Scheduling Virtual Network Reconfigurations in Parallel in Hybrid Optical/Electrical Datacenter Networks

Xiaoqin Pan, Sicheng Zhao, Hao Yang, Shaofei Tang, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—A hybrid optical/electrical datacenter network (HOE-DCN) integrates the benefits of electrical packet switching (EPS) and optical circuit switching (OCS) for better architectural scalability. Meanwhile, as each network service usually involves multiple virtual machines (VMs) that form a virtual network (VNT), how to reconfigure the VNTs in an HOE-DCN to adapt to the dynamic network environment becomes an interesting and important problem to tackle. In this work, we optimize the procedure of VNT reconfigurations that remap VNTs to given virtual network embedding (VNE) schemes, *i.e.*, scheduling the required VM migrations and optical cross-connect (OXC) reconfiguration properly such that the overall VNT reconfiguration latency can be minimized. We first assume that all the VM migrations should be scheduled before the OXC reconfiguration to minimize service interruptions, and formulate a mixed integer linear programming (MILP) model to solve the scheduling problem exactly. Then, with the same assumption, we propose a time-efficient heuristic to schedule parallel VM migrations in batches such that the bandwidth competition can be significantly relieved. Finally, we divide the OXC reconfiguration into several progressive steps, and design a joint scheduling algorithm to arrange VM migrations together with the OXC reconfiguration steps.**

***Index Terms*—Hybrid optical/electrical datacenter networks, Network virtualization, VM migration, Parallel reconfiguration.**

I. Introduction

THE recent rising of cloud computing and data science has made datacenter (DC) one of the most important facilities in the Internet [1], while the more than 27% of annual growth rate will drive intra-DC traffic to skyrocket in the near future [2]. Therefore, datacenter networks (DCNs) have to address the upward pressure from multiple aspects for being more flexible, scalable and energy-efficient [3]. This promotes people to architect hybrid optical/electrical DCN (HOE-DCN) [4–6], which upgrades the inter-rack network from purely based on electrical packet switching (EPS) to one that orchestrates EPS with optical circuit switching (OCS). Specifically, in addition to EPS-based switches that are arranged in a hierarchical topology, top-of-rack (ToR) switches can also be interconnected with one or more optical cross-connects (OXCs) whose throughputs are much higher. Hence, HOE-DCN can integrate the merits of OCS (*e.g.*, larger bandwidth capacity and higher energy efficiency [7–9]) with those of EPS, and satisfy various quality-of-service (QoS) demands from network services more cost-effectively.

Meanwhile, in a DCN, each network service usually involves the cooperation of multiple virtual machines (VMs). For instance, the well-known Apache Hadoop [10] allows for the distributed processing of large data sets across clusters of VMs. Therefore, the VM cluster of a network service can be modeled as one virtual network (VNT), where each of its virtual nodes (VNs) is a VM, and the communication channel between a VM pair is a virtual link (VL). Then, the deployment of the network service is equivalent to solving the famous virtual network embedding (VNE) problem [11–13], which finds proper resource allocations to embed all the VNs and VLs in a VNT on substrate nodes and substrate paths, respectively. However, as the network environment of a DCN is usually dynamic, the optimality of a network service's initial VNE scheme can be progressively degraded over time. This motivates people to reconfigure the VNE schemes of network services adaptively (*e.g.*, re-balancing the IT resource usages on server racks [14]), especially in HOE-DCNs [15, 16].

Note that, the actual implementation of VNT reconfigurations in HOE-DCNs needs to solve two challenging problems. Firstly, we need to obtain new VNE schemes for VNTs [17, 18], based on the status of an HOE-DCN and an optimization objective (*e.g.*, load-balancing or energy-saving). Secondly, given the new VNE schemes, we need to design the procedure to accomplish the VNT reconfigurations quickly in the HOE-DCN. The first problem has been studied in [19], while the second one has not been fully explored yet. Nevertheless, the second problem is actually more challenging, and the quality of its solution can significantly affect the performance of VNT reconfigurations in an HOE-DCN. This is because the procedure of VNT reconfigurations involves the scheduling of VM migrations, VL remappings and OXC reconfigurations.

As a VM migration usually takes tens of seconds or even minutes [20], it is the major contributor to the latency of VNT reconfiguration. Meanwhile, the scheduling of VL remappings is not trivial either, because VNT reconfiguration can change the inter-rack topology of an HOE-DCN due to the one-to-one connectivity of an OXC [21]. Fig. 1 gives an example on this. The top of the figure shows a VNT that consists of three VMs, and its VNE scheme is illustrated in the HOE-DCN below. Specifically, *VMs* 1-3 are mapped on *Racks* 1, 3 and 4, respectively, and the VL between *VMs* 1 and 3 is mapped on an optical connection between *Racks* 1 and 4 through the

X. Pan, S. Zhao, H. Yang, S. Tang and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (email: zqzhu@ieee.org).

X. Pan is also with the Engineering Technology Center, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

H. Yang is also with the School of Information Engineering, Southwest University of Science and Technology, Mianyang, Sichuan 621010, China.

Manuscript received on December 22, 2020.

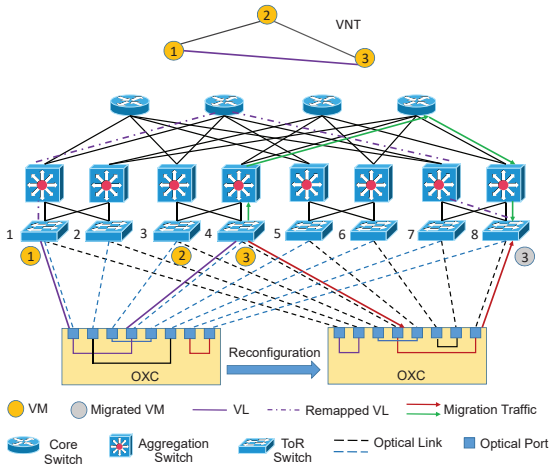


Fig. 1. Example on VNT reconfiguration in an HOE-DCN.

OXC. Then, the VNT reconfiguration needs to remap *VM* 3 to *Rack* 8. Before the OXC is reconfigured to connect *Racks* 4 and 8, the VM migration can only use the EPS part of the inter-rack network, while after the OXC has been reconfigured, it can use both the EPS and OCS parts. We can see that VL remappings in an HOE-DCN can modify the bandwidth throughput between ToR switch pairs, and might in turn affect the duration of subsequent VM migrations. Finally, after *VM* 3 has been migrated to *Rack* 8, the VL between *VMs* 1 and 3 can only be mapped on EPS-based links, since *Racks* 1 and 8 are not connected through the OXC.

The example in Fig. 1 suggests that we have to schedule VM migrations and VL remappings in a highly correlated way. This makes the problem of scheduling VNT reconfigurations different from and more complex than the scheduling of VM migrations in conventional DCNs [22, 23]. Moreover, if we would like to use the live migration scenario to minimize service interruptions [24] (*i.e.*, hitless VNT reconfigurations), the dependencies among VMs, VLs and substrate links (SLs) can be even more complex because we need to consider the traffic of VM migrations and normal network services jointly.

Previously, we conducted a preliminary study in [25] on the problem mentioned above, and designed two algorithms that can schedule parallel VM migrations in one shot and in multiple shots, respectively, to accomplish VNT reconfigurations in an HOE-DCN. The simulation results suggested that the multi-shot algorithm can achieve shorter VNT reconfiguration latency, because it relieved the bandwidth competition among VM migrations. However, the study in [25] still needs to be improved from two perspectives. Firstly, bandwidth bottleneck can happen on either the source or the destination ToR switch of a VM migration, but the network model in [25] ignored the bottleneck on destination ToR switches. Secondly, the multi-shot algorithm can be improved to better utilize the bandwidth resources in an HOE-DCN for faster VNT reconfigurations.

In this paper, we extend our work in [25] to perform a more comprehensive study on the scheduling of VNT reconfigurations in an HOE-DCN. We first improve the network model, assume that all the VM migrations are scheduled before the OXC reconfiguration to avoid unnecessary service interruptions, and formulate a mixed integer linear programming (MILP) model based on the migration unit (MU) to solve the problem exactly. Then, with the same assumption, we propose a time-efficient heuristic to schedule parallel VM migrations in batches. Finally, we remove the restriction that the OXC should be reconfigured in one step, divide it into a few progressive steps, and propose a joint scheduling algorithm that can arrange VM migrations together with the OXC reconfiguration steps. Extensive simulations are conducted to evaluate our proposals, and the results indicate that the joint scheduling algorithm achieves the shortest VNT reconfiguration latency.

The rest of the paper is organized as follows. In Section II, we survey the related work. Section III describes the problem definition. The MILP model is formulated in Section IV, and we explain the principle of algorithm design in Section V. Section VI proposes the algorithms. Simulations are discussed in Section VII. Finally, Section VIII summarizes the paper.

II. Related Work

For a variety of reasons, the VNTs of network services in a DCN need to be reconfigured [14]. For example, VNT reconfigurations help to re-balance resource usages [26, 27], reduce job completion time of service tasks [28], and save energy consumption [29]. Without considering HOE-DCNs, the studies in [30–32] have considered the scheduling of VM migrations in traditional DCNs to optimize VNT reconfigurations. For instance, the study in [31] leveraged the centralized control provided by software-defined networking (SDN) [33–36] to maximize the bandwidth allocated to VM migrations.

Note that, the key problem of scheduling VM migrations is to allocate and schedule bandwidth for data transfers, which has been addressed in a few studies for network environments related to DCs [37–41]. Nevertheless, no matter which assumption that the studies were based on (*i.e.*, the network topology is constant [37–39] or time-varying [40, 41]), they took the network topology as a known input and did not try to schedule topology changes together with data transfers. This makes these studies fundamentally different from the one considered in this work, because we need to jointly schedule VM migrations and OXC reconfiguration steps to optimize the procedure of VNT reconfigurations in an HOE-DCN.

Previously, we have investigated the VNT reconfigurations in an HOE-DCN in [3, 6, 16, 21]. By leveraging the idea of knowledge-defined networking (KDN) [42], we proposed a network orchestration framework based on deep reinforcement learning in [3], to improve the management agility of HOE-DCNs. In [6, 16], the network orchestration framework was experimentally demonstrated to verify that it can coordinate the EPS and OCS parts of the inter-rack network of an HOE-DCN to achieve application-aware network service provisioning. We also designed deterministic algorithms in [21] to calculate the new VNE schemes for re-balancing the IT resource usages in an HOE-DCN. However, none of these studies have optimized the procedure of VNT reconfigurations in an HOE-DCN, *i.e.*, how to jointly schedule VM migrations and VL remappings to minimize the overall VNT reconfiguration latency. To the best of our knowledge, our preliminary study in [25] is the only

one that considered the problem in the literature. Nevertheless, it still needs to be improved in a few aspects.

Other than HOE-DCNs, the recent studies in [43–45] experimentally demonstrated optical DCNs (O-DCNs) by leveraging the advances on optical switching. Specifically, they showed that with new optical switching technologies, the inter-rack network of a DCN can be reconfigured as fast as with EPS, and thus the EPS-based inter-rack network might be completely replaced by an optical network in the future. Although the proposals on O-DCNs in [43–45] are promising, our contributions in this work are orthogonal to theirs. This is because the focus of this work is to design algorithms for scheduling the VM migrations of VNT reconfiguration in an HOE-DCN, while the aforementioned studies concentrated on designing and demonstrating novel network architectures for O-DCNs.

III. Problem Description

This section defines the network model and explains the problem of scheduling VNT reconfigurations in an HOE-DCN.

A. Network Model

We model the topology of the HOE-DCN's inter-rack network as a graph $G_s(V_s, E_s)$, where V_s and E_s represent the sets of substrate nodes (SNs) and substrate links (SLs), respectively. Here, each SN $v_s \in V_s$ denotes a server rack that consists of a rack of servers and a ToR switch, while each SL $e_s = (v_s, u_s) \in E_s$ is a network connection in the inter-rack network, which can use either EPS or OCS. Similar to that in [25], we assume that the EPS part of the inter-rack network is based on the well-known k-ray fat-tree topology [46], which is non-blocking, *i.e.*, the bandwidth to/from a ToR switch through it is only limited by the data-rate of the ToR switch's EPS port. The data-rate of an EPS port on ToR switch v_s is B_{v_s}.

Meanwhile, depending on the OXC's configuration, a pair of ToR switches can also communicate through the OCS part of the inter-rack network (*i.e.*, an OXC). However, due to its one-to-one connectivity, each ToR switch can talk with one and only one ToR switch at any given time through the OXC. If ToR switches v_s and u_s are connected through the OXC, we denote the corresponding bandwidth capacity as $B_{(v_s,u_s)}$. Later on, if v_s wants to talk with a ToR switch other than u_s through the OXC, we need to trigger an OXC reconfiguration, which can be done within hundreds of milliseconds by leveraging the SDN-based centralized control plane [16].

In this work, we assume that the OCS part of the inter-rack network only consists of one OXC, even though an optical DCN can have inter-cluster data transfers across multiple hops of optical switching [44]. This assumption is introduced due to two practical considerations. Firstly, VM migration is costly in terms of bandwidth usage [47], and thus we should make sure that its data transfers use as few hops of optical switching as possible. Secondly, as a commercial OXC can support a relatively large number of ports, *e.g.*, a configuration of 384×384 ports is feasible [48], we can use one OXC to interconnect hundreds of ToR switches. Hence, it might not be necessary to migrate VMs across multiple OXCs for VNT reconfiguration, for modular HOE-DCN management. Hence, we treat all the racks, which are interconnected by an OXC, as a large point-of-delivery (PoD), and for each VNT reconfiguration, the VMs can only be remapped within it.

The topology of the VNT of a network service is modeled as an undirected graph $G_r(V_r, E_r)$, where V_r and E_r are the sets of virtual nodes (VNs) and virtual links (VLs), respectively. Each VN $v_r \in V_r$ is a VM that is deployed in a server rack and runs the tasks of the network service, and its memory size is defined as c_{v_r}. The VL $e_r = (v_r, u_r) \in E_r$ interconnects VMs v_r and u_r with a bandwidth requirement of $b_{(v_r,u_r)}$.

B. Scheduling of VNT Reconfigurations in an HOE-DCN

In this work, we optimize the procedure of VNT reconfigurations in an HOE-DCN. Specifically, for a VNT $G_r(V_r, E_r)$, its original and new VNE schemes can be denoted as

$$\mathcal{F} = \begin{cases} \mathcal{F}_N: & V_r \mapsto V_s, \\ \mathcal{F}_L: & E_r \mapsto P_s, \end{cases} \quad \mathcal{F}' = \begin{cases} \mathcal{F'}_N: & V_r \mapsto V_s, \\ \mathcal{F'}_L: & E_r \mapsto P_s, \end{cases} \tag{1}$$

where $\mathcal{F}_N$ and $\mathcal{F}_L$ are the original node and link mapping schemes, respectively, $\mathcal{F}'_N$ and $\mathcal{F}'_L$ are the new mapping schemes, P_s denotes the set of pre-calculated substrate paths in $G_s(V_s, E_s)$. A VNT reconfiguration needs to change $\mathcal{F}_N$ and $\mathcal{F}_L$ to $\mathcal{F'}_N$ and $\mathcal{F'}_L$, respectively (*i.e.*, reconfiguring certain VMs and VLs to use new substrate elements). Hence, we need to schedule the VM migrations and VL remappings to achieve the shortest overall VNT reconfiguration latency[1]. Here, as VM migrations usually take much longer time than VL remappings, we only consider the time used for VM migrations when calculating the latency of the VNT reconfiguration.

To minimize the service interruption during VNT reconfiguration, we assume that live VM migration [47] is used to remap each VM. This means that each VM is migrated with the "make-before-break" scenario. Specifically, the VM and all of its VLs keep running on the original server, until the VM's memory data has been transferred to the new server in $\mathcal{F'}_N$ and the VM and its VLs have been activated there. Note that, as the VM migrations need to be conducted when the HOE-DCN runs network services, they can only be scheduled with the residual bandwidth left by the active network services.

To facilitate the analysis of the process of VM migrations, we introduce the following definition.

Definition 1. *A **migration unit (MU)** m includes all the VMs whose source and destination ToR switches are the same. With $\mathcal{F}_N$ and $\mathcal{F'}_N$, we obtain all the MUs to store in set $\mathbb{M}$.*

Fig. 2 shows illustrative examples on the MUs. For instance, *MU* 1 includes two VMs that need to be migrated from *Rack* 1 to *Rack* 2. Hence, by considering all the VMs in an MU jointly, we can schedule the VM migrations more efficiently, *e.g.*, all the VMs in *MU* 3 can be migrated through both the EPS and OCS parts of the inter-rack network, while those in *MUs* 1 and 2 can only use the EPS part. Therefore, to reduce the overall VNT reconfiguration latency, we need to schedule the data transfers of MUs in proper sequence and allocate bandwidth to them accordingly, based on the status of the HOE-DCN.

[1]As we focus on optimizing the transition from $\mathcal{F}$ to $\mathcal{F}'$, $\mathcal{F}$ and $\mathcal{F}'$ are known inputs to the algorithms designed in this work. Hence, how or for what purpose $\mathcal{F}'$ is calculated is out of the scope of this paper.

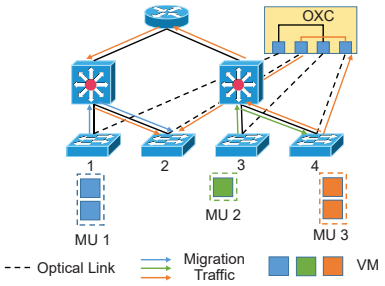


Fig. 2. Example on scheduling VM migrations based on MUs.

IV. MILP for Scheduling VNT Reconfigurations

Migrating MUs in parallel might help to reduce the overall VNT reconfiguration latency. However, in addition to VM migrations, the VNT reconfigurations in an HOE-DCN might also invoke OXC reconfiguration to remap VLs. Note that, to avoid unnecessary service interruptions during the VNT reconfiguration, each VL should be remapped after the related VM(s) have been migrated, if required. Hence, for simplicity, we first assume that all the VM migrations should be scheduled before the OXC reconfiguration, and formulate an MILP to tackle the scheduling of VM migrations based on MUs in this section. Then, in Sections V and VI, we will remove the restriction that the OXC should be reconfigured in one step, divide the OXC reconfiguration into a few progressive steps, and study how to schedule VM migrations together with OXC reconfiguration steps, for more efficient scheduling algorithms.

The MILP is based on a discrete time model, *i.e.*, the bandwidth allocated to VM migrations can be updated at intervals spaced by a time slot (TS) of Δt [37]. The MILP's parameters and variables are listed in Tables I and II, respectively.

Objective:

The objective is to minimize the overall migration time.

$$\text{Minimize} \quad t_{max}, \tag{2}$$

where the value of t_{max} can be obtained as

$$t_{max} = \max_{v_s \in V_s} (N_{v_s}) \cdot \Delta t, \tag{3}$$

where N_{v_s} denotes the last time slot (TS) used by rack v_s to migrate the VMs on it, and satisfies the following equation

$$N_{v_s} \geq t \cdot z_m^i \cdot f_i^t \cdot s_m^{v_s}, \ \forall m \in \mathbb{M}, v_s \in V_s, i \in I, t \in [1, T], \tag{4}$$

where t denotes the index of a TS, and the term $z_m^i \cdot f_i^t \cdot s_m^{v_s}$ equals 1 if the VM migration(s) from rack v_s use TS t, and 0 otherwise. Eq. (3) can be linearized as

$$t_{max} \geqslant t \cdot z_m^i \cdot f_i^t \cdot s_m^{v_s} \cdot \Delta t, \quad \forall m, \ v_s, \ i, \ t. \tag{5}$$

Constraints:

$$\sum_{m \in \mathbb{M}} x_m^t \cdot s_m^{v_s} \leq b_{v_s}^{\text{out},t}, \quad \forall v_s, \ t, \tag{6}$$

$$\sum_{m \in \mathbb{M}} x_m^t \cdot d_m^{v_s} \leq b_{v_s}^{\text{in},t}, \quad \forall v_s, \ t. \tag{7}$$

Eqs. (6)-(7) ensure that the EPS bandwidth allocated for migrating MUs from/to rack v_s does not exceed the corresponding available bandwidth capacity in the t-th TS.

$$y_m^t \cdot s_m^{u_s} \cdot d_m^{v_s} \leq b_{(u_s, v_s)}^t, \quad \{v_s, u_s \in V_s : v_s \neq u_s\}, \ \forall m, \ t. \tag{8}$$

TABLE I
Definitions of Parameters

Substrate Network (SNT)	
$G_s(V_s, E_s)$	The HOE-DCN's inter-rack topology (*i.e.*, the SNT).
$b_{v_s}^{\text{in}}/b_{v_s}^{\text{out}}$	The available EPS bandwidth capacity to/from rack $v_s \in V_s$, right before the VM migrations.
$b_{(u_s, v_s)}$	The available OCS bandwidth capacity from rack u_s to rack v_s before the VM migrations, and it equals 0 if u_s is not connected with v_s through the OXC.
it_{v_s}	The available resources capacity of rack v_s before the VM migrations.
$L_{(u_s, v_s)}$	The boolean parameter that equals 1 if rack u_s can talk with v_s through the OXC, and 0 otherwise.
B_{v_s}	The EPS bandwidth capacity on rack v_s.
$B_{(v_s, u_s)}$	The OCS bandwidth capacity between racks v_s and u_s.
Virtual Network (VNT)	
$\mathbb{M}$	The set of MUs.
c_m	The total memory data of all the VMs in MU $m \in \mathbb{M}$.
$s_m^{u_s}/d_m^{u_s}$	The boolean parameter that equals 1 if rack u_s is the source/destination rack of MU $m \in \mathbb{M}$, and 0 otherwise.
s_m/d_m	The source/destination rack of MU $m \in \mathbb{M}$.
$\tilde{s}_m^{u_s}/\tilde{d}_m^{u_s}$	The boolean parameter that equals 1 if rack u_s connects with the destination/source rack of MU $m \in \mathbb{M}$ through the OXC, and 0 otherwise.
$\tilde{x}_m/\hat{x}_m$	The EPS bandwidth usage of the VLs to be remapped, in its source/destination rack of MU $m \in \mathbb{M}$.
$\tilde{y}_m$	The OCS bandwidth of the VLs that are mapped on the optical connection to the source rack of MU $m \in \mathbb{M}$ before its migration.
$\hat{y}_m$	The OCS bandwidth of the VLs to be remapped on the optical connection to the destination rack of MU $m \in \mathbb{M}$ after its migration.
D_m	The minimum bandwidth that needs to be allocated to MU $m \in \mathbb{M}$ for enabling its migration.
Auxiliary Parameters	
T	The maximum number of time slots (TS') that can be used to accomplish the VM migrations.
Δt	The duration of a TS.
I	The set of feasible durations (continuous TS') that can be used to migrate MUs. For instance, if $T = 3$, we can obtain $I = \{1, 2, 3, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$. This parameter is introduced to ensure that each VM migration will not be interrupted in the middle of operation.
f_i^t	The boolean parameter that equals 1 if the duration $i \in I$ includes the t-th TS, and 0 otherwise.

Eq. (8) ensures that the OCS bandwidth allocated for migrating MU m from rack u_s to rack v_s dose not exceed the corresponding available bandwidth capacity in the t-th TS.

$$x_m^t \geq 0, \ y_m^t \geq 0, \quad \forall m, \ t. \tag{9}$$

Eq. (9) ensures non-negative bandwidth allocations.

$$\begin{aligned} x_m^t &\leq \sum_{i \in I} z_m^i \cdot f_i^t \cdot B_{v_s}, \quad \forall m, \ t, \\ y_m^t &\leq \sum_{i \in I} z_m^i \cdot f_i^t \cdot B_{(v_s, u_s)}, \quad \forall m, \ t. \end{aligned} \tag{10}$$

Eq. (10) ensures that when it is not within the selected duration of MU m, no bandwidth is allocated to migrate it.

$$\sum_{i \in I} [x_m^t + y_m^t \cdot L_{(s_m, d_m)} - D_m] \cdot z_m^i \cdot f_i^t \geq 0, \quad \forall m, \ t. \tag{11}$$

Eq. (11) ensures that the bandwidth allocated for migrating MU m should not be smaller than the required minimum bandwidth. Note that, although Eq. (11) is nonlinear because

TABLE II
DEFINITIONS OF VARIABLES

t_{max}	The overall VM migration time in TS'.
x_m^t/y_m^t	The EPS/OCS bandwidth allocated to migrate MU m in the t-th TS.
$b_{v_s}^{\text{in},t}/b_{v_s}^{\text{out},t}$	The available EPS bandwidth to/from rack v_s in t-th TS.
$b_{(u_s,v_s)}^t$	The available OCS bandwidth from rack u_s to rack v_s in the t-th TS.
$it_{v_s}^t$	The available IT resources on rack v_s in the t-th TS.
S_m^t	The boolean variable that equals 1 if MU m is scheduled to migrate in the t-th TS, and 0 otherwise.
E_m^t	The boolean variable that equals 1 if MU m has completed its migration in the t-th TS, and 0 otherwise.
r_m^t	The size of data to be transferred for MU m in t-th TS.
z_m^i	The boolean variable that equals 1 if the migration of MU m selects duration $i \in I$, and 0 otherwise.
N_{v_s}	The last TS used by rack v_s to migrate the VMs on it, *i.e.*, the end time of all the VM migrations from rack v_s.

it multiplies z_m^i with x_m^t and y_m^t, it can be easily linearized with standard techniques since z_m^i is a binary variable.

$$\begin{cases} r_m^t \geq c_m, & t = 1, \\ r_m^t \geq 0, & t \in [2, T]. \end{cases} \tag{12}$$

$$r_m^t = r_m^{t-1} - (x_m^{t-1} + y_m^{t-1}) \cdot \Delta t, \quad \forall m,\ \forall t \in [2, T]. \tag{13}$$

Eq. (12)-(13) ensure that the value of r_m^t is correctly selected.

$$1 - E_m^t \leq r_m^t \leq c_m \cdot (1 - E_m^t), \quad \forall m,\ t. \tag{14}$$

Eq. (14) states that each MU m has been migrated successfully when the remaining data to be transferred becomes 0.

$$S_m^t \geq \sum_{i \in I} z_m^i \cdot f_i^t + E_m^t, \quad \forall m,\ t. \tag{15}$$

Eq. (15) ensures that each MU m can only be migrated in its selected duration, and the start time of its migration should be earlier than the end time.

$$\sum_{i \in I} z_m^i = 1, \quad \forall m. \tag{16}$$

Eq. (16) ensures that each MU m can select one and only one duration for its migration.

$$\Delta t \cdot \sum_{t \in [1,T]} \sum_{i \in I} (x_m^t + y_m^t) \cdot z_m^i \cdot f_i^t \geq c_m, \quad \forall m. \tag{17}$$

Eq. (17) ensures that each MU m should be migrated within its selected duration. Similar to Eq. (11), this constraint is also nonlinear and can be linearized with standard techniques.

$$\begin{cases} b_{v_s}^{\text{out},t} = b_{v_s}^{\text{out}} + \sum_{m \in \mathbb{M}} E_m^t \cdot (\tilde{x}_m \cdot s_m^{v_s} - \hat{x}_m \cdot d_m^{v_s}) \\ \qquad + \sum_{m \in \mathbb{M}} E_m^t \cdot (\hat{y}_m \cdot \tilde{s}_m^{v_s} - \tilde{y}_m \cdot \tilde{d}_m^{v_s}), \\ b_{v_s}^{\text{in},t} = b_{v_s}^{\text{in}} + \sum_{m \in \mathbb{M}} E_m^t \cdot (\tilde{x}_m \cdot s_m^{v_s} - \hat{x}_m \cdot d_m^{v_s}) \\ \qquad + \sum_{m \in \mathbb{M}} E_m^t \cdot (\hat{y}_m \cdot \tilde{s}_m^{v_s} - \tilde{y}_m \cdot \tilde{d}_m^{v_s}), \end{cases} \quad \forall v_s,\ t. \tag{18}$$

Eq. (18) ensures that when MU m has been migrated in the t-th TS, the available EPS bandwidth capacity from/to a related rack v_s (the source/destination racks and the ones which connect with them through the OXC) is updated correctly.

$$\begin{aligned} b_{(v_s,u_s)}^t = & \sum_{m \in \mathbb{M}} E_m^t \cdot [\tilde{y}_m \cdot (s_m^{v_s} \cdot \tilde{d}_m^{u_s} + s_m^{u_s} \cdot \tilde{d}_m^{v_s})] \\ & - \sum_{m \in \mathbb{M}} E_m^t \cdot [\hat{y}_m \cdot (d_m^{u_s} \cdot \tilde{s}_m^{v_s} + d_m^{v_s} \cdot \tilde{s}_m^{u_s})] + b_{(v_s,u_s)}, \\ & \{v_s, u_s \in V_s : v_s \neq u_s\},\ \forall t. \end{aligned} \tag{19}$$

Eq. (19) ensures that when MU m has been migrated in the t-th TS, the available OCS bandwidth capacity between the racks, *i.e.*, the source or destination rack and that one connects with it through the OXC, is updated correctly.

$$it_{v_s}^t = it_{v_s} + \sum_{m \in \mathbb{M}} c_m \cdot (E_m^t \cdot s_m^{v_s} - S_m^t \cdot d_m^{v_s}),\ it_{v_s}^t \geq 0,\ \forall v_s,\ t. \tag{20}$$

Eq. (20) ensures that the resources on the source and destination racks of MU m are updated correctly throughout the process. Note that, when a MU migration starts, we reserve enough resources (*i.e.*, c_m) on its destination rack immediately to ensure that the migration can be accomplished successfully. Hence, when we are migrating an MU, it consumes resources on both the source and destination racks simultaneously.

V. MULTI-SHOT PARALLEL VNT RECONFIGURATIONS

It will be time-consuming to solve the MILP model formulated above. Meanwhile, to fully explore the flexibility of OCS for the scheduling of VM migrations, we might need a multi-shot approach that can schedule parallel VM migrations together with OXC reconfigurations in batches. However, the optimization of the multi-shot parallel VM migrations is $\mathcal{NP}$-hard. This is because by restricting that VM migrations can only use the OCS bandwidth in the HOE-DCN, we can reduce the optimization to a general case of Coflow scheduling with OCS, which is known to be $\mathcal{NP}$-hard [49]. Therefore, it might not be feasible to design a polynomial time exact algorithm for it. To this end, this section discusses our considerations for designing time-efficient heuristics to solve the problem.

A. Proper Procedure for VM Migrations

We first need to finalize the procedure of the VM migration algorithm. Although the one-shot approach (*i.e.*, migrate all the VMs in parallel based on the current status of the HOE-DCN) is intuitive, it might not always lead to the shortest migration time, due to the bandwidth competition among VM migrations [25]. We addressed this issue in [25], and designed a multi-shot approach to schedule parallel VM migrations together with OXC reconfigurations in batches, such that the bandwidth competition at source racks can be relieved. We still assumed that in each batch, VM migrations are scheduled before OXC reconfiguration. Nevertheless, the multi-shot approach does not consider the bandwidth competition at destination racks.

Fig. 3 provides an example to explain the effect of the bandwidth competition at destination racks. We have three MUs to migrate (*MUs* 1-3), their memory sizes are 1, 1 and 3 GB, respectively, the bandwidth usages of the normal service traffic of their VMs are 2, 1 and 2 Gbps, respectively, and the available input/output bandwidths on *ToR Switches* 1-3 are 5, 2

and 4 Gbps, respectively. In this case, the multi-shot approach, which tries to migrate the VMs in multiple batches, while in each batch, a few VMs are migrated in parallel [25], will first migrate *MU* 2, and then handle *MUs* 1 and 3, as shown in Fig. 3(a). The overall migration time is $\left(\frac{1}{2}+\frac{3}{4}\right)\cdot 8 = 10$ seconds, and the bandwidth competition on *ToR Switch* 1 actually prolongs the migration time of *MU* 3. On the other hand, if we use the one-shot approach, the MUs can be migrated as illustrated in Fig. 3(b), and the overall migration time is $\left(\frac{1+1}{2}\right)\cdot 8 = 8$ seconds. Hence, for this particular example, the multi-shot approach even performs worse than the one-shot approach. This is because the multi-shot approach in [25] does not properly address the dependencies among MUs due to the bandwidth competition at source and destination racks.

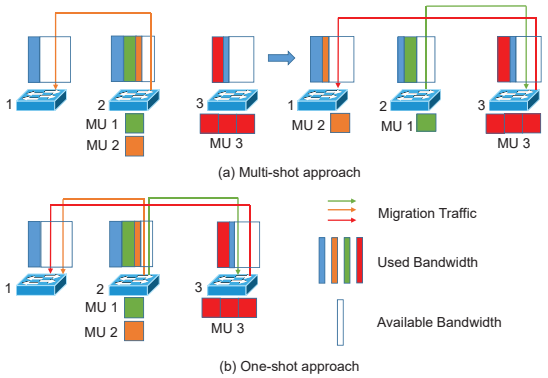


Fig. 3. Example on effect of the bandwidth competition at destination racks.

B. Resolving Dependencies among MUs

As the one-shot approach can hardly resolve the dependencies among MUs, we still need to design algorithms based on the multi-shot approach, but take the bandwidth competition at source and destination racks into account this time. To realize the multi-shot approach with bandwidth competition avoidance (Multi-Shot-BCA), we define the concept of group MU.

Definition 2. *A **group MU (GMU)** refers to those MUs whose destination racks are the same and the bottleneck of their migrations is at the ToR switch of the destination rack.*

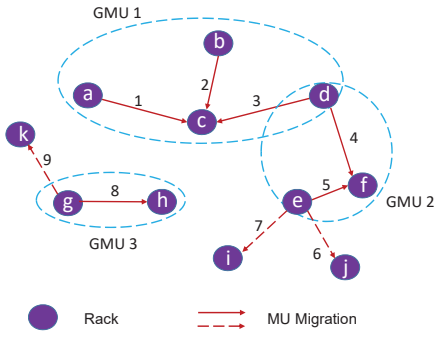


Fig. 4. Example on obtaining GMUs with an AG.

The GMUs can be obtained by generating an auxiliary graph (AG) according to the status of the HOE-DCN. In the AG, each node denotes a rack, and two nodes are connected with a directed link if there is an MU to be migrated between the corresponding racks. We use a solid directed link to represent an MU if the bottleneck of its migration is at the destination rack, and use a dash directed link otherwise. Fig. 4 shows an example on the AG for obtaining GMUs, where the smallest GMU is *GMU* 3 and it only contains *MU* 8, while the largest one is *GMU* 1 that includes *MUs* 1-3. Note that, *MUs* 6, 7 and 9 are not included in any GMU, because the bottlenecks of their migrations are at the source racks.

With GMUs, we can schedule VM migrations in a way that the bandwidth competition can be alleviated, *i.e.*, maximizing the available bandwidth for subsequent VM migrations.

1) Allocate-with-weight (AWW): To allocate bandwidth to MUs in a GMU, we propose the allocate-with-weight (AWW) scheme. Specifically, the AWW scheme allocates bandwidth to each MU m in a GMU G according to the MU's weight w_m, which is calculated based on the available bandwidth capacity of its destination rack and its remaining data to be transferred,

$$w_m = \frac{r_m^t}{\sum\limits_{m' \in G} r_{m'}^t}, \tag{21}$$

where r_m^t denotes the remaining data to be transferred of MU m at time t. Then, all the MUs of a GMU can finish their migrations concurrently, instead of some MUs using too much bandwidth at the ToR switch of the destination rack and thus slowing down the migrations of others.

2) Cost-effectiveness-first (CEF): Meanwhile, the bandwidth competition at source racks should also be addressed, and thus we design the cost-effectiveness-first (CEF) scheme to select the most cost-effective MUs to migrate at first, such that the bandwidth competition at source racks can be relieved. The cost-effectiveness of MU m is defined as

$$g_m = \frac{b_m^t}{r_m^t}, \tag{22}$$

where b_m^t denotes the total bandwidth usage of the service traffic of MU m at time t. The rationale behind Eq. (22) is that if we first migrate the MU that uses the most bandwidth for its service traffic and has the least remaining data to be transferred, we can maximize the available bandwidth for subsequent MU migrations from the same source rack.

VI. Heuristic Algorithm Design

In this section, we design heuristic algorithms for scheduling VNT reconfigurations in an HOE-DCN based on GMUs and the AWW and CEF schemes. Specifically, the heuristic algorithms are designed in two scenarios: 1) all the VM migrations are scheduled before the OXC reconfiguration, and 2) the OXC reconfiguration is divided into a few progressive steps and VM migrations are scheduled together with the steps. Hence, the first scenario is the Multi-Shot-BCA using separate scheduling (Multi-Shot-BCA-S), while the second one is the Multi-Shot-BCA using joint scheduling (Multi-Shot-BCA-J). Both scenarios use the similar procedure, which includes the phases of preprocessing, bandwidth allocation and MU migration, and OXC reconfiguration and VL remapping.

A. Preprocessing

Algorithm 1 explains the procedure of preprocessing. First of all, *Lines* 1 and 2 obtain the MUs and GMUs according to the original and new VNE schemes of VNTs (*i.e.*, $\mathcal{F}$ and $\mathcal{F}'$, respectively). Here, all the obtained GMUs are stored in set $\mathbb{G}_1$, while the remaining MUs in $\mathbb{M}$ are put in set $\mathbb{G}_2$. This distinguishes whether the bottlenecks of the MU migrations are at destination or source racks. Then, we calculate the estimated migration time of each GMU in $\mathbb{G}_1$ and the cost-effectiveness of each MU in $\mathbb{G}_2$, and sort the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ according to them, respectively (*Lines* 3-10). Next, in *Lines* 11-13, we get the set of racks $\mathbb{R}$ that are both source racks of the MUs in $\mathbb{G}_2$ and destination racks of the GMUs in $\mathbb{G}_1$, and set the time-dependence of the related MUs accordingly. Specifically, if $v_s \in \mathbb{R}$ is the source rack of an MU $m \in \mathbb{G}_2$ and it is also the destination rack of a GMU $G \in \mathbb{G}_1$, we set G to be depended on m, which means that the migration of m should be scheduled before that of G. By doing so, we can maximize the available bandwidth for VM migrations. The time complexity of *Algorithm* 1 is $O(|\mathbb{G}_1| + |\mathbb{G}_2|)$.

Algorithm 1: Preprocessing

```
1  get MUs based on F and F' to store in set M;
2  get GMUs with M to put in set G_1 and have G_2 = M \ G_1;
3  for each GMU G ∈ G_1 do
4  |  hypothetically migrate all the MUs in G in parallel in
   |  the HOE-DCN to get the estimated migration time τ;
5  end
6  sort GMUs in G_1 in descending order of estimated
   migration time;
7  for each MU m ∈ G_2 do
8  |  calculate the cost-effectiveness of m with Eq. (22);
9  end
10 sort MUs in G_2 in descending order of cost-effectiveness;
11 store source racks of the MUs in G_2 in set SG_2, and store
   destination racks of the GMUs in G_1 in set DG_1;
12 R = SG_2 ∩ DG_1;
13 set time-dependence of MU migrations based on R;
14 G'_1 = G'_2 = ∅;
```

B. Bandwidth Allocation and MU Migration

The procedure of allocating bandwidth and migrating MUs in TS' is shown in *Algorithm* 2. Here, we record the GMUs and MUs that are currently being migrated in sets $\mathbb{G}'_1$ and $\mathbb{G}'_2$, respectively, which have been initialized as empty sets in the preprocessing in *Algorithm* 1. *Lines* 1-8 first maintain the bandwidth allocations to the GMUs and MUs that are being migrated. For instance, if a previous OXC reconfiguration has made the available OCS bandwidth for migrating an MU to be 0, we will allocate the minimum required bandwidth in the EPS part to ensure a continuous migration (*Lines* 3-5). Then, the for-loop covering *Lines* 9-19 selects new GMUs from $\mathbb{G}_1$ to migrate. Specifically, for a GMU G, we first allocate bandwidth to the MUs in it with AWW (*Lines* 10-14), and then determine whether the MUs can be migrated (*Line* 15). If yes, we finalize the bandwidth allocations to the MUs, insert the GMU in $\mathbb{G}'_1$, and remove it from $\mathbb{G}_1$ (*Lines* 16-17). Next, *Lines* 20-34 try to find new MUs in $\mathbb{G}_2$ to migrate with CEF. Similarly, if an MU can start its migration, we finalize the bandwidth allocated to it, insert the MU in $\mathbb{G}'_2$, and remove it from $\mathbb{G}_2$ (*Lines* 29-32). Finally, we use *Lines* 35-41 to proceed the VM migrations for Δt according to the obtained bandwidth allocations and update the network status accordingly. The time complexity of *Algorithm* 2 is $O(|\mathbb{G}'_1 \cup \mathbb{G}'_2| + |\mathbb{G}_1| \cdot |G| + |\mathbb{G}_2| + |\mathbb{M}|)$.

C. Overall Procedure

Algorithm 3 shows the overall procedure of the proposed scheduling algorithm, *i.e.*, Multi-Shot-BCA. Here, *Lines* 5-9 are introduced to support Multi-Shot-BCA-J, which schedules VM migrations together with OXC reconfiguration steps. Specifically, to limit the operation complexity, we predefine the largest number of OXC reconfiguration steps as N, and leverage *Algorithm* 4 to check whether and how an OXC reconfiguration should be invoked based on the current status of the HOE-DCN (*Lines* 6-8). If the OXC reconfiguration defined in $\mathcal{F}'$ has not been accomplished after N steps, *Line* 13 finishes the remaining reconfiguration. The effect of N on the performance of VNT reconfigurations will be investigated with simulations in Section VII. It can be seen that if we set $N = 0$, *Algorithm* 3 becomes Multi-Shot-BCA-S, which schedules all the VM migrations before OXC reconfiguration.

D. OXC Reconfiguration

As we have discussed in Section III-B, OXC reconfiguration changes the inter-rack topology of an HOE-DCN. Hence, if we schedule OXC reconfiguration steps with VM migrations, the bandwidth competition on bottleneck racks might be further relieved. *Algorithm* 4 shows how to schedule an OXC reconfiguration for this purpose. *Lines* 1-2 are for the initialization, where $flag$ is introduced to indicate whether an OXC reconfiguration is invoked in the subsequent procedure, and set $\mathbb{P}$ stores all the OXC ports that should be reconfigured to accomplish the VNT reconfigurations in the HOE-DCN. Then, we check all the GMUs and MUs in $\mathbb{G}_1 \cup \mathbb{G}'_1$ and $\mathbb{G}_2 \cup \mathbb{G}'_2$ (*i.e.*, those that have not completed their migrations yet), respectively, to find those whose migrations will be prolonged the most due to the bottlenecks at their destination or source rack, and store the OXC's ports to the GMU's destination rack and the MU's source rack in set $\mathbb{O}$ (*Lines* 3-12).

Next, for each port p_i in $\mathbb{O} \cap \mathbb{P}$, we make sure that no MU is currently being migrated through it, get the OXC's ports that MUs can use to be migrated from/to it, and store the obtained ports in set $\mathbb{O}_1$. Then, for each port p_j in $\mathbb{O}_1 \cap \mathbb{P}$, the for-loop that covers *Lines* 17-25 checks it against port p_i to see whether reconfiguring the two port pairs related to them can bring bandwidth gain to the subsequent VM migrations. Specifically, the bandwidth gain brought to the rack connecting to port p_i by reconfiguring the four port p_i, p_j, $\tilde{p}_i$ and $\tilde{p}_j$ is

$$b_j = \sum_{\{m:p_i \leftrightarrow p_j\}} r_m^t + \Delta b_1 - \Delta b_2, \tag{23}$$

where the first term on the right side is the remaining data that could be migrated between p_i and p_j through the OXC

Algorithm 2: Allocate Bandwidth to MU Migrations

for *each MU* $m \in \mathbb{G}'_1 \cup \mathbb{G}'_2$ **do**
  get source and destination racks of m as v_s and u_s;
  if $b^t_{(v_s,u_s)} = 0$ **then**
    allocate the minimum bandwidth D_m in the EPS part and update network status;
  end
end
allocate bandwidth to each GMU $G \in \mathbb{G}'_1$ with AWW and update network status;
allocate bandwidth to each MU $m \in \mathbb{G}'_2$ with CEF and update network status;
for *each GMU* $G \in \mathbb{G}_1$ **do**
  get destination rack of G as u_s;
  for *each MU* $m \in G$ **do**
    get source rack of m as v_s;
    get weight w_m with Eq. (21) and allocate w_m ratio of available EPS/OCS bandwidth to m;
  end
  if *(the memory in rack* u_s *is enough) AND (all the MUs that GMU* G *depends on have been migrated)* **then**
    finalize the bandwidth allocation to GMU G and update network status;
    insert G in $\mathbb{G}'_1$ and remove it from $\mathbb{G}_1$;
  end
end
update $\{g_m\}$ and sort MUs in $\mathbb{G}_2$ accordingly;
for *each MU* $m \in \mathbb{G}_2$ **do**
  get source and destination racks of m as v_s and u_s;
  if $it^t_{u_s} - c_m > 0$ **then**
    if $b^t_{(v_s,u_s)} > 0$ **then**
      $b^t_m = \min\left(\frac{r^t_m}{\Delta t}, b^t_{(v_s,u_s)}\right)$;
    else
      $b^t_m = \min\left(\frac{r^t_m}{\Delta t}, b^{\text{in},t}_{u_s}, b^{\text{out},t}_{v_s}\right)$;
    end
    if $b^t_m \geq D_m$ **then**
      finalize bandwidth allocation to MU m as b^t_m in EPS/OCS parts and update network status;
      insert m in $\mathbb{G}'_2$ and remove it from $\mathbb{G}_2$;
    end
  end
end
for *each MU* $m \in \mathbb{M}$ **do**
  migrate m according to the finalized bandwidth allocation for Δt, and update the remaining data r^t_m;
  if $r^t_m = 0$ **then**
    remove m from $\mathbb{M}$, $\mathbb{G}'_1$ and $\mathbb{G}'_2$;
    update network status and remap related VLs;
  end
end

after reconfiguring the ports, Δb_1 refers to the total bandwidth of the VLs that should be remapped onto the new optical connection between p_i and p_j, and Δb_2 is the total bandwidth of the VLs that are currently mapped on the optical connection between p_i and $\tilde{p}_i$. In *Lines* 26-30, we obtain the maximum bandwidth gain b_{max}, and check whether it is positive. If yes, an OXC reconfiguration should be invoked (*Line* 28). Finally, the algorithm returns the value of $flag$ in *Line* 33. The time complexity of *Algorithm* 4 is $O(|\mathbb{M}|+|V_s|+|\mathbb{O}\cap\mathbb{P}|\cdot|\mathbb{O}_1\cap\mathbb{P}|)$.

Algorithm 3: Overall Procedure of Multi-Shot-BCA

invoke preprocessing with *Algorithm* 1;
$t = 0$, $n = 0$;
while $\mathbb{M} \neq \emptyset$ **do**
  apply *Algorithm* 2 to allocate bandwidth and proceed the VM migrations for Δt;
  if $n < N$ **then**
    if Algorithm *4 reconfigures the OXC* **then**
      $n = n + 1$;
    end
  end
  $t = t + 1$;
end
$T = t \cdot \Delta t$;
reconfigure the OXC and related VLs according to $\mathcal{F}'$, and update the status of HOE-DCN;

VII. Performance Evaluations

In this section, we perform numerical simulations to evaluate the performance of our proposed algorithms.

A. Simulation Setup

Our simulations assume that the EPS part of the HOE-DCN uses a k-ray fat-tree topology [46], where there are $\frac{k^2}{2}$ racks and they belong to a large PoD. Each ToR switch has $\frac{k}{2}$ Ethernet ports to connect to the EPS part of the inter-rack network, and it also equips an optical port to the OXC. In the simulations, we consider the $\{6, 28, 46\}$-ray fat-tree topologies as the small-, medium- and large-scale cases, *i.e.*, there are $\{18,\ 392,\ 1,058\}$ racks/ToR switches in the HOE-DCN, respectively. Here, the largest case assumes that the OXC can interconnect $1,058$ ToR switches, considering the upper-limit on the port-count of a real-world OXC [50] and the modular design of a practical network control and management (NC&M) system [43].

We set the bandwidth capacity of each Ethernet port on a ToR switch as 10 Gbps, while that of an optical port is 100 Gbps. Meanwhile, the memory capacity of each rack is set as $512 \cdot k$ GB for all the cases. We use the Poisson traffic model to generate dynamic VNTs with random topologies [21, 51], *i.e.*, each VNT is set up and torn down on-the-fly in the simulations. In each VNT, the number of VMs and the bandwidth demand of each VL are randomly selected, and each pair of VMs are connected with a probability of 0.5. Hence, the average number of VLs in a VNT is $\frac{n(n-1)}{4}$, where n is the average number of VMs in the VNT. The memory size of each VM is uniformly distributed within the range defined by Amazon EC2 VM instances [52], and other parameters of the VMs are either derived from realistic cases or based on the observations in our previous experiments [6, 16]. Table III summarizes the key parameters used in the simulations.

As our proposed algorithms optimize the procedure of VNT reconfigurations from the original VNE $\mathcal{F}$ to the new one $\mathcal{F}'$, they do not restrict for what purpose $\mathcal{F}'$ was calculated. Without loss of generality, the simulations choose the load-balancing scenario considered in [21] to calculate the new VNE schemes, and use them as the inputs to our scheduling

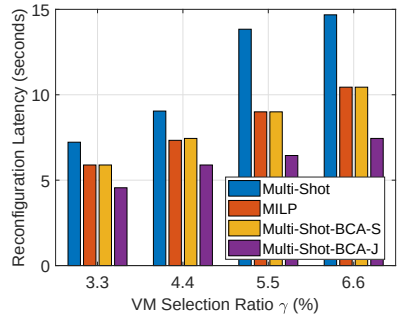
Algorithm 4: Obtain OXC Reconfiguration Scheme

$\mathbb{O} = \emptyset$, $flag = 0$;
get reconfiguration port set $\mathbb{P}$ by checking $\mathcal{F}$ and $\mathcal{F}'$;
for *each GMUs* $G_j \in \mathbb{G}_1 \cup \mathbb{G}_1'$ **do**
  hypothetically migrate all the MUs in G_j in parallel in the HOE-DCN to get the estimated migration time τ_j;
end
$G^* = \underset{G_j \in \mathbb{G}_1 \cup \mathbb{G}_1'}{\operatorname{argmax}} (\tau_j)$;
get the destination rack of GMU G^* as u_s^*;
for *each rack* $v_s \in V_s$ *that has MUs in* $\mathbb{G}_2 \cup \mathbb{G}_2'$ **do**
  hypothetically migrate all the MUs on v_s in parallel in the HOE-DCN to get the estimated migration time τ_{v_s};
end
$v_s^* = \underset{v_s \in V_s}{\operatorname{argmax}} (\tau_{v_s})$;
get the OXC's ports that connect to u_s^* and v_s^*, and store them in set $\mathbb{O}$;
for *each port* $p_i \in \mathbb{O} \cap \mathbb{P}$ **do**
  if *no MU migrating from/to port* p_i *currently* **then**
    $\mathbb{O}_1 = \emptyset$;
    get the OXC's ports that MUs can use to be migrated from/to port p_i, and store them in set $\mathbb{O}_1$;
    for *each port* $p_j \in \mathbb{O}_1 \cap \mathbb{P}$ **do**
      get the OXC's ports currently connecting with p_i and p_j to denote as $\tilde{p}_i$ and $\tilde{p}_j$, respectively;
      hypothetically reconfigure the OXC to connect p_i with p_j and $\tilde{p}_i$ with $\tilde{p}_j$;
      get the bandwidth gain b_j of the rack that connects to p_i with Eq. (23);
      update the available bandwidth of the racks that connect to p_i, p_j, $\tilde{p}_i$ and $\tilde{p}_j$;
      if *the HOE-DCN can support currently-active VM migrations after reconfiguring the four ports on the OXC* **then**
        record b_j together with the OXC reconfiguration scheme;
      end
    end
    $b_{max} = \underset{p_j \in \mathbb{O}_1 \cap \mathbb{P}}{\operatorname{argmax}} (b_j)$;
    if $b_{max} > 0$ **then**
      reconfigure the OXC according to the scheme associated with b_{max}, and remove p_i from $\mathbb{P}$;
      $flag = 1$;
    end
  end
end
return($flag$);

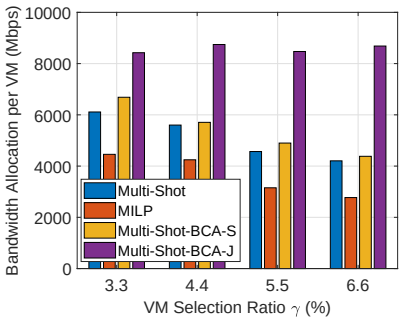
algorithms. Specifically, in each simulation, we first use the VNE algorithm in [51] to provision newly-generated VNTs ($\mathcal{F}$) and release the resources occupied by the expired ones, and then leverage the algorithms developed in [19, 21] to select VMs to migrate and calculate the new VNE and OXC schemes for the related VNTs ($\mathcal{F}'$). Note that, the algorithms in [19, 21] use a parameter γ, which defines the ratio for selecting the VMs to migrate for load-balancing. Next, when both $\mathcal{F}$ and $\mathcal{F}'$ have been determined, we apply the algorithms designed in this work to optimize the procedure of VNT reconfigurations for the shortest overall reconfiguration latency. The simulations compare four scheduling algorithms, *i.e.*, the Multi-Shot developed in [25], the MILP, Multi-Shot-BCA-S and Multi-Shot-BCA-J. To ensure sufficient statistical accuracy, we average the results from 10 independent runs to obtain each data point.

TABLE III
SIMULATION PARAMETERS

	Simulation Scale		
	Small	Medium	Large
# of racks	18	392	1,058
# of servers	54	5,488	24,334
Memory capacity of a rack (GB)	3,072	14,336	23,552
VMs in a VNT	[2, 16]	[2, 64]	[2, 100]
Bandwidth usage of a VL (Mbps)	(0, 60]	(0, 80]	(0, 100]
Memory size of a VM (GB)	[1, 2]	[2, 25]	[2, 50]



(a) Overall VNT reconfiguration latency



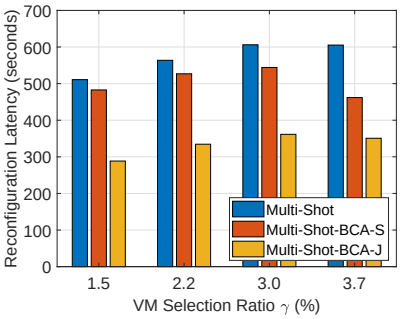
(b) Average bandwidth allocation per VM migration

Fig. 5. Simulation results of the small-scale case (18 racks in total).

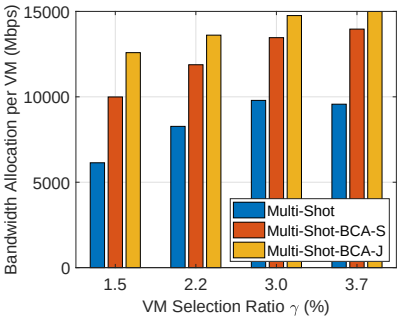
B. Small-Scale HOE-DCN

Due to the time complexity of the MILP, we first consider small-scale case that uses the 6-ray fat-tree as the EPS part. Here, we have $\Delta t = 1$ second and $D_m = 200$ Mbps, and set the largest number of OXC reconfiguration steps as $N = 5$.

1) Overall Reconfiguration Latency: Fig. 5(a) shows the results on overall VNT reconfiguration latency. We observe that the algorithms developed in this work always achieve shorter reconfiguration latencies than Multi-Shot, which confirms that the bandwidth competition avoidance (BCA) scheme proposed in this work can relieve the bandwidth competition among VM migrations more effectively. It is also interesting to observe that Multi-Shot-BCA-J can even achieve shorter reconfiguration latency than the MILP. This is because Multi-Shot-BCA-J removes the restriction that all the VM migrations should be scheduled before the OXC reconfiguration. In other



(a) Overall VNT reconfiguration latency



(b) Average bandwidth allocation per VM migration

Fig. 6. Simulation results of the medium-scale case (392 racks in total).

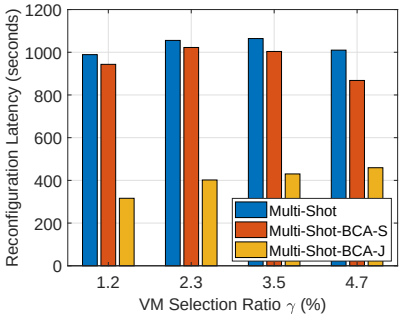
words, the results in Fig. 5(a) verify the benefit of scheduling VM migrations together with OXC reconfiguration steps.

2) Bandwidth Allocation to VM Migrations: The results on average bandwidth allocation to each VM migration are plotted in Fig. 5(b). It is interesting to notice that the average bandwidth allocation from Multi-Shot is not the smallest among all the algorithms. This actually explains why Multi-Shot provides the longest overall VNT reconfiguration latency. Specifically, Multi-Shot allocates too much bandwidth to certain VM migrations and does not address the bandwidth competition among VM migrations properly. Hence, although one VM might be migrated quickly, the overall migration time gets prolonged because certain VMs cannot start their migrations before others have been migrated (*i.e.*, the scheduling of VM migrations is sub-optimal). The average bandwidth allocations increase from the MILP to Multi-Shot-BCA-S and Multi-Shot-BCA-J, which is because Multi-Shot-BCA-J can reconfigure the OXC during VNT reconfiguration to squeeze more bandwidth for VM migrations.

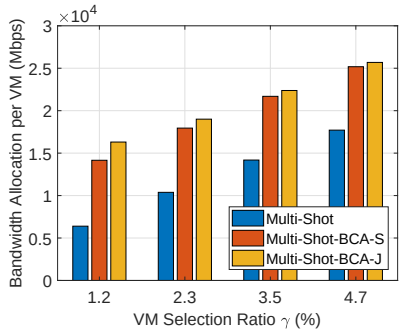
3) Time Complexity: The running time of the algorithms is listed in Table IV. As expected, the MILP runs the slowest and its running time increases fast with the scale of the problem (*i.e.*, the VM selection ratio γ). Meanwhile, it is promising to see that both Multi-Shot-BCA-S and Multi-Shot-BCA-J run faster than Multi-Shot. This is because Multi-Shot needs to solve a lightweight linear programming (LP) to finalize the scheduling scheme. To this end, we can conclude that compared with Multi-Shot, Multi-Shot-BCA-S and Multi-Shot-BCA-J not only achieve shorter overall VNT reconfiguration time, but also use shorter running time.

TABLE IV
AVERAGE RUNNING TIME (SECONDS)

6-ray Fat-tree				
γ	3.3%	4.4%	5.5%	6.6%
Multi-Shot	0.72	0.79	0.94	1.44
MILP	65.69	80.14	132.97	3001.03
Multi-Shot-BCA-S	0.03	0.03	0.02	0.02
Multi-Shot-BCA-J	0.10	0.14	0.09	0.10



(a) Overall VNT reconfiguration latency



(b) Average bandwidth allocation per VM migration

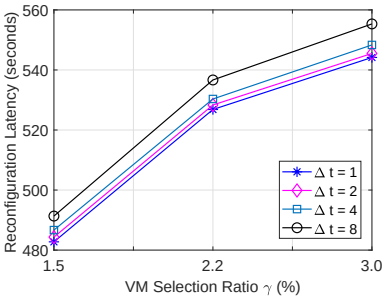
Fig. 7. Simulation results of the large-scale case (1,058 racks in total).

C. Medium-Scale and Large-Scale HOE-DCNs

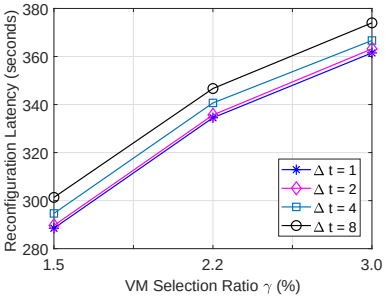
Next, we consider HOE-DCNs whose scales are relatively large, *i.e.*, with 28- and 46-ray fat-tree topologies. This time, we still have $\Delta t = 1$ second, and the values of D_m and N are not changed. Due to the time complexity of the MILP, we only simulate Multi-Shot, Multi-Shot-BCA-S, and Multi-Shot-BCA-J. Figs. 6 and 7 show the results on overall VNT reconfiguration latency and average bandwidth allocation to each VM migration, respectively, which follow the similar trends of those in Fig. 5. Note that, Multi-Shot-BCA-J not only provides the shortest overall VNT reconfiguration latency among the three algorithms, but also achieves larger advantages over the other two algorithms when the scale of the HOE-DCN increases. This further verifies its scalability.

We then investigate how the value of TS duration Δt affects the performances of Multi-Shot-BCA-S and Multi-Shot-BCA-J. Specifically, we consider the HOE-DCN with the 28-ray fat-tree topology, choose Δt from $\{1, 2, 4, 8\}$ seconds, and run the simulations. Fig. 8 shows the results on overall VNT reconfiguration latency. We observe that for both algorithms, reducing Δt can shorten the overall reconfiguration latency.

This is because a shorter Δt ensures that bandwidth allocations to VM migrations can be updated more timely. Meanwhile, it can be seen that for both algorithms, their performances with $\Delta t = 2$ seconds are similar to that with $\Delta t = 1$ second. Note that, although reducing Δt leads to shorter reconfiguration latency, it also increases the complexity of VNT reconfiguration scheduling. Hence, we should adjust the value of Δt empirically to balance the tradeoff between overall reconfiguration latency and operational complexity.



(a) Multi-Shot-BCA-S



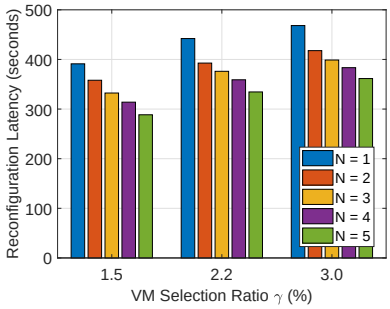
(b) Multi-Shot-BCA-J

Fig. 8. Effect of Δt on overall reconfiguration latency (28-ray fat-tree).

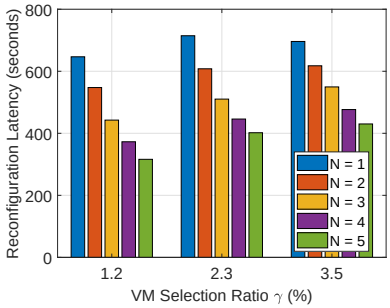
Finally, we analyze the effect of the largest number of OXC reconfiguration steps (N) on the performance of Multi-Shot-BCA-J. Here, we still have $\Delta t = 1$ second, and change $N \in [1, 5]$. Fig. 9 illustrates the results on overall reconfiguration latency, which indicates that the overall reconfiguration latency decreases with N. This is because a larger N provides us more flexibility to reconfigure the OCS part of an HOE-DCN to expedite VM migrations. Meanwhile, we also notice that with the increase of N, the performance gain on overall reconfiguration latency shrinks. Note that, the values of Δt and N affect the tradeoff between operational complexity and algorithm performance significantly, but they are set empirically in the simulations. Therefore, for the real deployment of our proposed algorithm, our future work will optimize these two parameters according to the status of a practical HOE-DCN.

VIII. Conclusion

In this work, we studied how to optimize the procedure of VNT reconfigurations in an HOE-DCN by scheduling VM migrations and OXC reconfiguration steps properly. We first assumed that all the VM migrations should be scheduled before the OXC reconfiguration to avoid unnecessary service interruptions during VNT reconfigurations, and formulated an



(a) HOE-DCN with 28-ray fat-tree



(b) HOE-DCN with 46-ray fat-tree

Fig. 9. Effect of N on overall reconfiguration latency (Multi-Shot-BCA-J).

MILP model based on MU to solve the scheduling problem exactly. Then, with the same assumption, we proposed a time-efficient heuristic to schedule parallel VM migrations in batches such that the bandwidth competition can be significantly relieved. Finally, we removed the restriction that the OXC reconfiguration should be accomplished in one step, divided it into a few progressive steps, and designed a joint scheduling algorithm to arrange VM migrations together with the OXC reconfiguration steps. Extensive simulations were conducted to evaluate our scheduling algorithms, and the results suggested that our proposed algorithms with bandwidth competition avoidance performed significantly better than the existing benchmark, and the joint scheduling algorithm can achieve the shortest VNT reconfiguration latency.

Acknowledgments

This work was supported in part by the NSFC projects 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

References

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] Cisco Global Cloud Index: Forecast and Methodology, 2016-2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html.

[3] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient data center networks," *IEEE Commun. Mag.*, vol. 58, pp. 86–92, Jan. 2020.

[4] N. Farrington *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 339–350, Oct. 2010.

[5] G. Wang *et al.*, "c-through: Part-time optics in data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 327–338, Aug. 2010.

[6] Q. Li *et al.*, "Scalable knowledge-defined orchestration for hybrid optical/electrical datacenter networks," *J. Opt. Commun. Netw.*, vol. 12, pp. A113–A122, Feb. 2020.

[7] 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.*, vol. 31, pp. 15–22, Jan. 2013.

[8] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[9] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[10] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*. Apache Software Foundation, 2007.

[11] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[12] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[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.*, vol. 24, pp. 3648–3661, Dec. 2016.

[14] S. Shaw and A. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," in *Proc. of ICCCT 2014*, pp. 87–95, Sept. 2014.

[15] W. Lu *et al.*, "Leveraging predictive analytics to achieve knowledge-defined orchestration in a hybrid optical/electrical DC network: Collaborative forecasting and decision making," in *Proc. of OFC 2018*, pp. 1–3, Mar. 2018.

[16] H. Fang *et al.*, "Predictive analytics based knowledge-defined orchestration in a hybrid optical/electrical datacenter network testbed," *J. Lightw. Technol.*, vol. 37, pp. 4921–4934, Oct. 2019.

[17] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[18] S. Zhao, D. Li, K. Han, and Z. Zhu, "Proactive and hitless vSDN reconfiguration to balance substrate TCAM utilization: From algorithm design to system prototype," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, pp. 647–660, Jun. 2019.

[19] S. Zhao and Z. Zhu, "Network service reconfiguration in hybrid optical/electrical datacenter networks," in *Proc. of ONDM 2020*, pp. 1–6, May 2020.

[20] B. Kong *et al.*, "Demonstration of application-driven network slicing and orchestration in optical/packet domains: On-demand vDC expansion for Hadoop MapReduce optimization," *Opt. Express*, vol. 26, pp. 14 066–14 085, May 2018.

[21] S. Zhao and Z. Zhu, "On virtual network reconfiguration in hybrid optical/electrical datacenter networks," *J. Lightw. Technol.*, vol. 38, pp. 6424–6436, Aug. 2020.

[22] K. Chen, C. Chen, and P. Wang, "Network aware load-balancing via parallel VM migration for data centers," in *Proc. of ICCCN 2014*, pp. 1–8, Aug. 2014.

[23] S. Xiao *et al.*, "Traffic-aware virtual machine migration in topology-adaptive DCN," in *Proc. of ICNP 2016*, pp. 1–10, Oct. 2016.

[24] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. of NSDI 2005*, pp. 273–286, May 2005.

[25] S. Zhao, X. Pan, and Z. Zhu, "On the parallel reconfiguration of virtual networks in hybrid optical/electrical datacenter networks," in *Proc. of GLOBECOM 2020*, pp. 1–6, Dec. 2020.

[26] W. Fang *et al.*, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Mar. 2015.

[27] J. Duan and Y. Yang, "A load balancing and multi-tenancy oriented data center virtualization framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, pp. 2131–2144, Aug. 2017.

[28] N. Chien, N. Son, and H. Dac Loc, "Load balancing algorithm based on estimating finish time of services in cloud computing," in *Proc. of ICACT 2016*, pp. 228–233, Jan. 2016.

[29] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. of CCGRID 2010*, pp. 826–831, May 2010.

[30] M. Bari *et al.*, "CQNCR: Optimal VM migration planning in cloud data centers," in *Proc. of IFIP 2014*, pp. 1–9, Jun. 2014.

[31] X. Yao *et al.*, "VM migration planning in software-defined data center networks," in *Proc. of HPCC 2016*, pp. 765–772, Dec. 2016.

[32] S. Ghorbani and M. Caesar, "Walk the line: Consistent network updates with bandwidth guarantees," in *Proc. of HotSDN 2012*, pp. 67–72, Aug. 2012.

[33] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–98, Apr. 2014.

[34] C. Chen *et al.*, "Demonstrations of efficient online spectrum defragmentation in software-defined elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 4701–4711, Dec. 2014.

[35] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.

[36] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[37] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with Netstitcher," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 74–85, Aug. 2011.

[38] 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.*, vol. 33, pp. 3005–3015, Jul. 2015.

[39] H. Zhang *et al.*, "Guaranteeing deadlines for inter-data center transfers," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 579–595, Feb. 2017.

[40] D. Xie, N. Ding, C. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in *Proc. of ACM SIGCOMM 2012*, pp. 199–210, Aug. 2012.

[41] X. Xie *et al.*, "Evacuate before too late: Distributed backup in inter-DC networks with progressive disasters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, pp. 1058–1074, May 2018.

[42] A. Mestres *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, pp. 2–10, Jul. 2017.

[43] P. Bakopoulos *et al.*, "NEPHELE: An end-to-end scalable and dynamically reconfigurable optical architecture for application-aware SDN cloud data centers," *IEEE Commun. Mag.*, vol. 56, pp. 178–188, Feb. 2018.

[44] X. Xue *et al.*, "ROTOS: A reconfigurable and cost-effective architecture for high-performance optical data center networks," *J. Lightw. Technol.*, vol. 38, pp. 3485–3494, Jun. 2020.

[45] D. Le *et al.*, "AgileDCN: An agile reconfigurable optical data center network architecture," *J. Lightw. Technol.*, vol. 38, pp. 4922–4934, Jun. 2020.

[46] Y. Zhang and N. Ansari, "On architecture design, congestion notification, TCP incast and power consumption in data centers," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 39–64, First Quarter 2012.

[47] A. Choudhary *et al.*, "A critical survey of live virtual machine migration techniques," *J. Cloud Comp.*, vol. 6, pp. 23:1–41, Nov. 2017.

[48] Polatis Series 7000 Software-Defined Optical Circuit Switch. [Online]. Available: https://www.polatis.com/series-7000-384x384-port-/software-controlled-optical-circuit-switch-sdn-enabled.asp.

[49] X. Huang, X. Sun, and T. Ng, "Sunflow: Efficient optical circuit scheduling for Coflows," in *Proc. of CoNEXT 2016*, pp. 297–311, Dec. 2016.

[50] J. Kim *et al.*, "1100×1100 port MEMS-based optical crossconnect with 4-dB maximum loss," *IEEE Photon. Technol. Lett.*, vol. 15, pp. 1537–1539, Oct. 2003.

[51] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[52] Amazon EC2 instances. [Online]. Available: https://aws.amazon.com/cn/ec2/instance-types/.