fragment created in the EON to the SP. After all these have been done, the VMgr replies a confirmation to the Broker.

The operations in a public DC in response to the sub-request from the Broker are similar, with the only difference that both network slicing and VM placement (VMP) should be done by the NVH and SVH, respectively, to create the vDC fragment. Then, the NVH and SVH connect to the ONOS and VMM in the SP's orchestrator, respectively, to hand over the vDC fragment. At this point, the vDC is created to include the original private DC and the newly-formed vDC fragments over the EON and public DCs. Note that, in the vDC, all the VMs/servers work just like in an intra-DC network based on Ethernet. Moreover, since all the physical/virtual switches in the vDC can be controlled by the ONOS in its orchestrator, the SP has full control over the vDC network, which means that it can establish arbitrary network connections among the VMs/servers with explicit routing and bandwidth configurations for Hadoop M/R optimization.

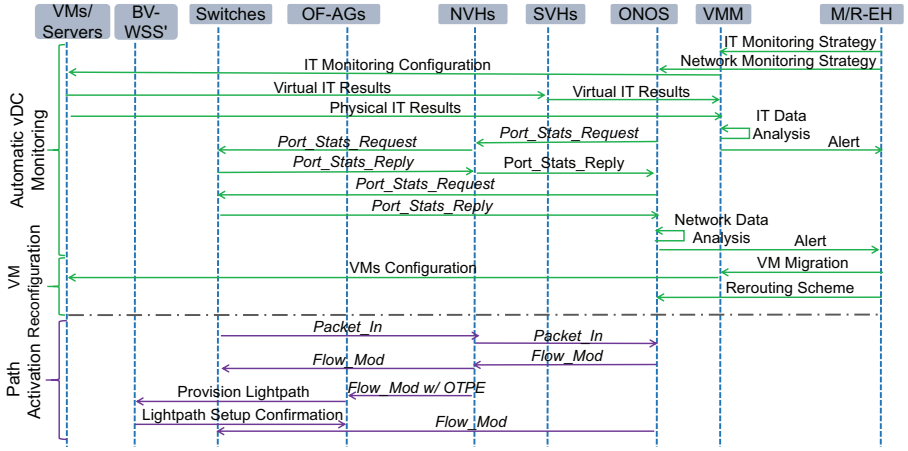

Fig. 5. Operation procedure for monitoring and managing a vDC.

4.2. vDC management

After obtaining the vDC, the SP should be able to monitor its operation in real-time and manage the IT/bandwidth resources in it through network orchestration, for M/R optimization. Figure 5 illustrates the operation procedure for these tasks. Here, we consider a scenario in which the SP's orchestrator detects unbalanced resource usage in the vDC and tries to re-balance the workloads. First of all, the M/R-EH deploys the monitoring strategies for IT and network resources in the VMM and ONOS, respectively. Then, based on the strategies, the VMM and ONOS monitors the resource utilization in the vDC by sending requests to VMs/servers and switches, respectively. Specifically, for the network part, the monitoring is achieved by leveraging

the *Port_Stats_Request* and *Port_Stats_Reply* messages defined in OF, while for the VM/server part, the VMM sends the IT monitoring configuration to the agents on VMs/servers and collects feedbacks from them. The collected monitoring data is then analyzed by the DAM in the orchestrator, and if unbalanced resource usage is detected (*e.g.*, in Fig. 5, we assume that the bandwidth utilization triggers the ONOS sending an alert message to the M/R-EH), network orchestration will be invoked to re-balance the workloads. Specifically, the M/R-EH instructs the VMM and ONOS to conduct VM migration and rerouting, respectively. Here, the "Path Activation" phase in Fig. 5 shows how to set up an end-to-end path with explicit routing configuration to connect a VM in a public DC and a server in the private DC across multiple domains. The VM's edge switch first sends a *Packet_In* message to the ONOS through the NVH of its domain, and the ONOS will then calculate the path and activate it by sending *Flow_Mod* messages to the related switches. For the packet domains, the physical switches in the private DC receives the *Flow_Mod* messages directly from the ONOS, while those in the public DCs get the translated messages through the NVHs of their domains. For the EON, the OF-AGs also receive the *Flow_Mod* messages through the NVH of its domain. Then, they parse the messages to get the routing and spectrum assignment (RSA) scheme of the inter-DC lightpath, and set up the lightpath by configuring their BV-WSS' accordingly.

5. Experimental demonstrations

Our experimental demonstrations include three parts. We first verify that the proposed system can monitor and manage the private DC well, and obtain the performance benchmark of running M/R jobs in the private DC only. Then, an experiment is conducted to confirm that application-driven on-demand vDC expansion across optical/packet domains can be achieved for M/R optimization. Finally, we demonstrate that after being provisioned with the vDC, the SP can fully control the vDC network and further optimize the M/R jobs in it with network orchestration. The experiments are conducted with a small-scale network testbed that includes four optical/packet domains. To emulate a DC, we use Pica8 P-3297 switches with 10GbE optical ports as the core switches, and realize the aggregation and ToR switches with high-performance Linux servers running OVS. Under each ToR switch, there are a few Linux servers, which either work as the physical severs in the private DC or are configured with the Nova-Computer module in OpenStack to host the VMs in the public DCs. The EON uses Finisar 1×9 BV-WSS' that operate within $[1528.43, 1566.88]$ nm and have a bandwidth granularity of 12.5 GHz. On each BV-WSS, we equip a homemade OF-AG that is implemented based on OVS and runs on an embedded Linux board. The power losses in the EON are compensated with Erbium-doped fiber amplifiers (EDFAs). The network elements in control and management planes are all homemade and run on Linux servers. We program them either from scratch (*i.e.*, the M/R-EH and Broker) or by leveraging open-source software (*i.e.*, the orchestrator and V-Layers), as explained in Section 3.

5.1. Monitoring and managing private DC

We first bring up the private DC and deploy Hadoop M/R in it. For proof-of-concept demonstrations, we deploy the Hadoop cluster with one master node and one slave node. Here, the master node manages the Hadoop File System (HDFS) and also works as a YARN to manage the resources in the cluster, while the slave node called Slave1 runs the M/R jobs. We instantiate the master and slave nodes on two VMs, each of which has four CPU cores, 4 GB memory and 12 GB storage, and they are connected with 1 Gbps bandwidth. Then, we run the M/R jobs for WordCount in the Hadoop cluster with different workloads (*i.e.*, with data sizes of 100, 300 and 500 MB), and measure the job completion time of each workload by averaging the results from 5 independent runs. The results are plotted in Fig. 6(a), which shows that the job completion time increases sharply when the workload's data size changes from 100 MB to 300 MB.

The analysis can be verified by the results on CPU and memory utilizations in Figs. 6(d) and

6(e), respectively, which are measured on Slave1. Here, the data size of WordCount is 100 MB. In Fig. 6(d), we observe that the CPU utilization can go above 80% for a relatively long period. If we implement an IT monitoring strategy in the VMM and let it report an alert message to the M/R-EH when seeing CPU utilization going above 80% for 15 seconds, an alert message will be sent out by the VMM at $t \approx 30$ seconds in Fig. 6(d). The Wireshark capture in Fig. 6(b) and the parsed message in Fig. 6(c) confirm that the alert message gets sent from the VMM to the M/R-EH for reporting abnormal CPU utilization. We also implement a strategy to let the VMM report that the state changes back to normal, when it sees CPU utilization staying below 80% for 15 seconds. Hence, the VMM will report the normal state to the M/R-EH at $t \approx 60$ seconds in Fig. 6(d), which is also confirmed by the results in Figs. 6(b) and 6(c).

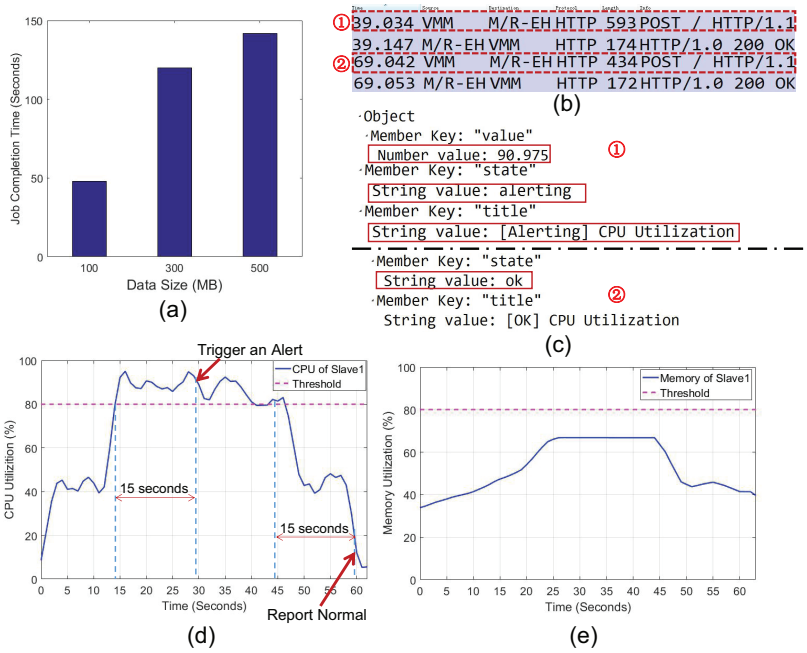

Fig. 6. Experimental results for monitoring and managing the private DC, (a) job completion time, (b) Wireshark capture of communication between VMM and M/R-EH, (c) details of the alert messages for reporting CPU usages, (d) CPU usage on Slave1 with 100 MB workload, and (e) memory usage on Slave1 with 100 MB workload.

5.2. Application-driven on-demand vDC expansion

After receiving the alert message, the M/R-EH determines that the resources in the private DC are insufficient for the M/R jobs and thus it will send a vDC request to the Broker, which will then invoke the on-demand vDC expansion. The Wireshark captures of control messages used for the vDC expansion are shown in Fig. 7. The messages collected on the Broker are in Fig. 7(a), and the corresponding explanations are as follows. In **Step 1**, the Broker receives the vDC request from the M/R-EH, and by analyzing the request, it determines that the vDC expansion will need to 1) deploy four and two VMs in public DC1 and public DC2, respectively, and 2) slice virtual networks across the optical/packet domains to connect the VMs with the servers in the private DC. In **Step 2**, the Broker updates its information on the abstracted IT/bandwidth resources in the domains by sending resource update requests to the VMgrs. Then, the Broker sends the sub-requests for the vDC expansion to the VMgrs in **Step 3**. Finally, after receiving

the confirmations from the VMgr, the broker hands over the newly-formed vDC to the SP in **Step 4**. We can see that our NSO system takes 21.13 seconds to accomplish the vDC expansion.

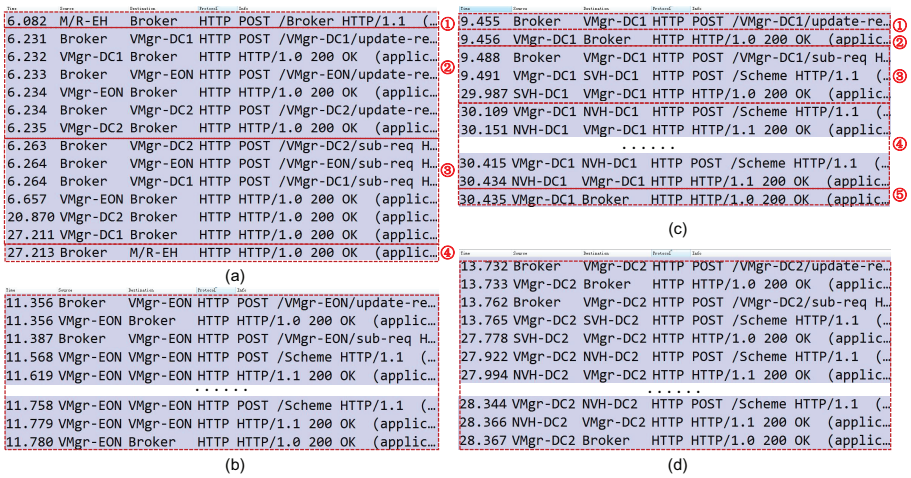

Fig. 7. Wireshark capture of control messages used for vDC expansion, (a) messages collected on Broker, (b) messages collected on EON VMgr, (c) messages collected on DC1 VMgr, and (d) messages collected on DC2 VMgr.

The messages collected on the VMgrs are in Figs. 7(b)-7(d). Since the NSO scheme to realize the vDC fragment in public DC1 is the most complex, we use it as an example to explain the operations of the VMgrs for the purpose of conciseness. In Fig. 7(c), the message exchanges for updating the Broker's information on the abstracted IT/bandwidth resources in public DC1 are in **Step 1**. In **Step 2**, DC1 VMgr receives the sub-request, and it informs the SVH in its domain to create four VMs in **Step 3**. After creating the VMs in suitable servers successfully, the network slicing scheme to satisfy the network requirement is sent to the NVH in its domain in **Step 4**. In **Step 5**, after having the NVH in its domain connected to the SP's orchestrator, DC1 VMgr informs the Broker about the successful creation of the vDC fragment in its domain. The creation of the vDC fragment in public DC1 takes 20.98 seconds in total, which can be broken into 20.53 seconds for VM creation and 0.45 seconds for network slicing. The messages in Fig. 7(b) indicates that the vSD-EON slicing in the EON takes 0.43 second, while those in Fig. 7(d) suggests that DC2 VMgr uses 14.63 seconds to create the vDC fragment in its domain. We can see that the Broker implements the vDC fragment creations in parallel to reduce time latency. To illustrate the vDC expansion, Fig. 8(a) show the physical topology of the network testbed, where we denote each substrate switch as "SN" and number the SNs in each domain independently. The topology of the vDC is illustrated in Fig. 8(b), where we denote each virtual switch as "VN" and number them continuously as in the same domain. Meanwhile, we also list partial mapping results for embedding the vDC in public DC1 and the EON in Table 1.

Table 1. Partial node and link mapping results to embed the vDC

Domain	Virtual Topology	Physical Topology
EON	VN1, VN2, VN3	SN5, SN1, SN6
	VN1-VN2	SN5-SN2-SN1
	VN1-VN3	SN5-SN6
Public DC1	VN4, VN5, VN6	SN1, SN2, SN3
	VN4-VN5	SN1-SN2
	VN4-VN6	SN1-SN3

Note that, the time required for instantiating a new VM depends on the size and location of the VM image, the IT resource requirement of the VM, the workload on the physical server, *etc*. Among these factors, the image size and IT resource requirement of the VM are determined by its SP and cannot be changed, while the remaining two can be optimized. However, even though we do our best to optimize the two factors, the room that can be squeezed from the latency is still very limited. This is because the time used for spawning and booting up the VM is the major part in the latency. Another way to reduce the latency is to create the VM in advance, *i.e.*, the InP can pre-deploy several types of standard VMs on its servers and directly handover the live VMs to the SPs to satisfy their VM requests. By doing so, the InP can reduce the latency of VM creation significantly, but the flexibility of creating customized VMs has been sacrificed.

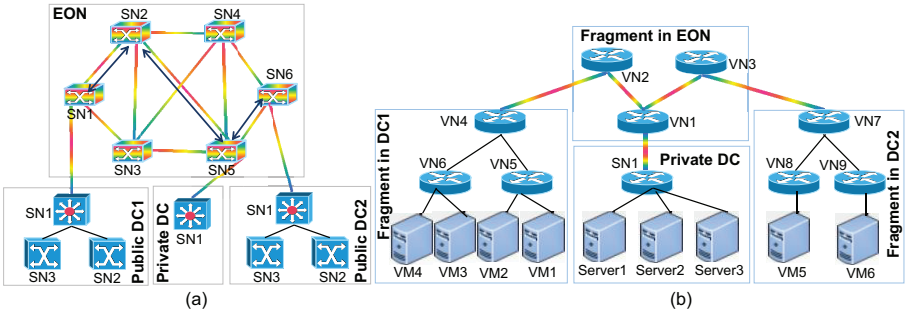

Fig. 8. (a) Physical topology of the network testbed, and (b) virtual topology of vDC.

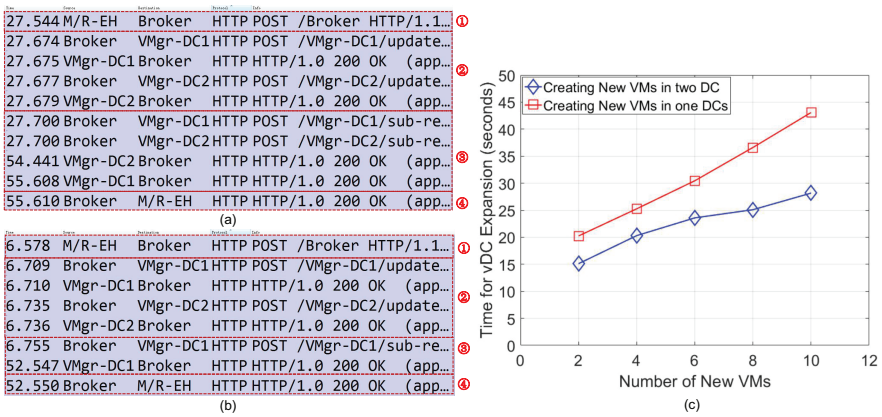

Fig. 9. Experimental results for evaluating NSO system's scalability on vDC expansion, (a) Wireshark capture of control messages used for vDC expansion in two public DCs, (b) Wireshark capture of control messages used for vDC expansion in one public DC, and (c) Time used for vDC expansion.

The experimental results in Fig. 7 indicate that in our small-scale network testbed, the latency of vDC expansion depends largely on the time used for instantiating VMs. Therefore, we conduct more experiments to study the scalability of our NSO system by letting it creating more new VMs in vDC expansion and measuring the total time used for the operations. Here, we consider two experimental scenarios, *i.e.*, all the new VMs are instantiated in one public DC and the new VMs are instantiated in two public DCs evenly, and the experimental setup is the same as discussed above. Figures 9(a) and 9(b) show the Wireshark captures of the control messages

used for vDC expansion in the two scenarios, respectively, for the worst case scenario (*i.e.*, there are 10 new VMs to be instantiated). The M/R-EH sends vDC expansion request to the Broker in **Step 1**, and in **Step 2**, the Broker determines the VM placement scheme that indicates the number of VMs to be instantiated in each DC according to the IT resource requirement. Note that, as these experiments are mainly for measuring the NSO system's scalability, the actual VM placement schemes are hard-coded into the Broker. In **Step 3**, the messages in Fig. 9(a) indicate that the Broker instructs the VMgrs of both DC1 and DC2 to create new VMs simultaneously, while Fig. 9(b) illustrates that all the new VMs are instructed to be instantiated in DC1. Finally, after all the new VMs have been instantiated successfully, the confirmation is sent back to the M/R-EH in **Step 4**. For the two-DC scenario, the vDC expansion totally uses 28.07 seconds, and the VM instantiations in DC1 and DC2 takes 27.91 and 26.74 seconds, respectively, since the operations in the two DCs can be parallel. For the one-DC scenario, the new VMs have to be instantiated in a serial manner, and thus it takes 45.97 seconds in total for the vDC expansion. Figure 9(c) shows the total time used for vDC expansion when the number of new VMs changes from 2 to 10. As expected, the one-DC scenario consumes more time than the two-DC scenario in general. Meanwhile, we notice that the curves in Fig. 9(c) change almost linearly with the number of new VMs, and even though instantiating two new VMs takes around 15 and 20 seconds in the two-DC and one-DC scenarios, respectively, the subsequent new VMs consume much less time to be instantiated. For instance, the average time used for creating a new VM becomes 2.807 and 4.597 seconds, for the the two-DC and one-DC scenarios, respectively, when there are 10 new VMs to instantiate. This is because even in the same DC, our NSO system can parallelize certain operations of VM instantiations when the VMs are created over different servers. The results in Fig. 9(c) verify that our NSO system has reasonably good scalability in terms of number of new VMs involved in the vDC expansion.

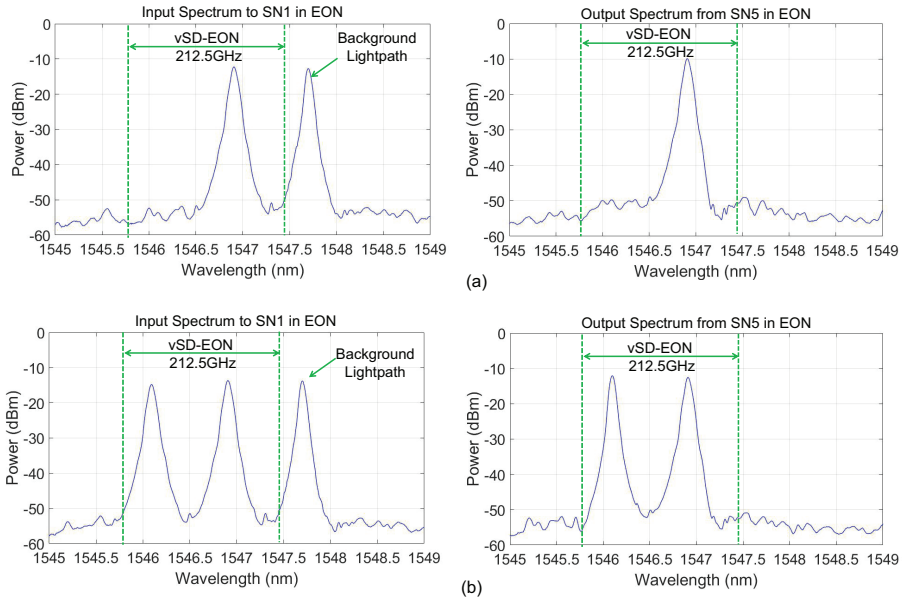

Fig. 10. Experimental results for flexible-grid optical switching in the vDC, (a) input and output spectra of one-lightpath case, and (b) input and output spectra of two-lightpath case.

We also conduct experiments to verify that our NSO system can explore the flexibility of EONs in the optical layer when creating/expanding vDCs. Here, for the vDC shown in Fig. 8(b), the NSO system allocates 212.5 GHz optical spectra (*i.e.*, central wavelengths ranging

within $[1545.87, 1547.47]$ nm) to the vSD-EON that interconnects the vDC fragments. Then, to confirm that the related flexible-grid based optical switching has been configured correctly, we set up one and two lightpaths in the vSD-EON from VN2 to VN1 and measure the input and output spectra on the corresponding substrate optical switches. The lightpaths are for 10 Gbps capacity, and based on the mapping relation in Table 1, VN1 and VN2 get mapped onto SN5 and SN1 in the EON in Fig. 8(a), respectively. Hence, we measure the input spectrum from the SN1 in public DC1 to the SN1 in the EON and the output spectrum from the SN5 in the EON to the SN1 in private DC, and show the results in Fig. 10. In Fig. 10(a), we observe that a 10 Gbps lightpath with a central wavelength of 1546.92 nm gets established successfully in the vSD-EON since its spectrum can be seen in both the input and output spectra, while the background lightpath (*i.e.*, with a central wavelength of 1547.72 nm) to the SN1 in the EON, which does not belong to the vDC, gets switched away and cannot be observed in the output spectrum. The results of the two-lightpath case are shown in Fig. 10(b), where we add another 10 Gbps lightpath with a central wavelength of 1546.12 nm to expand the bandwidth capacity from VN2 to VN1 in the vSD-EON. We can see that the spectra of the two lightpaths in the vDC still get switched correctly.

5.3. Network orchestration in vDC for Hadoop M/R optimization

After obtaining the vDC, the SP's M/R-EH decides to add a new slave node to the Hadoop cluster for M/R optimization. Specifically, it instructs the VMM in the orchestrator to copy the image of Slave1 to VM1 in Fig. 8(b) and configure it as the new slave node. To achieve this, a new path is activated in the vDC to support the VM duplication, and the related experimental results are shown in Fig. 11. The Wireshark captures of the related control messages are in Fig. 11(a). **Step 1** indicates that NVH of DC1 receives the *Packet_In* message from the edge switch that connects to VM1 in public DC1, and it translates the message to forward to the ONOS. In **Step 2**, the ONOS receives the *Packet_In* message, and calculates path between VM1 and Server1. In **Steps 3** and **4**, ONOS sends *Flow_Mod* messages through NVH of DC1 to SN2 and SN1 in DC1 to activate the VL for VN5-VN4. In **Step 5**, the *Flow_Mod* messages with optical transport protocol extensions (w/ OTPE) are forwarded to the OF-AGs of SN5, SN2, SN1 in the EON through NVH of EON, which are used to configure the corresponding BV-WSS' for setting up a new lightpath on the VL for VN1-VN2. The new lightpath uses a center wavelength of 1546.92 nm and provides 10 Gbps capacity to support the inter-DC connection between public DC1 and the private DC, and it is set up in **Step 6**. Figure 11(b) plots the optical spectrum of the new lightpath. Finally, in **Step 7**, the ONOS sends a *Flow_Mod* message directly to SN1 in the private DC to activate the path segment to Server1, which hosts Slave1. Hence, the end-to-end path for the VM duplication has been established. Note that, the VM duplication requires two-way communication, and the reverse path is activated similarly. Therefore, we omit the procedure here for conciseness.

With the newly-added slave node, namely Slave2, the M/R jobs have more resources to use. We redo the WordCount experiments in Section 5.1 with the same parameters, and the new results on CPU and memory utilizations on Slave1 are plotted in Figs. 12(a) and 12(b), respectively, for 100 MB workload. By comparing the new results with those in Figs. 6(d) and 6(e), we can see that the utilizations of IT resources become much less after the vDC expansion. The performance improvement achieved by the vDC expansion can be further verified by the results on job completion time in Fig. 12(c). After the vDC expansion and adding Slave2 in the Hadoop cluster, the NSO system can shorten the job completion time effectively, especially for the case with 500 MB data to analyze. Moreover, the job completion time increases linearly with the data size this time. These results confirm the vDC expansion's effectiveness on M/R optimization.

Next, we conduct an experiment to show that our proposed system can leverage the network orchestration in the vDC to further optimize M/R jobs, when there is other services running

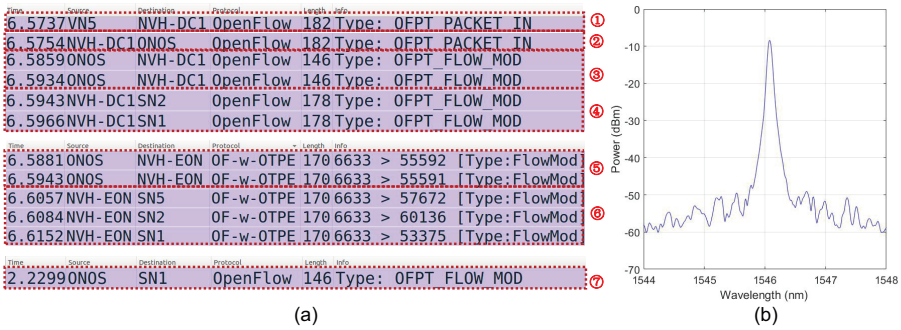

Fig. 11. (a) Experimental results for the path activation to duplicate a VM in vDC, and (b) optical spectrum of new lightpath.

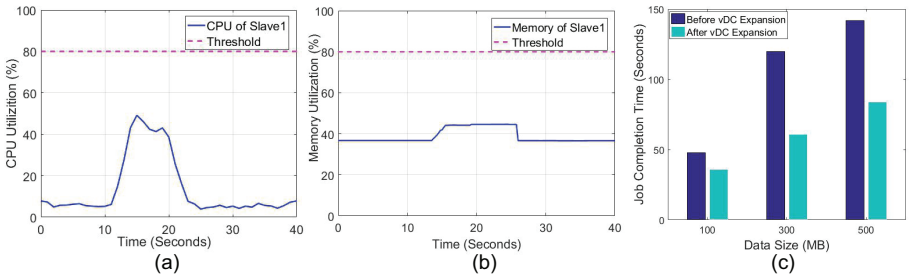

Fig. 12. Experimental results for running M/R jobs in the vDC after expansion, (a) CPU usage on Slave1 with 100 MB workload, (b) memory usage on Slave1 with 100 MB workload, and (c) job completion time.

simultaneously. Specifically, we assume that a backup service is suddenly started to transfer data from Server3 to VM2 in the vDC. Hence, the VL for VN4-VN5 (*i.e.*, with a bandwidth of 1 Gbps) would become the bottleneck to limit the performance of the M/R and backup services. The control messages collected on the M/R-EH to resolve the network bottleneck are shown in Fig. 13(a). In **Step 1**, the ONOS detects the abnormal bandwidth usage on the VL for VN4-VN5 (*i.e.*, over 70%), and sends an alert message to the M/R-EH. The details of the alert message are in Fig. 13(b). Then, in **Step 2**, the M/R-EH decides to migrate Slave2 on VM1 to VM3 and sends the message in Fig. 13(c) to the VMM. The VMM triggers the VM migration and sends a confirmation to the M/R-EH when the VM migration has been completed in **Step 3**. The detailed process to move Slave2 for M/R optimization is shown in Fig. 13(d), which indicates that the VM migration takes 9 seconds and uses an average throughput of 909 Mbps. When the VM migration has been done, the M/R-EH instructs the ONOS to reroute the related paths in the vDC to reconnect the Hadoop cluster. Finally, to verify the effectiveness of the network orchestration, we run M/R jobs for Teragen in the Hadoop cluster to create 100 MB, 500 MB and 1 GB random data, and compare the job completion time before and after the VM migration. Here, Teragen is used because it consumes more bandwidth than WordCount. The results in Fig. 13(e) confirm that the VM migration can reduce the job completion time significantly.

6. Conclusion

We designed and implemented an NSO system that can create and expand vDCs across optical/packet domains on-demand for M/R optimization. The proposed NSO system was realized in a small-scale network testbed that includs four optical/packet domains, and we conducted ex-

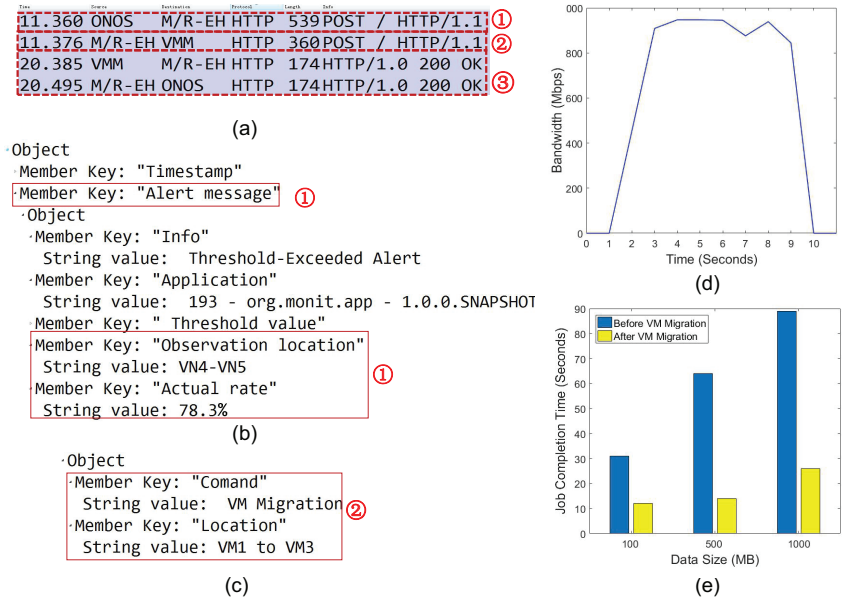

Fig. 13. Experimental results for network orchestration to avoid bandwidth bottleneck, (a) control messages collected on M/R-EH, (b) details of the alert message to report bandwidth usage, (c) details of the message from M/R-EH to VMM for VM migration, (d) process of VM migration to move Slave2 for M/R optimization, and (e) job completion time.

periments in it to demonstrate the whole operations of the data, control and management planes. The experimental results verified that application-driven on-demand vDC expansion across optical/packet domains can be achieved for M/R optimization, and after being provisioned with a vDC, the SP can fully control the vDC network and further optimize the M/R jobs in it with network orchestration.

7. Funding

NSFC Projects (61701472 and 61771445); CAS Key Project (QYZDY-SSW-JSC003); NGBWMCN Key Project (2017ZX03001019-004); China Postdoctoral Science Foundation (2016M602031); Fundamental Research Funds for the Central Universities (WK2100060021).

References and links

1. P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," IEEE Netw. **29**, 36–42 (2015).
2. A. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined WAN," in Proc. of ACM SIGCOMM, (2013), pp. 3–14.
3. J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," J. Lightw. Technol. **33**, 3005–3015 (2015).
4. P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," IEEE Trans. Multimedia **17**, 1297–1308 (2015).
5. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux virtual machine monitor," in Proc. of OLS, (2007), pp. 225–230.
6. S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo, "Explicit path control in commodity data centers: Design and applications," IEEE/ACM Trans. Netw. **24**, 2768–2781 (2016).
7. S. Zhao and D. Medhi, "Application-aware network design for Hadoop MapReduce optimization using software-defined networking," IEEE Trans. Netw. Serv. Manag. **14**, 804–816 (2017).
8. D. Borthakur, *The Hadoop Distributed File System: Architecture and Design* (Apache Software Foundation, 2007).

9. H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in Proc. ACM SIGCOMM, (2011), pp. 242–253.
10. An introduction to network slicing. [Online] https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf.
11. Network automation and orchestration. [Online] https://www.juniper.net/assets/cn/zh/local/pdf/whitepapers/2000541-en.pdf.
12. L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," J. Lightw. Technol. **32**, 450–460 (2014).
13. L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," IEEE/ACM Trans. Netw. **24**, 3648–3661 (2016).
14. "Spectral grids for WDM applications: DWDM frequency grid," ITU-T Rec. G.694.1, Feb. 2012.
15. Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," J. Lightw. Technol. **31**, 15–22 (2013).
16. L. Gong, X. Zhou, X. Liu, W. Zhao, W. Lu, and Z. Zhu, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," J. Opt. Commun. Netw. **5**, 836–847 (2013).
17. K. Christodoulopoulos, I. Tomkos, and E. Varvarigos, "Time-varying spectrum allocation policies and blocking analysis in flexible optical networks," IEEE J. Sel. Areas Commun. **31**, 13–25 (2013).
18. W. Fang, M. Zeng, X. Liu, W. Lu, and Z. Zhu, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," IEEE Commun. Lett. **20**, 1539–1542 (2016).
19. P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," J. Lightw. Technol. **35**, 5335–5346 (2017).
20. M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," J. Lightw. Technol. **34**, 3330–3341 (2016).
21. T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. Jackson, A. Lambeth, R. Lenglet, S. Li, A. Padmanabhan, J. Pettit, B. Pfaff, R. Ramanathan, S. Shenker, A. Shieh, J. Stribling, P. Thakkar, D. Wendlandt, A. Yip, and R. Zhang, "Network virtualization in multi-tenant datacenters," in Proc. of NSDI, (2014), pp. 203–216.
22. A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," IEEE Commun. Surveys Tuts. **18**, 655–685 (2016).
23. H. Chen, J. Zhang, Y. Zhao, J. Deng, W. Wang, R. He, W. Zhou, Y. Ji, H. Zhang, Y. Lin, and H. Yang, "Demonstration of cloud resources integrated provisioning over multi-domain software defined optical networks," in Proc. of ECOC, (2014), pp. 1–3.
24. W. Fang, M. Lu, X. Liu, L. Gong, and Z. Zhu, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," J. Opt. Commun. Netw. **7**, 314–324 (2015).
25. R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep., pp. 1–13 (2009).
26. A. Al-Shabibi, M. Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: Make your virtual SDNs programmable," in Proc. of ACM HotSDN, (2014), pp. 25–30.
27. S. Li, K. Han, H. Huang, Q. Sun, J. Liu, S. Zhao, and Z. Zhu, "SR-PVX: A source routing based network virtualization hypervisor to enable POF-FIS programmability in vSDNs," IEEE Access **5**, 7659–7666 (2017).
28. R. Vilalta, R. Munoz, R. Casellas, R. Martinez, S. Peng, M. Channegowda, T. Vlachogiannis, R. Nejabati, D. Simeonidou, X. Cao, T. Tsuritani, and I. Morita, "Dynamic multi-domain virtual optical network deployment with heterogeneous control domains," J. Opt. Commun. Netw. **7**, A135–A141 (2015).
29. A. Hammad, A. Aguado, S. Peng, R. Vilalta, A. Mayoral, R. Casellas, R. Martinez, R. Munoz, R. Nejabati, and D. Simeonidou, "On-demand virtual infrastructure composition over multi-domain and multi-technology networks," in Proc. of OFC, (2016), pp. 1–3.
30. J. Yin, J. Guo, B. Kong, and Z. Zhu, "Demonstration of survivable vSD-EON slicing with automatic data plane restoration to support reliable video streaming," in Proc. of OFC, (2017), pp. 1–3.
31. J. Yin, J. Guo, B. Kong, H. Yin, and Z. Zhu, "Experimental demonstration of building and operating QoS-aware survivable vSD-EONs with transparent resiliency," Opt. Express **25**, 15468–15480 (2017).
32. Z. Zhu, B. Kong, J. Yin, S. Zhao, and S. Li, "Build to tenants' requirements: On-demand application-driven vSD-EON slicing," J. Opt. Commun. Netw. **10**, A206–A215 (2018).
33. Y. Lee, G. Bernstein, N. So, T. Kim, K. Shiomoto, and O. de Dios, "Research proposal for cross stratum optimization (CSO) between data centers and networks," IETF Draft [Online]. https://tools.ietf.org/html/draft-lee-cross-stratum-optimization-datacenter-00.
34. D. Ceccarelli and Y. Lee, "Framework for abstraction and control of transport networks," IETF Draft [Online]. https://tools.ietf.org/html/draft-ceccarelli-actn-framework-07.
35. Network Function Virtualisation (NFV), Oct. 2014. [Online] https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.
36. D. Cheng, X. Zhou, P. Lama, J. Wu, and C. Jiang, "Cross-platform resource scheduling for Spark and MapReduce on YARN," IEEE Trans. Comput. **66**, 1341–1353 (2017).
37. S. Tang, B. Lee, and B. He, "Dynamic job ordering and slot configurations for MapReduce workloads," IEEE Trans. Serv. Comput. **9**, 4–17 (2016).

38. Open vSwitch [Online]. http://www.openvswitch.org/.
39. C. Chen, X. Chen, M. Zhang, S. Ma, Y. Shao, S. Li, M. Suleiman, and Z. Zhu, "Demonstrations of efficient online spectrum defragmentation in software-defined elastic optical networks," J. Lightw. Technol. **32**, 4701–4711 (2014).
40. Z. Zhu, C. Chen, S. Ma, L. Liu, X. Feng, and B. Yoo, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," J. Lightw. Technol. **33**, 1508–1514 (2015).
41. X. Chen, Z. Zhu, L. Sun, J. Yin, S. Zhu, A. Castro, and B. Yoo, "Incentive-driven bidding strategy for brokers to compete for service provisioning tasks in multi-domain SD-EONs," J. Lightw. Technol. **34**, 3867–3876 (2016).
42. L. Sun, X. Chen, and Z. Zhu, "Multi-broker based service provisioning in multi-domain SD-EONs: Why and how should the brokers cooperate with each other?" J. Lightw. Technol. **35**, 3722–3733 (2017).
43. The OpenStack Foundation [Online]. https://www.openstack.org/foundation/.
44. Open Network Operating System (ONOS) [Online]. http://onosproject.org/.
45. virt-manager [Online]. https://virt-manager.org/.
46. M. Bolte, M. Sievers, G. Birkenheuer, O. Niehorster, and A. Brinkmann, "Non-intrusive virtualization management using libvirt," in Proc. of DATE, (2010), pp. 574–579.
47. influxDB [Online]. https://www.influxdata.com/.
48. Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vNFG) provisioning in multidomain elastic optical networks," J. Lightw. Technol. **35**, 2712–2723 (2017).
49. M. Turowski and A. Lenk, "Vertical scaling capability of OpenStack," in Proc. of ICSOC, (2014), pp. 351–362.