Scalable Knowledge-defined Orchestration for Hybrid Optical/Electrical Datacenter Networks [Invited]

Qinhezi Li, Hongqiang Fang, Deyun Li, Jianquan Peng, Jiawei Kong, Wei Lu, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—To better provision fast-emerging network applications with various quality-of-service (QoS) demands, datacenter network (DCN) operators need an effective network orchestration scheme that can timely coordinate IT and bandwidth resources for differentiated services. In this work, we consider a hybrid optical/electrical DCN (HOE-DCN), and study how to achieve scalable knowledge-defined network orchestration (KD-NO) for managing the delay-sensitive and delay-tolerant applications in it. For delay-sensitive applications, we leverage a multi-agent scheme to distribute the tasks of placing virtual machines (VMs) in server racks and routing VM traffic in electrical/optical inter-rack clouds to two cooperative deep reinforcement learning (DRL) modules, respectively. Then, we utilize a classic algorithm based module to provision delay-tolerant applications with the residual resources in the HOE-DCN. We design the operation and coordination procedure of the KD-NO system, and build a small HOE-DCN testbed that consists of four server racks to demonstrate its performance experimentally. Experimental results indicate that our KD-NO system can make timely and correct network orchestration decisions, and have better convergence performance compared with the existing benchmark.**

***Index Terms*—Knowledge-defined networking (KDN), Artificial intelligence (AI), Network orchestration, Predictive analytics, Datacenter network (DCN), Deep reinforcement learning (DRL), Store-and-forward**

I. Introduction

NOWADAYS, datacenter networks (DCNs) are facing great challenges from architectural scalability, resource utilization, and management agility [1, 2]. A recent analysis on global cloud traffic has revealed that traffic related to datacenters (DCs) has been increasing with an annual growth rate of 25% since 2016, its total volume will reach 20.6 Zettabytes (ZB) (*i.e.*, 1 ZB $= 10^{21}$ bytes), and $\sim$71% of the traffic is within DCs [3, 4]. Meanwhile, the average utilizations of IT and network resources are still around 25% in many DCNs, and the limited management agility results in relatively long lead time (*e.g.*, in weeks or even months) to deploy new services [5–7]. These issues can be potentially addressed from two perspectives, *i.e.*, architecting hybrid optical/electrical DCNs (HOE-DCNs) [8] and developing artificial intelligence (AI) assisted network orchestration schemes [1].

The idea of HOE-DCN is to add in an optical inter-rack cloud to give the top-of-rack (ToR) switches another option to communicate with, for resolving the bandwidth crunch and energy increase in a DCN. In other words, an HOE-DCN leverages the symbiosis of electrical packet switching (EPS) and optical circuit switching (OCS) to integrate their benefits for handling different types of inter-rack traffic more efficiently. For instance, the mice flows can be forwarded by EPS-based Ethernet switches to ensure sufficient agility, while the elephant flows can be routed through one or more optical cross-connects (OXCs) to avoid causing congestions in the electrical inter-rack cloud [8]. Note that, although the advances on flexible-grid elastic optical networks (EONs) can bring down the granularity of bandwidth allocation in an OCS network to 12.5 GHz or even narrower [9–12], the electrical inter-rack cloud is still needed. This is because even the reduced bandwidth granularity in an EON is still too large for mice flows and the reconfiguration latency of an optical switch could be too long for delay-sensitive traffic.

The introduction of HOE-DCN can potentially address the challenge from architectural scalability, but those from resource utilization and management agility can only be resolved with an effective network orchestration scheme [13, 14]. Here, the network orchestration refers to the mechanism that can jointly optimize the allocations of IT and bandwidth resources to guarantee various quality-of-service (QoS) requirements for active applications [15–18]. Meanwhile, developed based on software-defined networking (SDN) [19–21], knowledge-defined networking (KDN) [22] introduces artificial intelligence (AI) in control plane to realize automatic and agile network control and management (NC&M) and satisfy stringent QoS requirements. This inspires us to consider knowledge-defined network orchestration (KD-NO) for HOE-DCNs [1].

We initially architected the KD-NO in [23], and demonstrated that it could further optimize the well-known "delay-energy" trade-off in an HOE-DCN to push down both sides of the tussle. Then, we improved the KD-NO in [24, 25] by adding a deep reinforcement learning (DRL) module, which can extract high-level knowledge from the telemetry data and the predictions of bandwidth and IT demands, to achieve more effective network orchestration. Nevertheless, despite its superior performance, the enhanced KD-NO in [24, 25] still bears the scalability issues in two-fold. Firstly, it only assigns one DRL module to coordinate both the placement of virtual machines (VMs) in server racks and the routing of VM traffic in electrical/optical inter-rack clouds. This, however, can hardly be a scalable solution considering the volumes of

Q. Li, H. Fang, D. Li, J. Peng, J. Kong, W. Lu, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received July 5, 2019.

VMs and traffic in a typical HOE-DCN. Secondly, it only considers the delay-sensitive applications whose bandwidth and IT resource demands cannot be scheduled in the time domain. Note that, there are also delay-tolerant applications in DCNs, which can be put on hold temporarily to mitigate the contention on traffic and/or IT resources [26–28].

In this work, we try to relieve the aforementioned scalability issues for the KD-NO of HOE-DCNs. Specifically, we further improve the KD-NO designed in [24, 25] from the following two perspectives. We first leverage the multi-agent scheme [29] to divide the centralized DRL module in [24, 25] into two cooperative ones, which handle the placement of VMs in server racks and the routing of VM traffic in electrical/optical inter-rack clouds, respectively, for delay-sensitive applications. Then, we incorporate a classic algorithm based module to schedule the provisioning of delay-tolerant applications with the residual resources in the next provision cycle. In other words, we re-architect the centralized DRL module in [24, 25] as a hybrid integration of multi-agent DRL modules and a classic algorithm based module. We design the operation and coordination procedure of the new KD-NO, and build a small HOE-DCN testbed that consists of four racks to demonstrate its performance experimentally. Experimental results indicate that our KD-NO can also make timely and correct network orchestration decisions, and have better convergence performance compared with the KD-NO in [24, 25].

The rest of the paper is organized as follows. We briefly survey the related work in Section II. Section III describes the design of our KD-NO system, including the system architecture and functional modules. The operation and coordination procedure of the KD-NO is discussed in Section IV. Then, we present the experimental setup and results in Section V. Finally, Section VI summarizes the paper.

II. Related Work

Since the introduction of KDN, people have tried to incorporate various machine learning (ML) techniques in control plane to make NC&M more effective and intelligent. For the current state-of-the-art of ML's applications in optical communications and networks, one is suggested to refer to the surveys in [30, 31]. It is known that ML can be roughly categorized as supervised learning, unsupervised learning, and reinforcement learning [32]. Here, both supervised and unsupervised learning models need to be trained in the offline manner, *i.e.*, they have to be trained with sufficiently-large training data sets before being put into operation. Therefore, the normal use-cases of supervised/unsupervised learning in KDN are traffic/service demand prediction [33–36] and network anomaly detection [37–40]. Note that, our KD-NO needs to make reconfiguration decisions to address the dynamic network environment in an HOE-DCN, and this will rule out both supervised and unsupervised learning because online training would be required.

Since reinforcement learning supports online training [32], it fits perfectly with the design requirement of the KD-NO. Specifically, the KD-NO is the agent, the dynamic HOE-DCN is the environment, the reconfiguration decisions are the actions of the agent, and the HOE-DCN's service quality to active applications is the agent's reward from the environment. To this end, the four basic elements of reinforcement learning, *i.e.*, the agent, environment, actions, and reward, all appear naturally. Meanwhile, a reinforcement learning model can be referred to as DRL, if it utilizes a "deep" neural network that consists of many processing layers to learn the representations of data with multiple levels of abstraction [32]. Hence, DRL enables us to address complicated optimizations in dynamic network environments effectively, especially when the state data is high-dimensional [41, 42]. The advantages of DRL motivate us to design the KD-NO based on it.

Although DRL is the right way to go, we, to the best of our knowledge, have not found any existing studies that leveraged it to optimize the network orchestration in an HOE-DCN. Therefore, we designed our DRL-based KD-NO system in [24, 25], to make smart and timely decisions for improving the matching degree between the HOE-DCN's configuration and the applications running in it. Even though the experimental results in [25] indicated that the enhanced KD-NO could make timely and correct decisions to effectively reduce the job completion time of Hadoop applications, it still suffers from limited scalability. This is because an HOE-DCN is actually a complex system, and if we architect the KD-NO using single-agent DRL, the dimensionality of its search space would grow rapidly with the number and variety of active applications, the number of VMs and servers involved, and the complexity of the inter-rack topology. The enormous search space will make it difficult to train the DRL model, since its convergence rate would be extremely slow.

The aforementioned dilemma could be relieved by architecting the KD-NO using the multi-agent DRL [29]. Specifically, in a multi-agent DRL model, the online optimization is accomplished by multiple agents interacting with the environment and learning simultaneously. Since the optimization is divided into multiple sub-problems, each of which gets addressed by an agent, the scalability would be better. Depending on whether the agents mutually ignore each other, multi-agent DRL can be categorized as independent learning and joint-action learning [43]. Apparently, the KD-NO discussed in this work belongs to the joint-action learning. The applications of multi-agent DRL in communications and networks are recently surveyed in [44], which suggests that the technique has not been applied to solve the network orchestration in DCNs yet.

III. System Design

In this section, we introduce the design of our KD-NO system, and discuss its functional modules in detail.

A. System Architecture of HOE-DCN with KD-NO

Fig. 1 shows the arrangement of the data plane of an HOE-DCN with our KD-NO system. Here, the servers to deploy VMs are organized in racks, each of which has a top-of-rack (ToR) switch to bridge inter-rack communications. The ToR switches are interconnected with electrical/optical inter-rack clouds. Here, the electrical inter-rack cloud follows the conventional hierarchical packet network architecture, while the optical one is essentially based on an optical switch (e.g.,

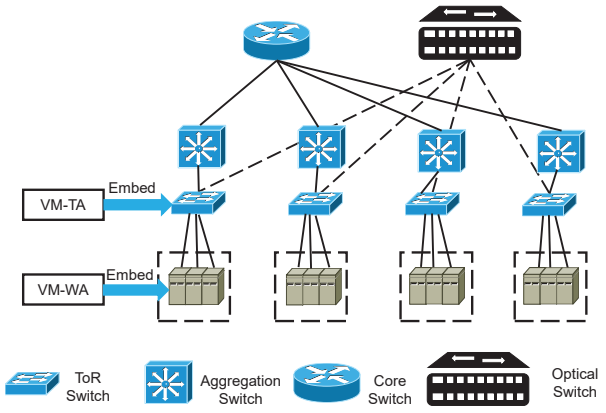


Fig. 1. Data plane of HOE-DCN with KD-NO, VM-TA: VM traffic agent, VM-WA: VM workload agent.

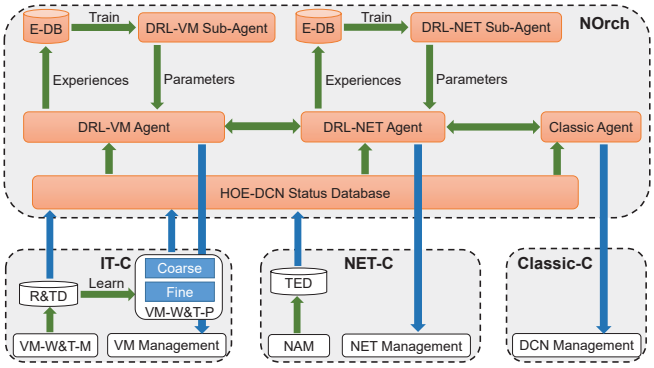


Fig. 2. Control plane of HOE-DCN with KD-NO.

an optical cross-connect (OXC)) that connects to all the optical ports on the ToR switches. Similar to the systems discussed in [24, 25], we embed a VM traffic agent (VM-TA) on each ToR switch to collect the traffic matrix of related VMs, and insert a VM workload agent (VM-WA) on each server to monitor the IT resource consumption of the VMs deployed on it.

The system design of the control plane of our KD-NO system is illustrated in Fig. 2, which includes four major components, *i.e.*, the network orchestrator (NOrch), the IT controller (IT-C), the network controller (NET-C), and the classic controller (Classic-C). Here, the IT-C and NET-C are the controllers to coordinate the IT and bandwidth resources allocated to delay-sensitive applications, respectively, while the Classic-C is the network orchestrator to provision delay-tolerant applications. The design of the NOrch explains our main idea of the "hybrid integration" of multi-agent DRL modules and a classic algorithm based module. With such a design, the NOrch can handle the resource orchestration in the HOE-DCN in a more generic and scalable manner.

B. Functional Modules

We elaborate on the functional modules in Fig. 2 as follows.

1) IT-C: It manages the VMs for delay-sensitive applications, *i.e.*, where to deploy the VMs and when to invoke a VM migration to avoid the contention on IT resources. We allocate a VM workload and traffic monitor (VM-W&T-M) in it to interact with the VM-TAs and VM-WAs embedded in the data plane (as shown in Fig. 1), for collecting telemetry data regarding the HOE-DCN. The received telemetry data gets stored in the IT resource and traffic database (R&TD), which is then fed to the VM workload and traffic predictor (VM-W&T-P) to obtain future workloads and traffic matrixes.

The VM-W&T-P is actually based on a set of long-short-term memory (LSTM) structures [45], each of which is responsible for forecasting the time series of specific workload or traffic. Note that, one enhancement, which we implement in this work over the design in [24, 25], is that two predictors are assigned to each time series for realizing predictions in coarse and fine time-granularities, respectively. Specifically, the results from the coarse-grained predictors help the DRL modules in the NOrch make wise network orchestration decisions for provisioning delay-sensitive applications, while the fine-grained predictors are added in this work to get the necessary future information for scheduling delay-tolerant applications. Moreover, the coarse-grained predictors make the DRL modules converge fast, after the KD-NO system starts initially or loses its optimal state due to unexpected reasons, but the fine-grained predictors only kick in when the KD-NO system has converged to its optimal state. This is another reason why we need the two sets of predictors.

The telemetry data in the R&TD and the predictions from the VM-W&T-P are sent to the HOE-DCN status database in the NOrch periodically. The VM management module in the IT-C is in charge of invoking VM deployment and migration based on the instructions from the NOrch.

2) NET-C: It coordinates the routing of traffic from delay-sensitive applications over the electrical/optical inter-rack clouds. Here, we have a network abstraction module (NAM) to abstract the topology of inter-rack clouds, collect traffic routing information, and monitor the volumes of traffic going through switch ports. The telemetry data from the NAM gets stored in the traffic engineering database (TED), which in turn forwards the data to the HOE-DCN status database in the NOrch periodically. At the southbound direction, the network management module (NET management) routes the traffic from delay-sensitive applications over the electrical/optical inter-rack clouds according to the instructions from the NOrch.

3) Classic-C: It manages the service provisioning of delay-tolerant applications according to the instructions from the Classic Agent in the NOrch. As delay-tolerant applications essentially utilize the residual resources left by the delay-sensitive ones, we do not need to leverage DL/DRL to predict and arrange them, and a straightforward classic algorithm will be good enough to schedule them with deterministic information. In this work, we simplify the delay-tolerant applications and only consider those that involve delay-tolerant bulk data transfers [46], *i.e.*, the applications will not cause intensive IT resource utilizations[1]. Hence, the DCN management module will just regulate the applications' data transfers, *i.e.*, controlling their state time, durations, and data-rates.

[1]In principle, scheduling delay-tolerant applications to use deterministic residual IT resources does not have fundamental difference from scheduling them to use deterministic bandwidth resources. Therefore, the simplification here would not restrict the generality of our approach.

4) NOrch: There are four major components in the NOrch. Here, the DRL-based agent for VMs (DRL-VM Agent), experience database (E-DB), and the DRL-based substitute agent for VMs (DRL-VM Sub-Agent) actually form a component to make intelligent decisions on migrating the VMs for delay-sensitive applications. Specifically, in operation, the DRL-VM Agent periodically collects the latest network status from the HOE-DCN status database, and utilizes its trained neural network to wisely determine where and how to migrate the VMs for delay-sensitive applications. To efficiently train the DRL-VM Agent to do so, we add the E-DB and DRL-VM Sub-Agent in. Here, the DRL-VM Agent and DRL-VM Sub-Agent share the same neural network architecture, which adopts deep deterministic policy gradient (DDPG) and the Actor-Critic learning strategy [47].

In online training, the DRL-VM Agent makes decisions on VM migration based on the latest network status from the HOE-DCN status database, and records the current network state, its action on VM migration, the action's reward, and the network state after the action (*i.e.*, the next network state) as an entry of experience in the E-DB, after each decision making. When sufficient entries of experience get accumulated in the E-DB, the DRL-VM Sub-Agent first uses them to train its neural network, and then updates the neural network in the DRL-VM Agent with the training results. In this way, the DRL-VM Agent can make intelligent decisions at the front end, while the burden of its online training gets offloaded to the DRL-VM Sub-Agent that works in the background.

For the DRL-VM Agent and DRL-VM Sub-Agent, we define the following variables

- m_v: the boolean variable that equals 1 if we need to migrate VM v, and 0 otherwise.
- g_v: the boolean variable that equals 1 if VM v has degraded performance due to IT contention, and 0 otherwise.
- $g_{r,p}$: the boolean variable that equals 1 if port p in ToR switch r is congested, and 0 otherwise.

Then, their reward is formulated as

$$R_{vm} = -\alpha \cdot \sum_{v} g_v - \beta \cdot \sum_{r,p} g_{r,p} - \gamma \cdot \sum_{v} m_v, \tag{1}$$

where α , β and γ are positive weighting constants. In other words, the reward in Eq. (1) pushes the DRL-VM Agent to minimize the occurrences of IT and bandwidth resource contentions with the smallest number of VM migrations.

Similarly, the DRL-based agent for network (DRL-NET Agent), E-DB, and the DRL-based substitute agent for network (DRL-NET Sub-Agent) form the component to make intelligent decisions on routing the traffic from delay-sensitive applications over the electrical/optical inter-rack clouds. The multi-agent based scheme here results in modular designs for DRL-VM Agent and DRL-NET Agent such that they can arrange the VMs and traffic of delay-sensitive applications in a separated but coordinated manner. Hence, it significantly reduces the state-action space sizes of the DRL modules, eases their online training, and effectively improves the scalability of our KD-NO system. The design and operation principle of the DRL-NET Agent and the DRL-NET Sub-Agent are the same as those mentioned above, except for the reward formulation. We add a new variable to indicate the reconfiguration of the electrical/optical inter-rack clouds.

- m: the boolean variable that equals 1 if we need to reconfigure the inter-rack topology, and 0 otherwise.

Then, the reward of the DRL-NET Agent and the DRL-NET Sub-Agent is formulated as

$$R_{net} = -\beta \cdot \sum_{r,p} g_{r,p} - \delta \cdot m, \tag{2}$$

where δ is a positive weighting constant.

To come up with the optimal HOE-DCN configuration, the coordination of the DRL-VM Agent and DRL-NET Agent is as follows. In each service cycle, the DRL-VM Agent first makes the decision on VM migration based on the latest network status in the HOE-DCN status database, and instructs the VM management module in the IT-C to implement it. Then, based on the updated network status in the HOE-DCN status database, the DRL-NET Agent determines the configuration of the electrical/optical inter-rack clouds, and accomplishes the traffic routing with the NET management module in the NET-C. At this moment, the service provisioning of delay-sensitive applications is done, and the agent based on classic algorithm (Classic Agent) kicks in to arrange the data transfers of delay-tolerant ones. Specifically, under the constraint that the services of delay-sensitive applications should not be affected, Classic Agent uses a straightforward classic algorithm to determine the state time, duration, and data-rate of each serverable data transfer based on the residual bandwidth in the electrical/optical inter-rack clouds. The introduction of the Classic Agent further improves the scalability of our KD-NO.

IV. Operation and Coordination Procedure

A. Operation Design

In this work, we run Hadoop applications [48] in our HOE-DCN testbed as the delay-sensitive applications, while the delay-tolerant applications are chosen as bulk data transfers. The VMs for Hadoop applications are grouped into clusters, each of which can run CPU- or/and I/O-bound jobs. To emulate real Hadoop workloads, we generate the jobs according to the cluster-usage traces released by Google [49–51]. We reduce the timescale of Google job pattern for 120 times (*i.e.*, mapping 24 hours to 12 minutes) to expedite the experiments. The duration of each service cycle is set as 2 minutes. Based on this settings, we use Fig. 3 to explain the operation and coordination procedure of our KD-NO system.

It can be seen that our KD-NO system mainly involves six threads, each of which covers different portion(s) of the three operation phases. In **Phase I**, which is the initial state, we use *Thread* 1 to generate Hadoop jobs in the HOE-DCN testbed (*i.e.*, *Step* 1), and *Thread* 1 will run through all the three phases. Then, *Thread* 3 utilizes the VM-W&T-M to collect enough telemetry data regarding the VMs' workloads and traffic without invoking any reconfiguration on the HOE-DCN (*i.e.*, *Step* 2). Next, we summarize the telemetry data samples within each minute to obtain an aggregated data point, and *Thread* 4 uses the aggregated data points as the training set to train the coarse-grained predictors in the VM-W&T-P in the

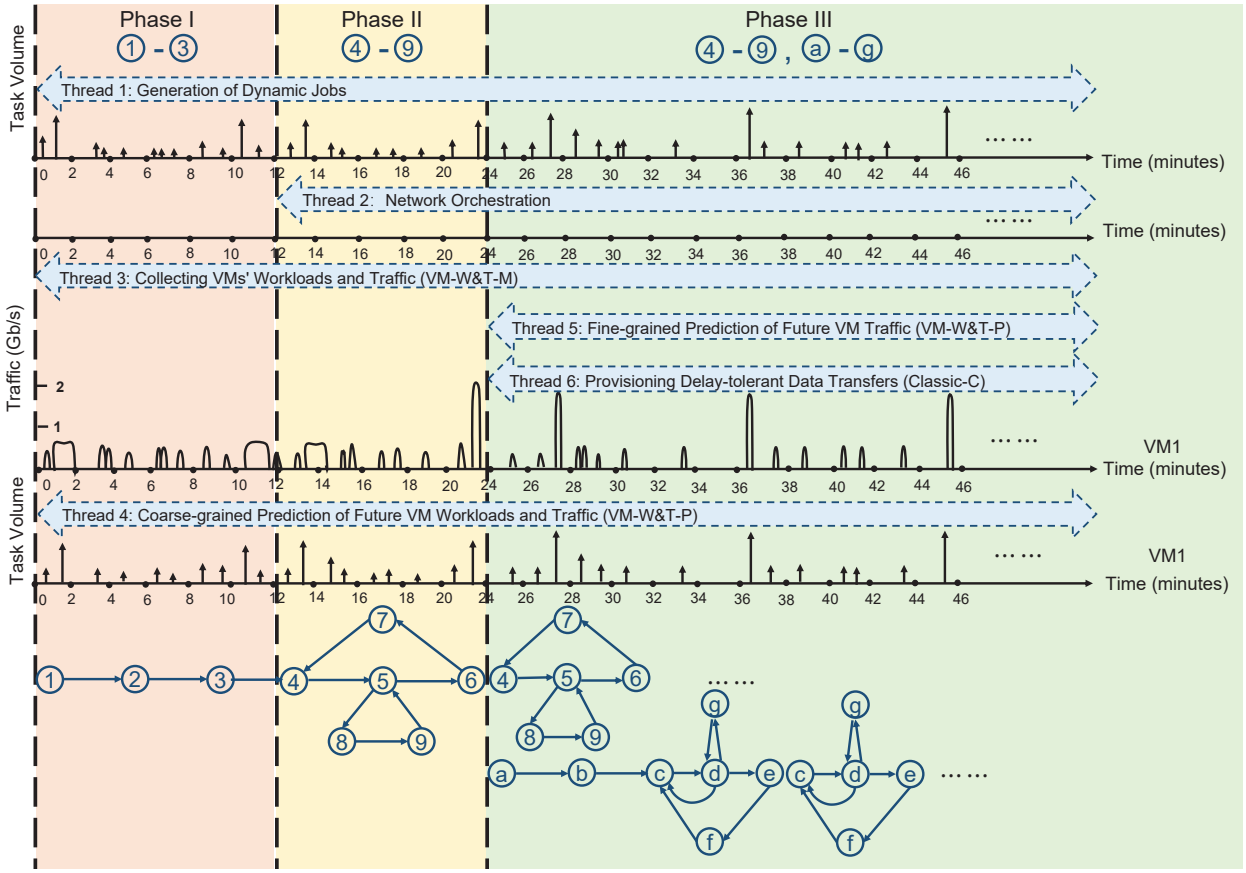


Fig. 3. Operation and coordination procedure of our KD-NO system.

offline manner (*i.e.*, *Step* 3). The running of *Threads* 3 and 4 will also cover all the three phases. After the KD-NO system has confirmed that the coarse-grained predictors in the VM-W&T-P can achieve sufficient prediction accuracy, it proceeds to **Phase II**. In summary, **Phase I** involves *Threads* 1, 3 and 4 and the events in it happen as *Step* 1→*Step* 2→*Step* 3.

Phase II is the non-steady network orchestration state. Starting from this phase, we put the DRL-VM and DRL-NET Agents into operation to orchestrate the IT and bandwidth resources in the HOE-DCN for provisioning Hadoop applications, and train the DRL-VM and DRL-NET Sub-Agents in the background. This is essentially *Thread* 2, which runs through **Phases II** and **III**. The details regarding **Phase II** are explained as follows. Firstly, at the beginning of each service cycle, *Thread* 4 forecasts the workloads and traffic of Hadoop-related VMs in this cycle and sends the results to the NOrch (*i.e.*, *Step* 4). Secondly, *Thread* 2 utilizes the DRL-VM and DRL-NET Agents to make network orchestration decisions, and then implement the decisions via the VM management and NET management modules (*i.e., Step* 5). Finally, *Thread* 3 collects telemetry data regarding the VMs' workloads and traffic (*i.e.*, *Step* 6), based on which *Thread* 4 updates the coarse-grained predictors in the VM-W&T-P on-the-fly with transfer learning (*i.e., Step* 7).

Therefore, the online training loop of the coarse-grained predictors in the VM-W&T-P works as *Step* 4→*Step* 5→*Step* 6→*Step* 7→*Step* 4, in **Phase II**. Meanwhile, the DRL-VM and DRL-NET Agents in *Thread* 2 collect network orchestration experiences (*i.e., Step* 8), to train their Sub-Agents in the background. Then, the updated parameters from the Sub-Agents get implemented in the DRL-VM and DRL-NET Agents periodically, for making them perform better (*i.e., Step* 9). Hence, the online training loop of the DRL-based modules works as *Step* 5→*Step* 8→*Step* 9→*Step* 5, in **Phase II**.

Finally, when the operations of the DRL-based modules in *Thread* 2 converge, the KD-NO system enters **Phase III**, which is the steady network orchestration state. Here, the online training loops of the coarse-grained predictors in the VM-W&T-P and the DRL-based modules still work according to their procedures in **Phase II**. Nevertheless, two new threads (*i.e., Threads* 5 and 6) are put into operation in **Phase III**. *Thread* 5 leverages the VM-W&T-M to collect enough telemetry data regarding the VMs' workloads and traffic (*i.e., Step* a), and first trains the fine-grained predictors in the VM-W&T-P with it (*i.e., Step* b).

When the offline training has been done, the fine-grained predictors in the VM-W&T-P can each time precisely forecast the traffic fluctuation on inter-rack links within the next 20 seconds (*i.e., Step* c). This information is sent to *Thread* 6, which then utilizes the Classic-C to schedule the data transfers of delay-tolerant applications and implement the scheduling results with the DCN management module (*i.e., Step* d). The operation involving *Steps* c and d can be repeated several times in each service cycle. Meanwhile, since the network

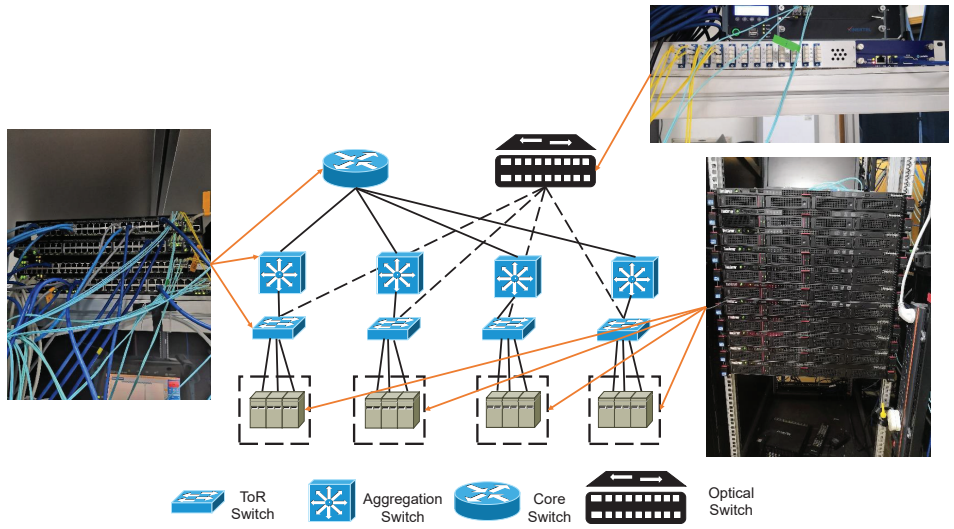


Fig. 4. Experimental setup of our HOE-DCN testbed.

orchestration for Hadoop applications may affect network status from time to time, we insert *Step g* in the operation, which updates the network status correctly before proceeding to scheduling the data transfers of delay-tolerant applications.

Next, *Thread* 5 collects telemetry data regarding the VMs' workloads and traffic (*i.e.*, *Step e*), based on which it updates the fine-grained predictors in the VM-W&T-P on-the-fly with transfer learning (*i.e., Step f*). Therefore, the online training loop of the fine-grained predictors in the VM-W&T-P works as *Step c*→*Step d*→*Step e*→*Step f*→*Step c*, in **Phase III**.

B. Scalability Analysis

The scalability of a DRL model can usually be quantified with the sizes of its state and action spaces. Hence, we analyze the state and action spaces of single- and multi-agent based KD-NO systems as follows.

Parameters:

- N_v: the total number of active VMs in the HOE-DCN.
- N_s: the total number of servers in the HOE-DCN.
- N_t: the total number of ToR switches in the HOE-DCN.
- L_c: the number of quantified levels for the CPU workload that a VM can take.
- L_t: the number of quantified levels for the traffic that a VM pair can have in between.
- M_v: the maximal number of VM migrations that can be invoked in each service cycle.

For the single-agent KD-NO in [24, 25], each state can be represented by the locations and CPU workloads of all the active VMs, and the traffic matrix of all the VM pairs. Therefore, its state space size can be formulated as

$$S_{sa} = (N_s \cdot L_c)^{N_v} \cdot (L_t)^{\frac{N_v \cdot (N_v-1)}{2}}, \tag{3}$$

while its actions include both VM migration and inter-rack cloud reconfiguration, and thus its action space size is

$$A_{sa} = \binom{M_v}{N_v} \cdot (N_s)^{M_v} \cdot 2^{\frac{N_t \cdot (N_t-1)}{2}}. \tag{4}$$

Here, since an inter-rack reconfiguration can change the communication between a ToR switch pair from the electrical cloud to the optical one or *vice versa*, the space size of the actions is $2^{\frac{N_t \cdot (N_t-1)}{2}}$.

For the multi-agent KD-NO, we first analyze the DRL-VM Agent. Its state space size $S_{da,v}$ takes the expression in Eq. (3), and its actions are only VM migrations with a size of

$$A_{da,v} = \binom{M_v}{N_v} \cdot (N_s)^{M_v}. \tag{5}$$

For the DRL-NET Agent, it does not need to consider the CPU workloads of VMs, and thus its state space size is

$$S_{da,t} = (N_s)^{N_v} \cdot (L_t)^{\frac{N_v \cdot (N_v-1)}{2}}, \tag{6}$$

and its actions only involve inter-rack reconfigurations with a space size of

$$A_{da,t} = 2^{\frac{N_t \cdot (N_t-1)}{2}}. \tag{7}$$

Eqs. (3)-(7) indicate that the state and action spaces of the multi-agent KD-NO are much smaller than those of the single-agent KD-NO, *i.e.*, its scalability would be much better.

V. Experimental Demonstrations

In this section, we first briefly introduce our system implementation and experimental setup, and then we perform several experiments to demonstrate the effectiveness of our KD-NO.

A. System Implementation and Experimental Setup

To demonstrate our proposed KD-NO system, we prototype a small-scale HOE-DCN testbed that includes four server racks. We develop the control plane based on the OpenStack cloud platform, while the data plane is built with Linux servers, hardware-based OpenFlow switches, and a reconfigurable OXC as the optical switch. The electrical inter-rack cloud is based on the hardware-based OpenFlow switches, which work as the ToR, aggregation and core switches and have ports based on 1 GbE. On the other hand, the optical inter-rack cloud is essentially the reconfigurable OXC that connects to

the 10 GbE optical ports on all the ToR switches. As shown in Fig. 2, the control plane consists of the IT-C, NET-C, Classic-C, and NOrch. For the IT-C, the VM management module is realized with OpenStack APIs, while the VM-W&T-M are implemented based on collectd [52] and sFlow [53]. We program the NET-C based on ONOS [54], and all the DL/DRL modules in the control plane are implemented based on TensorFlow. Both the Classic Agent and Classic-C are homemade to schedule and provision delay-tolerant data transfers with a simple algorithm, while the actual data transfers are realized with Pktgen-DPDK [55].

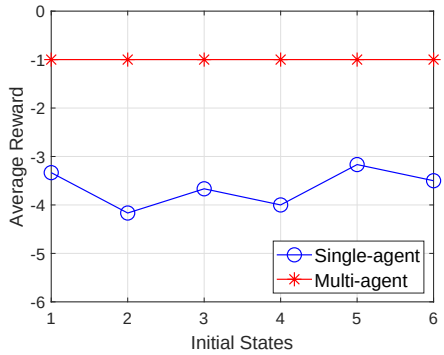
We conduct experiments in two scenarios to demonstrate the effectiveness of our KD-NO system. Firstly, the experiments compare the multi-agent based KD-NO with the single-agent based one in [24, 25]. Secondly, we utilize the KD-NO system to orchestrate IT and bandwidth resources in the HOE-DCN testbed to provision delay-sensitive Hadoop applications and delay-tolerant data transfers simultaneously.

B. Benchmarking over Single-agent DRL

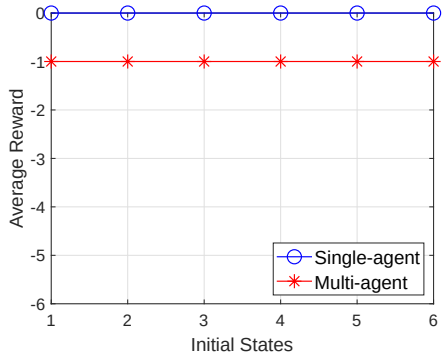
In this experiment, we use the single-agent based KD-NO system in [24, 25] as the benchmark to discuss the pros and cons of the multi-agent based one that is proposed in this work. The scalability of single- and multi-agent based KD-NO systems in terms of state/action space sizes has been analyzed in Section IV-B. Next, we will use experimental results to further justify the scalability improvement achieved by multi-agent KD-NO. We first restrict the number of training rounds that the sub-agents in the two KD-NO systems can perform in the initial state (*i.e.*, **Phase I** in Fig. 3) as $1,000$, and compare the average rewards obtained by the two systems. Specifically, we initialize the DRL-based modules with 6 different states, and for each initial state, we run the experiment for 6 times to average the results, for ensuring sufficient statistical accuracy.

Fig. 5(a) compares the average rewards from the single- and multi-agent based KD-NO systems. We observe that the number of training rounds is restricted, our multi-agent based KD-NO system outperforms the single-agent based benchmark for all the 6 cases. Moreover, we observe that the rewards from the multi-agent based KD-NO system are the same no matter which initial state is used. This suggests that the multi-agent based KD-NO system has already converged to the optimal state within $1,000$ rounds of training. However, the rewards from the single-agent based benchmark vary with the initial states, and they are significantly lower than those from the multi-agent based KD-NO system. Hence, the results in Fig. 5(a) confirm that our multi-agent based KD-NO system has better scalability than the single-agent based benchmark.

On the other hand, if we remove the restriction on the number of training rounds, Fig. 5(b) shows the average rewards from the two KD-NO systems. This time, we can see that the rewards from both systems have converged and the single-agent based benchmark outperforms our multi-agent based KD-NO system. This is actually expected, because the overall optimization space becomes smaller after we dividing the single DRL-based agent into two collaborative ones. Therefore, although the multi-agent based KD-NO system can converge to its optimal state quickly, the optimal state might still be a local optimum in the overall optimization space of the single-agent based benchmark. Fortunately, the results in Fig. 5(b) also suggest that the performance gap between the single- and multi-agent based KD-NO systems is relatively small. Therefore, we can conclude that compared with the single-agent based benchmark, the multi-agent based KD-NO system proposed in this work achieves better tradeoff between convergence speed and training performance. This conclusion can be further verified with the results in Fig. 6, which shows that the multi-agent based KD-NO system converges much faster (4 minutes) than the single-agent based benchmark (10 minutes) in online training (*i.e.*, **Phases II** and **III** in Fig. 3). Here, the relative reward refers to the difference between the reward of the action made by the NOrch in a network state and the optimal reward in the same state.



(a) Average rewards when training rounds are restricted



(b) Average rewards when training rounds are unrestricted

Fig. 5. Comparisons on rewards from single-/multi-agent based KD-NO.

C. Orchestrating Various Applications

In this experiment, we deploy 12 VMs in the four racks and group them into four Hadoop clusters, each of which has one name-node and two data-nodes to process Hadoop jobs generated according to the description in Section IV. We consider four different scenarios as follows to demonstrate the effectiveness of our multi-agent based KD-NO system.

- **No KD-NO**: this scenario does not involve any KD-NO, which means that it only runs delay-sensitive Hadoop applications (*i.e.*, including both CPU- and I/O-bound jobs) and will not adjust their VM locations and inter-rack

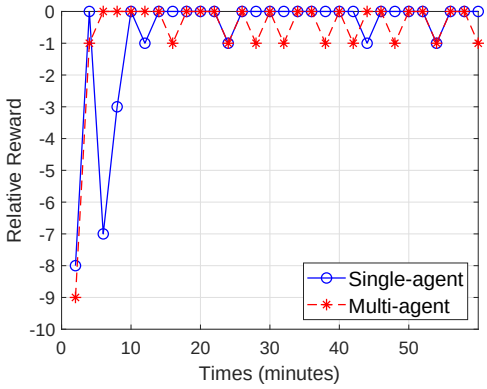


Fig. 6. Convergence performance in online training.

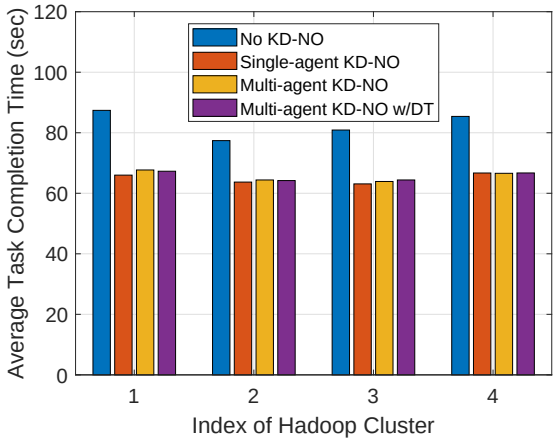


Fig. 7. Results on average job completion time in four scenarios.

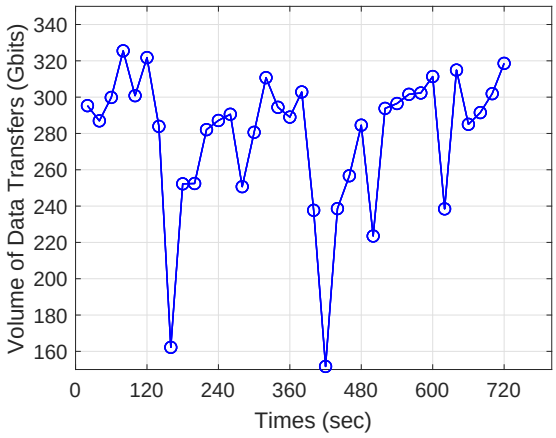


Fig. 8. Volume of data transfers in Multi-agent KD-NO w/DT.

traffic routing schemes throughout each experiment. The VMs are randomly deployed in the HOE-DCN testbed.

- **Single-agent KD-NO**: this scenario also only runs delay-sensitive Hadoop applications, but it leverages the single-agent based KD-NO system in [24, 25] to orchestrate the IT and bandwidth resources in the HOE-DCN testbed for the Hadoop jobs. Here, for fair comparison, we do not apply any restriction on the number of training rounds.
- **Multi-agent KD-NO**: this scenario uses our multi-agent based KD-NO to orchestrate the resources in the HOE-DCN testbed for delay-sensitive Hadoop applications.
- **Multi-agent KD-NO w/DT**: this scenario uses our multi-agent based KD-NO to provision both delay-sensitive Hadoop applications and delay-tolerant data transfers.

We plot the average job completion time of the Hadoop clusters in the four experimental scenarios in Fig. 7. It can be seen that: 1) all the scenarios with KD-NO provide significantly shorter job completion time than the one without KD-NO, which confirms the effectiveness of KD-NO, 2) the job completion time from the multi-agent based scenarios is just slightly longer than that from the single-agent based one[2], which verifies the performance of the multi-agent based KD-NO system, and 3) there is almost no noticeable difference between the job completion time from Multi-agent KD-NO and Multi-agent KD-NO w/DT, which suggests that the multi-agent based KD-NO system can provision delay-sensitive Hadoop applications and delay-tolerant data transfers according to their priorities, *i.e.*, the delay-tolerant data transfers would not compete with the delay-sensitive Hadoop applications for inter-rack bandwidth. For Multi-agent KD-NO w/DT, Fig. 8 shows the total volume of data transfers in the HOE-DCN every 20 seconds, which indicates how the multi-agent based KD-NO system adjusts data transfers adaptively to not only avoid affecting the delay-sensitive Hadoop applications but also utilize the residual inter-rack bandwidth effectively.

Note that, the reconfigurations invoked by KD-NO also cause inevitable service interruptions. To minimize this negative effect, we incorporate live VM migration, which keeps the applications running in a VM alive during migration, and optimize the implementation of SDN-based network reconfiguration. In the HOE-DCN, a VM migration and an inter-rack cloud reconfiguration cause 0.981 and 0.915 second service interruption (averaged over 20 measurements), respectively.

[2]In the experiments, we let the single-agent based KD-NO system train sufficient rounds in **Phase II**.

VI. Conclusion

In this paper, we investigated how to achieve scalable KD-NO for managing the delay-sensitive and delay-tolerant applications running in an HOE-DCN. For delay-sensitive applications, we leveraged a multi-agent scheme to distribute the tasks of placing VMs in server racks and routing VM traffic in inter-rack clouds to two cooperative DRL-based modules, respectively. Then, we designed a classic algorithm based module to provision delay-tolerant applications with the residual resources in the HOE-DCN. We described the operation and coordination procedure of the multi-agent based KD-NO system, and built a small HOE-DCN testbed that includes four racks to compare our proposal with a few benchmarks experimentally. Experimental results indicated that the multi-agent based KD-NO system can make timely and correct network orchestration decisions for managing delay-sensitive Hadoop applications and delay-tolerant data transfers simultaneously. The results also confirmed that our multi-agent

based KD-NO system has better scalability than the single-agent based benchmark, *i.e.*, it achieves better tradeoff between convergence speed and training performance.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC projects 61871357 and 61701472, CAS Key Project (QYZDY-SSW-JSC003), and SPR Program of CAS (XDC02010703)

REFERENCES

[1] W. Lu, L. Liang, B. Kong, B. Li, and Z. Zhu, "AI-assisted knowledge-defined network orchestration for energy-efficient datacenter networks," *IEEE Commun. Mag., in Press*, 2018.

[2] 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.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[3] "Cisco global cloud index: Forecast and methodology, 2016-2021." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[4] Y. Tian, R. Dey, Y. Liu, and K. Ross, "Topology mapping and geolocating for China's Internet," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, pp. 1908–1917, Sept. 2013.

[5] Z. Kerravala, "A data center fabric is critical to a next-generation data center," *Tech. Rep.*, pp. 1–9, Jan. 2014. [Online]. Available: https://www.customonline.com/assets/Z-Research-Data-Center-Fabric.pdf

[6] H. He, Y. Zhao, J. Wu, and Y. Tian, "Cost-aware capacity provisioning for internet video streaming CDNs," *Comput. J.*, vol. 58, pp. 3255–3270, Dec. 2015.

[7] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.

[8] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 339–350, Oct. 2010.

[9] 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.

[10] 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.*, vol. 5, pp. 836–847, Aug. 2013.

[11] Y. Yin, H. Zhang, M. Zhang, M. Xia, Z. Zhu, S. Dahlfort, and S. Yoo, "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.

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

[13] B. Kong, S. Liu, J. Yin, S. Li, and Z. Zhu, "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, 2018.

[14] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, K. Han, G. Fu, and Z. Zhu, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26 567–26 577, 2018.

[15] 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.*, vol. 7, pp. 314–324, Mar. 2015.

[16] P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," *IEEE Trans. Multimedia*, vol. 17, pp. 1297–1308, Aug. 2015.

[17] K. Wu, P. Lu, and Z. Zhu, "Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing," *IEEE Commun. Lett.*, vol. 20, pp. 684–687, Apr. 2016.

[18] W. Lu, P. Lu, Q. Sun, S. Yu, and Z. Zhu, "Profit-aware distributed online scheduling for data-oriented tasks in cloud datacenters," *IEEE Access*, vol. 6, pp. 15 629–15 642, 2018.

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

[20] S. Li, D. Hu, W. Fang, S. Ma, C. Chen, H. Huang, and Z. Zhu, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[21] X. Wang, Y. Tian, M. Zhao, M. Li, L. Mei, and X. Zhang, "PNPL: Simplifying programming for protocol-oblivious SDN networks," *Comput. Netw.*, vol. 147, pp. 64–80, Dec. 2018.

[22] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcon, M. Sole, V. Muntes-Mulero, D. Meyer, S. Barkai, and M. Hibbett, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, pp. 2–10, Jul. 2017.

[23] W. Lu, L. Liang, B. Kong, L. Li, and Z. Zhu, "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.

[24] Z. Zhu, W. Lu, L. Liang, and K. Kong, "Predictive analytics in hybrid optical/electrical DC networks," in *Proc. of OFC 2019*, pp. 1–3, Mar. 2019.

[25] H. Fang, W. Lu, Q. Li, J. Kong, L. Liang, B. Kong, and Z. Zhu, "Predictive analytics based knowledge-defined orchestration in a hybrid optical/electrical datacenter network testbed," *J. Lightw. Technol., in Press*, 2019.

[26] W. Lu and Z. Zhu, "Dynamic service provisioning of advance reservation requests in elastic optical networks," *J. Lightw. Technol.*, vol. 31, pp. 1621–1627, May 2013.

[27] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Trans. Cloud Comput.*, vol. 5, pp. 112–125, Jan. 2015.

[28] 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.

[29] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS One*, vol. 12, pp. 1–15, Apr. 2017.

[30] D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications," *J. Opt. Commun. Netw.*, vol. 10, pp. D126–D143, Oct. 2018.

[31] F. Khan, Q. Fan, C. Lu, and A. Lau, "An optical communication's perspective on machine learning and its applications," *J. Lightw. Technol.*, vol. 37, pp. 493–516, Jan. 2019.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[33] B. Li, W. Lu, S. Liu, and Z. Zhu, "Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks," *J. Opt. Commun. Netw.*, vol. 10, pp. D29–D41, Oct. 2018.

[34] J. Guo and Z. Zhu, "When deep learning meets inter-datacenter optical network management: Advantages and vulnerabilities," *J. Lightw. Technol.*, vol. 36, pp. 4761–4773, Oct. 2018.

[35] L. Velasco, L. Gifre, J. Izquierdo-Zaragoza, F. Paolucci, A. Vela, A. Sgambelluri, M. Ruiz, and F. Cugini, "An architecture to support autonomic slice networking," *J. Lightw. Technol.*, vol. 36, pp. 135–141, Jan. 2018.

[36] S. Liu, B. Niu, D. Li, M. Wang, S. Tang, J. Kong, B. Li, X. Xie, and Z. Zhu, "DL-assisted cross-layer orchestration in software-defined IP-over-EONs: From algorithm design to system prototype," *J. Lightw. Technol.*, vol. 37, pp. 4426–4438, Sept. 2019.

[37] G. Liu, K. Zhang, X. Chen, H. Lu, J. Guo, J. Yin, R. Proietti, Z. Zhu, and B. Yoo, "Hierarchical learning for cognitive end-to-end service provisioning in multi-domain autonomous optical networks," *J. Lightw. Technol.*, vol. 37, pp. 218–225, Jan. 2019.

[38] X. Chen, B. Li, R. Proietti, Z. Zhu, and B. Yoo, "Self-taught anomaly detection with hybrid unsupervised/supervised machine learning in optical networks," *J. Lightw. Technol.*, vol. 37, pp. 1742–1749, Apr. 2019.

[39] L. Velasco, B. Shariati, F. Boitier, P. Layec, and M. Ruiz, "A learning life-cycle to speed-up autonomic optical transmission and networking adoption," *J. Opt. Commun. Netw.*, vol. 11, pp. 226–237, May 2019.

[40] B. Niu, J. Kong, S. Tang, Y. Li, and Z. Zhu, "Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system," *IEEE Access*, vol. 7, pp. 82 413–82 423, 2019.

[41] X. Chen, B. Li, R. Proietti, H. Lu, Z. Zhu, and B. Yoo, "DeepRMSA: A deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *J. Lightw. Technol.*, vol. 37, pp. 4155–4163, Aug. 2019.

[42] B. Li, W. Lu, and Z. Zhu, "Deep-NFVOrch: Leveraging deep reinforcement learning to achieve adaptive vNF service chaining in EON-DCIs," *J. Opt. Commun. Netw., in Press*, 2019.

[43] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *J. Artif. Intell. Res.*, vol. 53, pp. 659–697, Aug. 2015.

[44] N. Luong, D. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts., in Press*, 2019.

[45] A. Weigend, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Routledge, 2018.

[46] W. Lu and Z. Zhu, "Malleable reservation based bulk-data transfer to recycle spectrum fragments in elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 2078–2086, May 2015.

[47] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, Feb. 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

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

[49] "Google Cluster-Usage Traces." [Online]. Available: https://github.com/google/cluster-data

[50] S. Di, D. Kondo, and W. Cirne, "Google hostload prediction based on bayesian model with optimized feature combination," *J. Parallel Distrib. Comput.*, vol. 74, pp. 1820–1832, Jan. 2014.

[51] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Tech. Rep. ISTC-CC-TR-12-101*, pp. 1–21, Apr. 2012. [Online]. Available: http://www.pdl.cmu.edu/PDL-FTP/CloudComputing/ISTC-CC-TR-12-101.pdf

[52] "collectd." [Online]. Available: https://collectd.org/

[53] "sFlow." [Online]. Available: https://sflow.org/

[54] "ONOS." [Online]. Available: https://onosproject.org/

[55] "Pktgen-dpdk." [Online]. Available: https://git.dpdk.org/