On the TPE Design to Efficiently Accelerate Hitless Reconfiguration of OCS-based DCNs

Qian Lv, Yuxiao Zhang, Shuoning Zhang, Ruoxing Li, Ke Meng, Bowen Zhang, Fuguang Huang, Xiaoliang Chen, and Zuqing Zhu, *Fellow, IEEE*

***Abstract*—Nowadays, the performance of data-center networks (DCNs) has become crucial for advancing large-scale computing applications. Hence, to improve the throughput, energy-efficiency and latency of DCNs, people are trying to replace the electrical packet switching (EPS) based spine switches with optical circuit switching (OCS) based ones. In an OCS-based DCN, topology engineering (TPE) is the key operation to dynamically reconfigure its inter-pod topology for accommodating traffic with optimized resource utilization. TPE consists of two highly-correlated steps, *i.e.*, optimizing the target physical inter-pod topology of the DCN based on a traffic matrix, and planning the procedure of OCS reconfiguration such that hitless transition can be achieved. In this paper, we study how to optimize the two steps jointly to efficiently accelerate the hitless reconfiguration of an OCS-based DCN. We formulate a mixed linear programming model (MILP) to solve the joint optimization exactly. Then, to solve the problem time-efficiently, we propose an approach that optimizes TPE design greedily according to various metrics to minimize the number of stages required in hitless reconfiguration for TPE. Extensive simulations verify the effectiveness of our proposals and demonstrate their benefits over existing benchmark.**

***Index Terms*—Optical circuit switching, Data-center networks (DCNs), Topology engineering, Hitless DCN reconfiguration.**

I. Introduction

OVER the past decade, the performance of data-centers (DCs) played a pivotal role in advancing the state of the art of large-scale computing applications [1, 2]. To address the ever-increasing computing and communication demands of these applications [3], data-center networks (DCNs) have been updated rapidly with electrical packet switching (EPS) techniques. Specifically, in these EPS-based DCNs, data is usually transferred between racks/pods through a hierarchical network whose topology is fixed and was planned according to peak traffic volumes (*i.e.*, the worst-case scenario). However, this limits the adaptivity and average resource utilization of DCNs [4], and moreover, EPS introduces high power consumption and long processing latency [5, 6]. As highly dynamic and skewed inter-rack/pod traffic has gradually dominated inter-rack/pod traffic in DCNs in recent years [7], more flexible and efficient DCN architectures are needed to meet the diverse topology/connectivity requirements [8, 9].

Compared with EPS paradigms, DCNs based on optical circuit switching (OCS) [10–15] can offer larger throughput, higher energy efficiency, and shorter data transfer latency potentially [4]. Furthermore, as an OCS-based DCN allows for dynamic reconfiguring its inter-rack/pod topology with topology engineering (TPE) [8], it can accommodate dynamic and skewed traffic demands better and effectively improve average resource usage in both the oblivious and traffic-aware manner [16]. Therefore, many OCS-based DCN architectures [7, 9, 17–20] have been proposed to explore the benefits of TPE. For instance, Fig. 1 illustrates the OCS-based DCN architecture developed by Google [7], which was evolved from the leaf-spine Clos architecture [21]. As shown in Fig. 1(a), the OCS-based DCN replaces the spine layer in Clos with a reconfigurable OCS layer (ROL), which directly interconnects the pods with OCS-based switches (*e.g.*, the micro-electro-mechanical systems (MEMS) based optical switches) to enable fast, reliable and energy-efficient inter-pod communications. The OCS-based switches in the ROL are divided into several groups (*e.g.*, *OCS'* 1 and 2 in Fig. 1(a)), and each pod connects to all the OCS groups for ensuring service availability during switch failures. By abstracting the ROL, we can get the logical inter-pod topology in Fig. 1(b). Then, by reconfiguring the OCS-based switches, TPE can rewire the inter-pod connections in the logical topology to adapt to various traffic matrices.

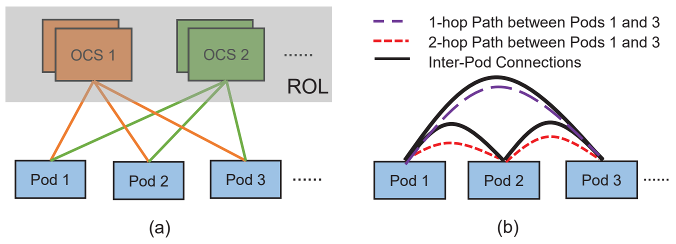


Fig. 1. Example on OCS-based DCN in leaf-spine, where the physical and logical inter-pod topologies are in (a) and (b), respectively.

Although TPE effectively improves the adaptivity of OCS-based DCNs, it can hardly be done in the hitless manner that does not induce noticeable service interruptions, due to the reconfiguration latency of OCS in milliseconds [8, 9]. This drawback makes it challenging for OCS-based DCNs to keep up with the rapid development of dynamic large-scale computing applications [22]. Therefore, it is relevant and necessary to optimize the design of TPE to facilitate hitless reconfiguration of OCS-based DCNs and make it support fine-grained traffic engineering (TE) better [8]. Specifically, a well-designed TPE scheme should consider the future (predicted/estimated) traffic matrix and the current physical inter-pod topology of an OCS-

Q. Lv, Y. Zhang, S. Zhang, R. Li, X. Chen, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (email: xlichen@ieee.org, zqzhu@ieee.org).

K. Meng, B. Zhang, and F. Huang are with Huawei Institute of Nanjing, Nanjing, Jiangsu 210012, China.

Manuscript received on April 5, 2024.

based DCN as inputs, to first optimize the target physical inter-pod topology according to the future traffic matrix and then plan the procedure of ROL reconfiguration to achieve hitless transition to it. These two steps of TPE are highly-correlated and thus should be optimized jointly, because the target physical inter-pod topology defines the complexity of the subsequent ROL reconfiguration, while the constraints associated with ROL reconfiguration affect the design of the target physical inter-pod topology. Nevertheless, most of the existing studies on TPE only tackled them separately [8, 23, 24], and to the best of our knowledge, how to optimize the two steps jointly has not been fully explored in the literature.

In this work, we study how to optimize the two steps of TPE jointly to efficiently accelerate the hitless reconfiguration of an OCS-based DCN. We assume that the OCS-based DCN is in the leaf-spine architecture shown in Fig. 1(a), due to its popularity in the industry. Then, in the first step, we need to get a target inter-pod topology that can realize or approximate the target logical inter-pod topology (obtained based on the future traffic matrix). This problem, although looks straightforward, is pretty challenging and complex, and we prove its $\mathcal{NP}$-hardness. Next, we formulate a mixed linear programming (MILP) model to tackle the joint optimization that covers the two steps of TPE. To improve the time efficiency of problem solving, we propose a time-efficient heuristic, which can solve the joint optimization greedily according to various metrics. Extensive simulations verify the effectiveness of our proposals and demonstrate the necessity of the joint optimization.

The rest of this paper is organized as follows. Section II briefly surveys the related work. In Section III, we present the problem description and formulate the MILP model. We propose the heuristic for TPE design in Section IV, and the simulations for performance evaluation are discussed in Section V. Finally, Section VI summarizes the paper.

II. Related Work

Since the late 2000s, researchers have considered the attempt of introducing OCS into DCNs to explore its benefits on throughput, latency, energy-efficiency and flexibility, and proposed numerous optics-related DCN architectures [17, 18, 25–30]. In order to explore the reconfigurability of these OCS-based DCN architectures to adapt to dynamic and skewed traffic matrices, people have designed a few DCN management approaches to improve the performance on throughput, latency, and resource utilization [20, 23, 24, 31–34]. The DCN management approaches optimized TPE or/and TE of OCS-based DCNs, where TPE determines the physical inter-rack/pod topology of an OCS-based DCN and TE plans the flow routing through the physical inter-rack/pod topology [8].

Between TPE and TE, TPE is more important to an OCS-based DCN, because its result defines the optimization space of TE and decides whether and how fast hitless reconfiguration can be accomplished. Therefore, a few existing studies have addressed the TPE of various OCS-based DCNs [9, 23, 24, 34, 35]. However, these studies only considered the first step of TPE, *i.e.*, getting the target physical inter-rack/pod topology of an OCS-based DCN based on a predicted/estimated traffic matrix. Note that, the second step of TPE (*i.e.*, planning the procedure of DCN reconfiguration to the target physical topology) should not be ignored, because it determines whether hitless transition can be achieved quickly, which is essential to ensure the quality-of-service (QoS) of many emerging large-scale computing applications [2]. Hence, researchers have also tackled the second step of TPE of different OCS-based DCNs. For example, Zhao *et al.* [36] proposed a progressive topology reconfiguration scheme for OSA [9], to guarantee the QoS of delay-sensitive flows and reduce the packet loss during reconfiguration, and the authors of [37] optimized the reconfiguration procedure of OCS-based DCNs in HyperFleX-LION [28]. Nevertheless, all these existing studies on TPE addressed its two steps separately or subsequently, which, as we have explained before, can lead to sub-optimal results.

To the best of our knowledge, how to optimize the two steps of TPE jointly to efficiently accelerate the hitless reconfiguration of an OCS-based DCN has not been addressed in the literature, especially for the leaf-spine architecture that has attracted intensive interests from the industry [7, 8, 19].

III. TPE to Accelerate Hitless OCS-based DCN Reconfiguration

In this section, we first describe the problem of designing TPE to accelerate hitless OCS-based DCN reconfiguration, then prove that its first step is $\mathcal{NP}$-hard, and finally formulate an MILP model to solve it optimally.

A. Problem Description

We consider an OCS-based DCN in the leaf-spine architecture shown in Fig. 1(a), where V pods are directly interconnected by the ROL that is built with OCS-based switches in T groups, each of which includes K switches. Each pod $v \in V$ possesses $L_{v,t,k}$ pair(s) of physical connections to the k-th switch in the t-th group. Here, each pair of physical connections are for duplex communications between two ports respectively on a pod and an OCS-based switch, and thus the connections can be abstracted as an undirected connection.

The first step of TPE (*i.e.*, *Step* 1 in the following discussions) is to optimize the target physical inter-pod topology. It can be further divided into two sub-steps:

1) Calculating target logical inter-pod topology based on the future (predicted/estimated) matrix of traffic among pods,
2) Optimizing target physical inter-pod topology to realize or approximate the obtained target logical inter-pod topology, such that the subsequent ROL reconfiguration can be accelerated with the smallest number of stages.

Note that, the current and target physical inter-pod topologies align with the current and future traffic matrices, respectively, and thus, the transition between them is hitless when the common part of the two traffic matrices is unaffected during the process. The first sub-step can be easily solved with matrix decomposition and matching [38], and thus its result solely depends on the future traffic matrix. Therefore, to optimize the two steps of TPE jointly, we can first tackle the first sub-step of *Step* 1 independently and then optimize the second sub-step of *Step* 1 and *Step* 2 jointly. Then, without lose of generality, we can assume that the result of the first sub-step of *Step* 1 (*i.e.*, the target logical inter-pod topology) is known as the input of the joint optimization considered in this work.

We denote the initial and target logical topologies with two sets of non-negative integer parameters $\{G_{u,v} : u, v \in V, u \neq v\}$ and $\{G'_{u,v} : u, v \in V, u \neq v\}$, where $G_{u,v}$ and $G'_{u,v}$ indicate the numbers of connections between pods v and u in the initial and target logical topologies, respectively. Similarly, the initial physical topology can also be represented with a set of non-negative integer parameters $\{S^{t,k}_{u,v} : u, v \in V, u \neq v, t \in [1, T], k \in [1, K]\}$, each of which tells the number of connections between pods u and v through the k-th switch in the t-th group. The common part of current and future traffic matrices is denoted as $\mathcal{R}$, where each flow $r(s_r, d_r, b_r) \in \mathcal{R}$ is defined by its source pod s_r, destination pod d_r, and bandwidth demand b_r. Then, the second sub-step of *Step* 1 is to optimize the target physical topology based on $\{G_{u,v}\}$, $\{G'_{u,v}\}$, $\{S^{t,k}_{u,v}\}$, and $\mathcal{R}$, such that the hitless reconfiguration to it in *Step* 2 can be accomplished progressively with the smallest number of stages. Here, to ensure hitless reconfiguration, each stage is based on "make-before-break" and contains two phases:

1) Obtaining an intermediate physical topology toward the target one, applying TE to drain the traffic on all the connections that will be torn down to realize the intermediate physical topology, and tearing down the connections.
2) Setting up new connections to realize the intermediate physical topology, and redistributing traffic with TE.

In order to make sure that the first phase of each stage does not cause unexpected congestion, we define η as the residual capacity ratio of the number of connections after the first phase to that after the second phase in previous stage, for each stage, and its value should never be less than a preset threshold η_{th}.

Figs. 2 and 3 provide illustrative examples to explain the hitless reconfiguration considered in this work. The initial physical and logical topologies are shown in Figs. 2(a) and 3(a), respectively, where the white lines on the two OCS-based switch groups in Fig. 2(a) indicate the initial connections between pods. Here, we assume that there are 2 switch groups, each of which includes a switch that contains 12 ports, and each pod connects 3 links to a switch group (the capacity of each link is 1 unit). There are 4 flows in the OCS-based DCN, and their sources, destinations and bandwidth demands are listed in the first column of the table in Fig. 3(f), and their TE results in the initial physical topology are also listed in the table. For instance, *Flow* 1 is between *Pods* 1 and 3 with a bandwidth demand of 1.5 units, and initially, it is split over two direct paths, which are through the first switches in the first and second groups, respectively (*i.e.*, the switches are respectively denoted as 1(1) and 2(1) in Fig. 3(f)).

Then, we start the hitless reconfiguration with $\eta_{th} = 65\%$. It can be seen that to reconfigure the initial physical topology in Fig. 2(a) to the target one in Fig. 2(c), we need to rewire 4 and 2 connections in *OCS'* 1(1) and 2(1), respectively, where the rewired connections are marked with colored lines in Fig. 2(c). Due to the constraint of $\eta_{th} = 65\%$, we cannot finish the physical topology reconfiguration in one stage. Therefore, the intermediate physical topology in Fig. 2(b) is designed,

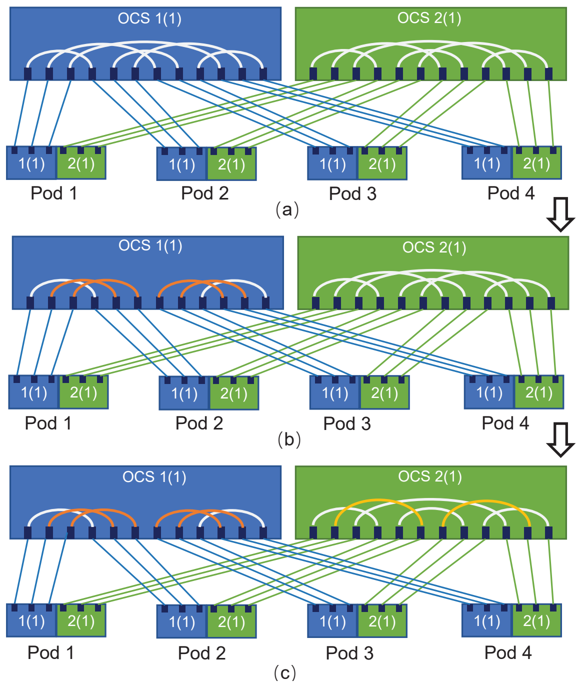
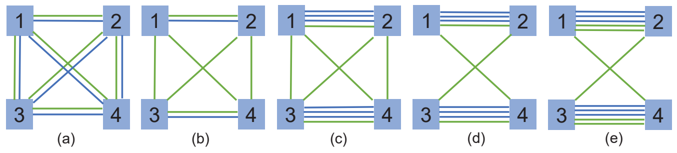


Fig. 2. Example on reconfiguring the initial physical topology in (a) to the target one in (c) with two stages, where the intermediate physical topology is in (b) (the physical topologies in (a)-(c) here correspond to the logic topologies in Figs. 3(a), 3(c) and 3(e), respectively).



Flow (u-v/b)	TE Results ($route$/OCS-$based\ Switch(es)$/b)				
	Initial Stage	Stage 1-1	Stage 1-2	Stage 2-1	Stage 2-2
1-3/1.5	1-3/1(1)/0.5; 1-3/2(1)/1.0;	1-3/2(1)/1.0; 1-2-3/2(1),2(1)/0.5;	1-3/2(1)/1.0; 1-2-3/2(1),2(1)/0.5;	1-2-3/2(1),2(1)/0.5; 1-4-3/2(1),1(1)/1.0;	1-2-3/2(1),2(1)/0.5; 1-4-3/2(1),1(1)/1.0;
1-2/1.5	1-2/1(1)/1.0; 1-2/2(1)/0.5;	1-2/1(1)/1.0; 1-2/2(1)/0.5;	1-2/1(1)/1.0; 1-2/2(1)/0.5;	1-2/1(1)/1.0; 1-2/2(1)/0.5;	1-2/1(1)/1.0; 1-2/2(1)/0.5;
3-4/2.0	3-4/1(1)/1.0; 3-4/2(1)/1.0;	3-4/1(1)/1.0; 3-4/2(1)/1.0;	3-4/1(1)/1.0; 3-4/2(1)/1.0;	3-4/1(1)/1.0; 3-4/2(1)/1.0;	3-4/1(1)/1.0; 3-4/2(1)/1.0;
2-4/0.5	2-4/2(1)/0.5;	2-4/2(1)/0.5;	2-4/2(1)/0.5;	2-3-4/2(1),1(1)/0.5;	2-3-4/2(1),1(1)/0.5;

(f)

Fig. 3. Continue of hitless reconfiguration example in Fig. 2, (a)-(e) logical topologies, and (f) TE results.

which only rewires 4 connections in *OCS* 1(1), and its logical topology is in Fig. 3(c). The hitless reconfiguration from the initial physical topology in Fig. 2(a) to the intermediate one in Fig. 2(b) is accomplished with two phases (*Stages* 1-1 and 1-2). Here, the logical topology after *Stage* 1-1 is shown in Fig. 3(b), where all the connections through the to-be-reconfigured ports in *OCS* 1(1) are torn down, and before tearing down the connections, the traffic on them has been redistributed to other connections as shown in Fig. 3(f). Similarly, the hitless reconfiguration from the intermediate physical topology in Fig. 2(b) to the target one in Fig. 2(c) is accomplished with *Stages* 2-1 and 2-2, whose logic topologies are in Figs. 3(d) and 3(e), respectively, and their TE results are listed in Fig. 3(f).

B. Optimization of Target Physical Inter-pod Topology

As explained above, the first sub-step of *Step* 1 needs to obtain and optimize the target physical inter-pod topology to realize or approximate the target logical inter-pod topology. Therefore, we first need to determine whether for an arbitrary target logical topology, there exists a target physical topology to realize it. The decision problem can be described as follows.

Parameters:

- V: the set of pods in the OCS-based DCN.
- T: the number of switch groups in ROL of the DCN.
- K: the number of OCS-based switches in each group.
- $L_{u,t,k}$: the number of physical links that pod $u \in V$ uses to connect to the k-th switch of the t-th group in ROL.
- $G'_{u,v}$: the number of connections between pods u and v in the target logical topology ($u, v \in V$).

Decision Variables:

- $D^{k,t}_{u,v}$: the integer variable that indicates the number of connections between pods u and v through the k-th switch of the t-th group in the target physical topology.

Constraints:

$$G'_{u,v} \leq \sum_{t=1}^{T}\sum_{k=1}^{K} D^{k,t}_{u,v}, \quad \forall u, v \in V. \tag{1}$$

Eq. (1) ensures that the target physical topology satisfies the target logical topology defined by $\{G'_{u,v}\}$.

$$\begin{cases} D^{k,t}_{u,v} = D^{k,t}_{v,u}, \ \{u, v : u, v \in V, u \neq v\}, \\ D^{k,t}_{u,u} = 0, \ \forall u \in V, \end{cases} \forall t \in [1, T], k \in [1, K]. \tag{2}$$

Eq. (2) ensures that the target physical topology is correctly determined to satisfy symmetry.

$$\sum_{u} D^{k,t}_{u,v} \leq L_{v,t,k}, \quad \forall v, t, k. \tag{3}$$

Eq. (3) ensures that the target physical topology does not require more ports than each pod can offer.

Theorem 1. *The decision problem of whether an arbitrary logical inter-pod topology can be realized by a physical inter-pod topology is $\mathcal{NP}$-complete.*

Proof: We prove that the decision problem is $\mathcal{NP}$-complete with the restriction method [39], *i.e.*, reducing it to a special case that is the general case of a known $\mathcal{NP}$-complete problem. Specifically, we apply the following restrictions:

- We set $T = 1$, *i.e.*, only one switch group is considered.
- We set $L_{u,t,k} = 1$, *i.e.*, pod $u \in V$ uses one physical link to connect to the k-th switch of the t-th group in ROL.
- We consider $G'_{u,v} \in \{0, 1\}$, *i.e.*, there is 0 or 1 connection between pods u and v in the target logical topology.

As we have $T = 1$, $D^{k,t}_{u,v}$ can be denoted as $D^{k}_{u,v}$, and then, Eqs. (1)-(3) are transformed as

$$G'_{u,v} \leq \sum_{k=1}^{K} D^{k}_{u,v}, \quad \forall u, v \in V, \tag{4}$$

$$\begin{cases} D^{k}_{u,v} = D^{k}_{v,u}, \ \{u, v : u, v \in V, u \neq v\}, \\ D^{k}_{u,u} = 0, \ \forall u \in V, \end{cases} \forall k \in [1, K], \tag{5}$$

$$\sum_{u} D^{k}_{u,v} \leq 1, \quad \forall v, k. \tag{6}$$

We know $D^{k}_{u,v} \in \{0, 1\}$ based on Eq. (6). As $D^{k}_{u,v}$ satisfies Eq. (4) and we have $G'_{u,v} \in \{0, 1\}$, $D^{k}_{u,v}$ satisfies Eq. (7) in certain cases of $u, v \in V$ according to Eq. (4).

$$\sum_{k=1}^{K} D^{k}_{u,v} \geq 1, \quad \forall u, v \in V. \tag{7}$$

Then, if we treat K as the number of colors and each connection between two nodes u and v as a virtual node, and connect two virtual nodes with a virtual link if the connections that they represent share same end-node(s), Eqs. (5)-(7) describe a general case of the multi-coloring problem. Specifically, the multi-coloring problem needs to determine when each virtual node needs to be colored with at least one color (Eq. (7)) and two connected virtual nodes have to be colored differently (Eq. (5)), whether we can color all the virtual nodes with no more than K colors? Therefore, we polynomially reduce the decision problem defined by Eqs. (4)-(7) to the multi-coloring problem. Obviously, if we can answer the multi-coloring problem ($\sum_{k} D^{k}_{u,v} \geq 1$) in polynomial time, we should be able to answer the classical graph coloring problem (GCP) ($\sum_{k} D^{k}_{u,v} = 1$) in polynomial time. This, however, is contradictive to the fact that the classical GCP is $\mathcal{NP}$-complete [40]. Hence, we prove that the decision problem of whether an arbitrary logical inter-pod topology can be realized by a physical inter-pod topology is $\mathcal{NP}$-complete. ■

We can easily provide an example to show that it is infeasible to find a physical inter-pod topology to realize a logical inter-pod topology. Specifically, we set $L_{u,t,k} = 3$, $T = 2$ and $K = 1$, and then it is not possible to obtain a target physical topology to realize the following logical topology.

$$G'_{u,v} = \begin{pmatrix} 0 & 3 & 3 \\ 3 & 0 & 3 \\ 3 & 3 & 0 \end{pmatrix}$$

This is because we can only map $G'_{u,v}$ to three or more OCS-based switches, such as $D^{1,1}_{u,v}$, $D^{2,1}_{u,v}$ and $D^{3,1}_{u,v}$ in the following.

$$D^{1,1}_{u,v} = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

$$D^{2,1}_{u,v} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

$$D^{3,1}_{u,v} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Note that, as the decision problem for the second sub-step of *Step* 1 of TPE design is $\mathcal{NP}$-complete, we can easily verify that the overall problem of TPE design is $\mathcal{NP}$-hard. Consequently, in the following analysis, when determining the target physical inter-pod topology, we only try to optimize it to realize or approximate the target logical inter-pod topology.

C. MILP Model

We formulate an MILP model to describe the joint optimization for TPE design as follows.

Objective:

The joint optimization is to design TPE for an OCS-based DCN such that the TPE-induced hitless reconfiguration can be finished with the smallest number of stages

$$\text{Minimize} \quad S, \tag{8}$$

where S is the number of stages required in *Step* 2.

Parameters:

- V: the set of pods in the OCS-based DCN.
- T: the number of switch groups in ROL of the DCN.
- K: the number of OCS-based switches in each group.
- $\mathcal{R}$: the matrix for the common inter-pod traffic between the current and future traffic matrices, where each flow r is denoted as $r(s_r, d_r, b_r)$.
- $L_{u,t,k}$: the number of physical links that pod $u \in V$ uses to connect to the k-th switch of the t-th group in ROL.
- $G'_{u,v}$: the number of connections between pods u and v in the target logical topology ($u, v \in V$).
- B_u: the capacity of a switch port on a pod $u \in V$.
- $S^{k,t}_{u,v}$: the number of connections between pods u and v through the k-th switch of the t-th group in the initial physical topology ($u, v \in V$).
- η_{th}: the threshold on residual capacity ratio of each stage.
- N: a large positive integer.

Decision Variables:

- $D^{k,t}_{u,v}$: the integer variable that indicates the number of connections between pods u and v through the k-th switch of the t-th group in the target physical topology.
- $M^{k,t,i}_{u,v}$: the integer variable for the number of connections between pods u and v through the k-th switch of the t-th group after the first phase of stage i.
- $C^{k,t,i}_{u,v}$: the integer variable for the number of connections between pods u and v through the k-th switch of the t-th group after the second phase of stage i.
- $z^{r,i}_{u,v}$: the integer variable that indicates the bandwidth occupied by flow $r \in \mathcal{R}$ on connection(s) between pods u and v after the first phase of stage i.
- $f^{r,i}_{u,v}$: the integer variable that indicates the bandwidth occupied by flow $r \in \mathcal{R}$ on connection(s) between pods u and v after the second phase of stage i.
- $h^{r,i}_{u,v}$: the boolean variable that equals 1 if the connections between pods u and v will be used to route the flow r after the first phase of stage i.
- $h'^{r,i}_{u,v}$: the boolean variable that equals 1 if the connections between pods u and v will be used to route the flow r after the second phase of stage i.
- S: the integer variable that indicates the number of stages required to finish the TPE-induced hitless reconfiguration.
- s_i: the boolean variable that equals 1 if stage i exists in the TPE-induced hitless reconfiguration, and 0 otherwise.
- $\phi^{k,t}_{u,v}$: the auxiliary integer variables for linearization.
- $\varphi^{k,t}_{u,v}$: the auxiliary boolean variables for linearization.

Constraints:

$$G'_{u,v} \le \sum_{t=1}^{T}\sum_{k=1}^{K} D^{k,t}_{u,v}, \quad \forall u, v \in V. \tag{9}$$

Eq. (9) ensures that the obtained target physical topology satisfies the target logical topology defined by $\{G'_{u,v}\}$.

$$\begin{cases} C^{k,t,i}_{u,v} = C^{k,t,i}_{v,u}, \ \{u, v : u, v \in V, u \neq v\}, \\ C^{k,t,i}_{u,u} = 0, \ \forall u \in V, \end{cases} \quad \forall t \in [1,T], k \in [1,K], i \in [0,N]. \tag{10}$$

Eq. (10) ensures that the intermediate physical topology in each stage is correctly determined.

$$\begin{cases} C^{k,t,0}_{u,v} = S^{k,t}_{u,v}, \\ C^{k,t,N}_{u,v} = D^{k,t}_{u,v}, \end{cases} \quad \forall u, v, t, k. \tag{11}$$

Eq. (11) ensures that the physical topologies of stages 0 and N are just the initial and target physical topologies, respectively.

$$\sum_{u} C^{k,t,i}_{u,v} \le L_{v,t,k}, \quad \forall v, t, k, i. \tag{12}$$

Eq. (12) ensures that the physical topology in each stage does not require more ports than each pod can offer.

$$\begin{cases} C^{k,t,i}_{u,v} \ge M^{k,t,i}_{u,v}, \\ C^{k,t,i-1}_{u,v} \ge M^{k,t,i}_{u,v}, \end{cases} \quad \forall u, v, t, k, i \in [1,N]. \tag{13}$$

Eq. (13) ensures that the relations between variables $\{M^{k,t,i}_{u,v}\}$ and $\{C^{k,t,i}_{u,v}\}$ are correctly determined.

$$\sum_{i=1}^{N}\left(C^{k,t,i-1}_{u,v} - M^{k,t,i}_{u,v}\right) = \frac{1}{2}\left(\phi^{k,t}_{u,v} + S^{k,t}_{u,v} - D^{k,t}_{u,v}\right), \ \forall u, v, t, k, \tag{14}$$

$$\sum_{i=1}^{N}\left(C^{k,t,i}_{u,v} - M^{k,t,i}_{u,v}\right) = \frac{1}{2}\left(\phi^{k,t}_{u,v} + D^{k,t}_{u,v} - S^{k,t}_{u,v}\right), \ \forall u, v, t, k, \tag{15}$$

$$\begin{cases} D^{k,t}_{u,v} - S^{k,t}_{u,v} \le \phi^{k,t}_{u,v}, \\ S^{k,t}_{u,v} - D^{k,t}_{u,v} \le \phi^{k,t}_{u,v}, \\ \phi^{k,t}_{u,v} \le D^{k,t}_{u,v} - S^{k,t}_{u,v} + (1 - \varphi^{k,t}_{u,v}) \cdot N, \\ \phi^{k,t}_{u,v} \le S^{k,t}_{u,v} - D^{k,t}_{u,v} + \varphi^{k,t}_{u,v} \cdot N, \end{cases} \quad \forall u, v, t, k. \tag{16}$$

Eqs. (14)-(16) ensure that the physical topology change between adjacent stages is correctly determined, where Eq. (16) is introduced to linearize $\left|D^{k,t}_{u,v} - S^{k,t}_{u,v}\right|$.

$$\sum_{u\in V}\sum_{v\in V}\sum_{k=1}^{K}\sum_{t=1}^{T} M^{k,t,i}_{u,v} \ge \sum_{u\in V}\sum_{v\in V}\sum_{k=1}^{K}\sum_{t=1}^{T} C^{k,t,i-1}_{u,v} \cdot \eta_{th}, \quad \forall i \in [1,N]. \tag{17}$$

Eq. (17) ensures that the constraint on residual capacity ratio is always satisfied in the first phase of each stage.

$$\begin{cases} \displaystyle\sum_{v\in V} \frac{z^{r,i}_{s_r,v} + z^{r,i}_{v,d_r}}{2} + z^{r,i}_{s_r,d_r} = b_r, \\ z^{r,i}_{v,d_r} = z^{r,i}_{s_r,v}, \{v : v \neq s_r, d_r, v \in V\}, \end{cases} \quad \forall r, i, \tag{18}$$

$$\begin{cases} \displaystyle\sum_{v\in V} \frac{f^{r,i}_{s_r,v} + f^{r,i}_{v,d_r}}{2} + f^{r,i}_{s_r,d_r} = b_r, \\ f^{r,i}_{v,d_r} = f^{r,i}_{s_r,v}, \{v : v \neq s_r, d_r, v \in V\}, \end{cases} \quad \forall r, i, \tag{19}$$

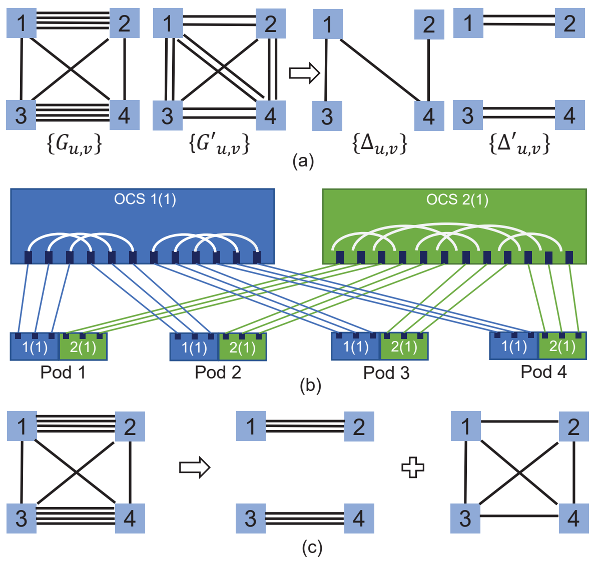


Fig. 4. Examples on (a) getting $\{\Delta'_{u,v}\}$ and $\{\Delta_{u,v}\}$ based on initial and target logical topologies, (b) initial physical topology, and (c) decomposing the initial physical topology in (b) to independent sub-topologies.

$$\sum_{r\in\mathcal{R}} f_{u,v}^{r,i} \le \sum_{k=1}^{K}\sum_{t=1}^{T} C_{u,v}^{k,t,i}\cdot \min\left(B_u, B_v\right), \quad \forall u,v,i, \tag{20}$$

$$\sum_{r\in\mathcal{R}} z_{u,v}^{r,i} \le \sum_{k=1}^{K}\sum_{t=1}^{T} M_{u,v}^{k,t,i}\cdot \min\left(B_u, B_v\right) \quad \forall u,v,i. \tag{21}$$

Eqs. (18)-(21) ensure that each flow $r \in \mathcal{R}$ is routed over available paths in each stage. More specifically, Eqs. (18) and (19) ensure the flow conservation conditions within 2 hops, while Eqs. (20) and (21) ensure that the bandwidth usages of all the flows in each stage satisfy the port capacity constraint.

$$\begin{cases} C_{u,v}^{k,t,i} - C_{u,v}^{k,t,i-1} \le N\cdot s_i, \\ C_{u,v}^{k,t,i-1} - C_{u,v}^{k,t,i} \le N\cdot s_i, \end{cases} \forall u,v,t,k,i\in[1,N]. \tag{22}$$

$$S = \sum_{i=1}^{N} s_i, \tag{23}$$

Eqs. (22)-(23) ensure that the number of required stages S is correctly determined.

IV. Design of Heuristic Algorithm

Although the MILP above can solve the joint optimization exactly, the problem-solving can be time-consuming or even intractable due to the $\mathcal{NP}$-hardness of the problem. In this section, we propose a time-efficient heuristic.

A. Overall Procedure

Given the inherent complexity in solving the problem of TPE design directly, we first decompose it into two subproblems, and then propose time-efficient heuristics to solve the subproblems in sequence. Specifically, according to our discussions in Section III-A, our TPE design considers

1) Optimizing target physical topology based on the target logical topology in consideration of minimizing the required stages in subsequent hitless reconfiguration (*i.e.*, the second sub-step of *Step* 1),
2) Obtaining a sequence of intermediate physical topologies to realize the hitless reconfiguration to the target physical topology with the smallest number of stages (*i.e.*, *Step* 2),

which correspond to the two subproblems addressed in the following. Different from the existing TPE design approaches that tackle the subproblems separately, our proposed algorithm tries to address them jointly, which means that when solving the first subproblem, we try to select the solution that will benefit the second subproblem most in the greedy manner. Moreover, we hope to point out that a well-designed greedy scheme for solving the first subproblem is not as straightforward as it seems to be. For example, an intuitive metric to consider when solving the first subproblem is the number of rewirings needed to change the initial physical topology to the target one [41]. However, as we will show in the performance evaluation in Section V, minimizing rewirings might not reduce the required stages from the second subproblem to the largest extent.

In the rest of this section, we present six algorithms (*i.e.*, *Algorithms* 1-6) to tackle the joint optimization of TPE design. In particular, *Algorithms* 1 and 2 describe the procedures for computing the target physical topology, while *Algorithms* 3-5 calculate three metrics that can be exploited by *Algorithm* 1 to assist in selecting sub-topologies greedily to build the target physical topology (*Line* 18 of *Algorithm* 1). *Algorithm* 6 is for optimizing the procedure of hitless reconfiguration. Consequently, different combinations of *Algorithms* 1, 2, 6 and *Algorithms* 3-5 lead to different TPE design algorithms, namely, Adapted-Gemini [8] (*Algorithms* 1, 2, 6 and *Algorithm* 3), Min-Rerouting (*Algorithms* 1, 2, 6 and *Algorithm* 4), and Simp-TPChange (*Algorithms* 1, 2, 6 and *Algorithm* 5).

B. Algorithm to Obtain Target Physical Topology

1) Main Procedures to Obtain Target Physical Topology: *Algorithm* 1 shows the procedure that we propose to solve the first subproblem to obtain the target physical topology for TPE. *Line* 1 compares the initial and target logical topologies to get the to-be-removed and to-be-added connections, and puts them in sets $\{\Delta'_{u,v}\}$ and $\{\Delta_{u,v}\}$, respectively. Fig. 4(a) shows an example on how to get $\{\Delta'_{u,v}\}$ and $\{\Delta_{u,v}\}$ based on initial and target logical topologies. Then, we decompose the initial physical topology into independent sub-topologies $\{\mathcal{G}^{k,t}\}$, each of which only uses an OCS-based switch (*Line* 2). Fig. 4 also explains the operation in *Line* 2. Specifically, with the initial physical topology in Fig. 4(b), we can decompose it to obtain two independent sub-topologies, each of which corresponds to an OCS-based switch, as shown in Fig. 4(c).

Next, *Lines* 3-11 try to directly implement as many to-be-added connections in $\Delta_{u,v}$ as possible. As the connections can be set up without removing any existing ones, implementing them helps to reduce the stages required in subsequent hitless reconfiguration. *Line* 12 initializes a flag for stopping the while-loop of *Lines* 13-24, which updates the sub-topologies in $\{\mathcal{G}^{k,t}\}$ iteratively until all the topology changes in $\{\Delta'_{u,v}\}$ and $\{\Delta_{u,v}\}$ have been applied or the number of iterations in *Algorithm* 2 reaches its maximum (I_m). In each iteration, we build the to-be-removed sub-topology with the connections, each of which carries the smallest traffic between a pod pair

Algorithm 1: Obtaining Target Physical Topology for TPE

```
   **Input**: $\{G_{u,v}\}$, $\{G'_{u,v}\}$, $\{S^{t,k}_{u,v}\}$, $\mathcal{R}$ and $I_m$.
   **Output**: Target physical topology $\{D^{k,t}_{u,v}\}$.
1  compare $\{G_{u,v}\}$ and $\{G'_{u,v}\}$ to get to-be-removed and
   to-be-added connections to put in $\{\Delta'_{u,v}\}$ and $\{\Delta_{u,v}\}$;
2  decompose $\{S^{k,t}_{u,v}\}$ into independent sub-topologies
   $\{\mathcal{G}^{k,t}\}$, each of which corresponds to a switch;
3  sort $\{\Delta_{u,v}\}$ in descending order of the number of
   to-be-added connections between pod pair $u$-$v$;
4  for each sub-topology in $\{\mathcal{G}^{k,t}\}$ do
5      for each $\Delta_{u,v}$ in $\{\Delta_{u,v}\}$ in sorted order do
6          add the connections that can be added between $u$
           and $v$ in $\mathcal{G}^{k,t}$, and update $\Delta_{u,v}$ accordingly;
7          if $\Delta_{u,v} = 0$ then
8              remove $\Delta_{u,v}$ from $\{\Delta_{u,v}\}$;
9          end
10     end
11 end
12 $flag = 0$;
13 while $\{\Delta_{u,v}\} \neq \emptyset$ AND $flag = 0$ do
14     initialize to-be-removed sub-topology $\{\Delta^*_{u,v}\} = \emptyset$;
15     for each pod pair $u$-$v$ with $\Delta'_{u,v} \neq 0$ do
16         select one connection between $u$ and $v$ and add it
           as $\Delta^*_{u,v}$ in $\{\Delta^*_{u,v}\}$;
17     end
18     apply a sub-procedure to obtain metric $\xi_{k,t}$, set of
       to-be-removed connections $\mathcal{G}^{k,t}_{-}$, set of to-be-added
       connections $\mathcal{G}^{k,t}_{+}$, and set of added connections $\mathcal{G}^{\star k,t}_{+}$
       for each sub-topology $\mathcal{G}^{k,t}$;
19     if all the sets in $\{\mathcal{G}^{\star k,t}_{+}\}$ are empty then
20         apply Algorithm 2 and set $flag = 1$;
21     else
22         select the sub-topology $\mathcal{G}^{k,t}$ whose metric $\xi_{k,t}$ is
           the largest, apply changes in $\mathcal{G}^{k,t}_{-}$ and $\mathcal{G}^{k,t}_{+}$ to
           $\mathcal{G}^{k,t}$, and update $\Delta_{u,v}$ and $\Delta'_{u,v}$;
23     end
24 end
25 get target physical topology $\{D^{k,t}_{u,v}\}$ based on $\{\mathcal{G}^{k,t}\}$;
```

(*Lines* 14-17), apply a sub-procedure to get a metric $\xi_{k,t}$ for each sub-topology $\mathcal{G}^{k,t}$ together with the to-be-removed, to-be-added and added connections related to it (*Line* 18). Here, we consider three scenarios for the sub-procedure in *Line* 18, and will discuss them in the next subsection. If there is at least one sub-topology to which we can add connection(s), *Line* 22 selects the sub-topology whose metric is the largest to implement the corresponding topology changes. Otherwise, we use *Algorithm* 2 to gradually reorganize sub-topologies $\{\mathcal{G}^{k,t}\}$ to add connections in $\{\Delta_{u,v}\}$. Finally, *Line* 25 obtains the target physical topology $\{D^{k,t}_{u,v}\}$ based on $\{\mathcal{G}^{k,t}\}$.

Algorithm 2 explains how to reorganize sub-topologies $\{\mathcal{G}^{k,t}\}$ if we cannot add a connection to any of them but $\{\Delta_{u,v}\}$ has not become empty yet. The while-loop of *Lines* 2-19 keeps modify the sub-topologies in $\{\tilde{\mathcal{G}}^{k,t}\}$, which is initialized as $\{\mathcal{G}^{k,t}\}$ in *Line* 1, until the set of to-be-added

Algorithm 2: Sub-procedure of *Algorithm* 1

```
1  $I = 0$, $\{\tilde{\mathcal{G}}^{k,t}\} = \{\mathcal{G}^{k,t}\}$;
2  while $\{\Delta_{u,v}\} \neq \emptyset$ and $I < I_m$ do
3      for each sub-topology in $\{\tilde{\mathcal{G}}^{k,t}\}$ do
4          select a connection between each connected pod
           pair $u$-$v$ in the sub-topology to add in $\{\Delta^{-\star}_{u,v}\}$;
5      end
6      $\{\tilde{\Delta}_{u,v}\} = \{\Delta^{-\star}_{u,v}\} + \{\Delta_{u,v}\} - \{\Delta'_{u,v}\}$;
7      $ind = 1$, $\{\mathcal{G}^{\star k,t}\} = \{\tilde{\mathcal{G}}^{k,t}\}$;
8      while $ind \leq K \cdot T$ do
9          for each sub-topology in $\{\mathcal{G}^{\star k,t}\}$ do
10             solve a max-flow problem to find connections
               in $\{\tilde{\Delta}_{u,v}\}$ that can be added to $\mathcal{G}^{\star k,t}$ ($\mathcal{G}^{\star k,t}_{+}$);
11             apply sub-procedure to update $\xi_{k,t}$ and $\mathcal{G}^{\star k,t}_{-}$;
12         end
13         select the sub-topology $\mathcal{G}^{\star k,t}$ whose metric $\xi_{k,t}$ is
           the largest ($\mathcal{G}^{\star k,t}_{\max}$), apply changes in $\mathcal{G}^{\star k,t}_{-}$ and
           $\mathcal{G}^{\star k,t}_{+}$ to $\mathcal{G}^{\star k,t}$, and update $\tilde{\Delta}_{u,v}$ and $\Delta'_{u,v}$;
14         remove sub-topology $\mathcal{G}^{\star k,t}_{\max}$ from $\{\mathcal{G}^{\star k,t}\}$ and
           update $\{\tilde{\mathcal{G}}^{k,t}\}$ with $\mathcal{G}^{\star k,t}_{\max}$;
15         $ind = ind + 1$;
16     end
17     update $\{\mathcal{G}^{k,t}\}$ and $\{\Delta_{u,v}\}$ with $\{\tilde{\mathcal{G}}^{k,t}\}$ and $\{\tilde{\Delta}_{u,v}\}$ if
       we have $|\{\tilde{\Delta}_{u,v}\}| < |\{\Delta_{u,v}\}|$;
18     $I = I + 1$;
19 end
20 return $\{\mathcal{G}^{k,t}\}$ and $\{\Delta_{u,v}\}$;
```

connections ($\{\Delta_{u,v}\}$) is empty or the iteration limit I_m is reached. Hence, the algorithm will terminate to return the sub-topologies $\{\mathcal{G}^{k,t}\}$ with the smallest to-be-added connections. Specifically, in each iteration, we first mark to remove a connection between each connected pod pair u-v in each sub-topology to create the opportunity for sub-topology re-organization (*Lines* 3-5). Then, we update the to-be-added connections in *Line* 6. The iterations in *Lines* 8-16 add to-be-added connections to the sub-topologies in $\{\tilde{\mathcal{G}}^{k,t}\}$ sequentially according to the metric $\xi_{k,t}$ of each sub-topology. Specifically, we try to find the most connections in $\{\tilde{\Delta}_{u,v}\}$ that can be added to each sub-topology, by solving a max-flow problem in *Line* 10, and then update the metric $\xi_{k,t}$ of sub-topology $\mathcal{G}^{\star k,t}$ accordingly (*Line* 11). Then, we choose the sub-topology with the largest $\xi_{k,t}$ to apply changes and update $\mathcal{G}^{\star k,t}$, $\{\tilde{\Delta}_{u,v}\}$ and $\{\Delta'_{u,v}\}$ accordingly in *Lines* 13-14. Next, we update $\{\mathcal{G}^{k,t}\}$ and $\{\Delta_{u,v}\}$ with $\{\tilde{\mathcal{G}}^{k,t}\}$ and $\{\tilde{\Delta}_{u,v}\}$, if the number of to-be-added connections in $\{\tilde{\Delta}_{u,v}\}$ is smaller than that of $\{\Delta_{u,v}\}$ (*Line* 17), and update the number of iterations I (*Line* 18).

2) Algorithm to Check and Update Sub-topologies: Algorithms 3-5 show the three scenarios that we design for the sub-procedure in *Line* 18 of *Algorithm* 1. First, *Algorithm* 3 determines the metric of each sub-topology based on the number of rewirings associated with it [41]. For each sub-topology $\mathcal{G}^{k,t}$, we first initialize the related variables in *Line* 2, where $\tilde{\mathcal{G}}^{k,t}$ is introduced to record the hypothetical changes to $\mathcal{G}^{k,t}$. *Lines* 3-6 find the to-be-removed and to-be-added connections for $\mathcal{G}^{k,t}$, and store them in $\mathcal{G}_-^{k,t}$ and $\mathcal{G}_+^{k,t}$, respectively. Then, we recheck the sub-topology to add back the connections that were removed in *Line* 3 but have no conflict with the newly-added connections (*i.e.*, their removals are not necessary), and update $\mathcal{G}_-^{k,t}$ accordingly (*Line* 7). Finally, the metric $\xi_{k,t}$ of sub-topology $\mathcal{G}^{k,t}$ is obtained as the difference between its to-be-added connections and its to-be-removed ones (*Line* 8).

Algorithm 3: Scenario for Minimizing Rewirings

```
1  for each sub-topology in $\{\mathcal{G}^{k,t}\}$ do
2      $\tilde{\mathcal{G}}^{k,t} = \mathcal{G}^{k,t}$, $\mathcal{G}_-^{k,t} = \mathcal{G}_+^{k,t} = \emptyset$;
3      try to remove as many connections in $\{\Delta^*_{u,v}\}$ from
       $\tilde{\mathcal{G}}^{k,t}_{u,v}$ as possible, and add the removed ones in $\mathcal{G}_-^{k,t}$;
4      for each $\Delta_{u,v}$ in $\{\Delta_{u,v}\}$ in sorted order do
5          add the connections that can be added between $u$
           and $v$ in $\tilde{\mathcal{G}}^{k,t}$, and insert them in $\mathcal{G}_+^{k,t}$;
6      end
7      check $\tilde{\mathcal{G}}^{k,t}$ to add back the connections in $\{\Delta^*_{u,v}\}$
       that have no conflict with newly-added connections,
       and update $\mathcal{G}_-^{k,t}$ accordingly;
8      $\xi_{k,t} = |\mathcal{G}_+^{k,t}| - |\mathcal{G}_-^{k,t}|$;
9  end
10 return $\{\xi_{k,t}\}$, $\{\mathcal{G}_-^{k,t}\}$, $\{\mathcal{G}_+^{k,t}\}$;
```

Algorithm 4 explains the scenario that determines the metric of each sub-topology based on the volume of traffic rerouting associated with it. It uses a similar procedure of *Algorithm* 3, except for *Lines* 8-12, which compute each sub-topology's metric based on the number of its to-be-added connections and sum of bandwidth utilization on its to-be-removed connections.

The last scenario aims to simplify the topology changes in subsequent hitless reconfiguration, as shown in *Algorithm* 5. Here, *Lines* 1-7 are similar to their counterparts in *Algorithm* 4. Then, the for-loop of *Lines* 9-16 is introduced to simplify topology changes. Specifically, for each to-be-removed connection $e \in \mathcal{G}_-^{k,t}$, we first get its bandwidth usage ζ (*Line* 10) and update the total bandwidth utilization on removed connections (*Line* 11). Next, we calculate paths within the hop-count limit h and record the available bandwidth for each flow on the paths in the current physical topology after applying the changes in $\tilde{\mathcal{G}}^{k,t}$ (*Lines* 13). In *Line* 14, we update the required bandwidth of the flows on removed connections and the available bandwidth on the paths. Finally, the metric of each sub-topology is calculated based on the number of its to-be-added connections, total bandwidth utilization on its to-be-removed connections, and the ratio of available bandwidth, after removing the to-be-removed connections, to the total bandwidth required by the flows on those connections (*Line* 17).

Algorithm 4: Scenario for Minimizing Traffic Rerouting

```
1  for each sub-topology in $\{\mathcal{G}^{k,t}\}$ do
2      $\tilde{\mathcal{G}}^{k,t} = \mathcal{G}^{k,t}$, $\mathcal{G}_-^{k,t} = \mathcal{G}_+^{k,t} = \emptyset$, $m = 0$;
3      try to remove as many connections in $\{\Delta^*_{u,v}\}$ from
       $\tilde{\mathcal{G}}^{k,t}_{u,v}$ as possible, and add the removed ones in $\mathcal{G}_-^{k,t}$;
4      for each $\Delta_{u,v}$ in $\{\Delta_{u,v}\}$ in sorted order do
5          add the connections that can be added between $u$
           and $v$ in $\tilde{\mathcal{G}}^{k,t}$, and insert them in $\mathcal{G}_+^{k,t}$;
6      end
7      check $\tilde{\mathcal{G}}^{k,t}$ to add back the connections in $\{\Delta^*_{u,v}\}$
       that have no conflict with newly-added connections,
       and update $\mathcal{G}_-^{k,t}$ accordingly;
8      for each to-be-removed connection in $\mathcal{G}_-^{k,t}$ do
9          get bandwidth utilization on the connection as $\zeta$;
10         $m = m + \zeta$;
11     end
12     $\xi_{k,t} = |\mathcal{G}_+^{k,t}| - m$;
13 end
14 return $\{\xi_{k,t}\}$, $\{\mathcal{G}_-^{k,t}\}$, $\{\mathcal{G}_+^{k,t}\}$;
```

Algorithm 5: Scenario for Simplifying Topology Changes

```
1  for each sub-topology in $\{\mathcal{G}^{k,t}\}$ do
2      $\tilde{\mathcal{G}}^{k,t} = \mathcal{G}^{k,t}$, $\mathcal{G}_-^{k,t} = \mathcal{G}_+^{k,t} = \emptyset$, $m = 0$;
3      try to remove as many connections in $\{\Delta^*_{u,v}\}$ from
       $\tilde{\mathcal{G}}^{k,t}_{u,v}$ as possible, and add the removed ones in $\mathcal{G}_-^{k,t}$;
4      for each $\Delta_{u,v}$ in $\{\Delta_{u,v}\}$ in sorted order do
5          add the connections that can be added between $u$
           and $v$ in $\tilde{\mathcal{G}}^{k,t}$, and insert them in $\mathcal{G}_+^{k,t}$;
6      end
7      check $\tilde{\mathcal{G}}^{k,t}$ to add back the connections in $\{\Delta^*_{u,v}\}$
       that have no conflict with newly-added connections,
       and update $\mathcal{G}_-^{k,t}$ accordingly;
8      $b = 0$, $d = 0$;
9      for each to-be-removed connection $e \in \mathcal{G}_-^{k,t}$ do
10         get bandwidth utilization on connection $e$ as $\zeta$;
11         $m = m + \zeta$;
12         for each flow $r$ using connection $e$ do
13             try to find paths within $h$ hops for flow $r$ in
               current physical topology $\{D^{k,t}_{u,v}\}$ after
               applying changes in $\tilde{\mathcal{G}}^{k,t}$, and record available
               bandwidth $b_r$ and required bandwidth $d_r$ of $r$;
14             $b = b + b_r$, $d = d + d_r$ ;
15         end
16     end
17     $\xi_{k,t} = |\mathcal{G}_+^{k,t}| - m + \alpha \cdot \frac{b}{d}$;
18 end
19 return $\{\xi_{k,t}\}$, $\{\mathcal{G}_-^{k,t}\}$, $\{\mathcal{G}_+^{k,t}\}$;
```

C. Algorithm to Realize Hitless Reconfiguration

After determining the target physical topology, we propose *Algorithm* 6 to get a series of intermediate physical topologies to realize the hitless reconfiguration to the target physical topology with the smallest number of stages (*i.e.*, solving *Step* 2 of TPE design). *Line* 1 is for the initialization. We first attempt to directly use the available free ports to set up to-be-add connections, making it easier to reroute flows later (*Line* 2). If the target physical topology can be achieved by adding connections directly, the reconfiguration is complete

Algorithm 6: Hitless Reconfiguration in Stages for TPE

```
Input: $\{S^{k,t}_{u,v}\}$, $\{D^{k,t}_{u,v}\}$, $\mathcal{R}$, $\eta_{th}$, and $h$.
Output: Number of required stages $S$.
1  $\{M^{k,t}_{u,v}\} = \{S^{k,t}_{u,v}\}$, $S = 0$;
2  try to set up as many to-be-added connections in $\{D^{k,t}_{u,v}\}$
   as possible, and update $\{M^{k,t}_{u,v}\}$;
3  if $\{M^{k,t}_{u,v}\} = \{D^{k,t}_{u,v}\}$ then
4      $S = S + 1$;
5  else
6      while $\{M^{k,t}_{u,v}\} \neq \{D^{k,t}_{u,v}\}$ do
7          $\eta = 1$;
8          for each sub-topology in $\{M^{k,t}_{u,v}\}$ do
9              for each pod pair u-v in the sub-topology do
10                 find to-be-removed connections by
                   comparing $\{M^{k,t}_{u,v}\}$ and $\{D^{k,t}_{u,v}\}$;
11                 select a connection $e$ based on the flow
                   distribution of connections;
12                 if $\eta > \eta_{th}$ then
13                     mark ports of $e$ as unavailable;
14                     $flag = 0$;
15                     for each flow $r$ on connection $e$ do
16                         try to find path(s) for $r$ prioritizing
                           the connections that will not be
                           removed, and have fewer hops
                           and lower available bandwidth;
17                         if the path(s) can be found then
18                             reroute flow $r$ with path(s);
19                             update DCN status;
20                         else
21                             restore routing of flows on $e$;
22                             $flag = 1$, break;
23                         end
24                     end
25                     if $flag = 0$ then
26                         remove $e$ from $\{M^{k,t}_{u,v}\}$, and
                           update $\eta$ and DCN status;
27                     end
28                 end
29             end
30         end
31         try to set up as many to-be-added connections in
           $\{D^{k,t}_{u,v}\}$ as possible, and update $\{M^{k,t}_{u,v}\}$;
32         $S = S + 1$;
33     end
34 end
```

(*Lines* 3-4). Otherwise, we proceed to the while-loop *Lines* 6-33 to gradually move the current physical topology ($\{M^{k,t}_{u,v}\}$) toward the target one ($\{D^{k,t}_{u,v}\}$) iteratively. In each iteration (each stage), *Line* 7 is for the initialization, where η records the residual capacity ratio of the current stage. Then, the for-loop of *Lines* 8-30 processes the to-be-removed connections. *Lines* 10-11 select a to-be-removed connection e. Before removing the connection e, we greedily reroute all the flows on it onto alternative paths, prioritizing the connections that will not be removed, have fewer hops and lower available bandwidth (*Lines* 15-19), and if this can be done, we remove e and update the network status (*Lines* 25-27), and revert all the rerouting done for the flows on e (*Lines* 21-22), otherwise. Finally, we use the ports freed by the removed connections to set up as many to-be-added connections as possible to move $\{M^{k,t}_{u,v}\}$ toward $\{D^{k,t}_{u,v}\}$, and proceed to the next stage (*Lines* 31-32).

D. Complexity Analysis

As *Algorithms* 3-5 are called in *Algorithms* 1 and 2, we discuss them first. The time complexity of *Algorithms* 3 and 4 are both $O(T \cdot K \cdot |V|^2)$, the complexity of *Algorithm* 5 is $O(T \cdot K \cdot |V|^{1+h} \cdot |R|)$, where the hop-count limit h is a constant and its value is usually relatively small (*e.g.*, 2 or 3) [8]. *Algorithm* 2 is a sub-procedure of *Algorithm* 1, and its complexity is $O(I_m \cdot K^2 \cdot T^2 \cdot (\mathcal{O} + |V|^3))$, where $\mathcal{O}$ is the complexity of one among *Algorithms* 3-5. As the complexity of other procedure in *Algorithm* 1 is much lower than that of *Algorithm* 2, the complexity of *Algorithm* 1 can be approximated as $O(I_m \cdot K^2 \cdot T^2 \cdot (\mathcal{O} + |V|^3))$. The complexity of *Algorithm* 6 is $O(|R| \cdot (|R| + |V| \cdot \log|V| + T \cdot K \cdot |V|) \cdot T^2 \cdot K^2 \cdot |V|^3)$.

V. Performance Evaluation

In this section, we evaluated the performance of our proposed algorithms with numerical simulations.

A. Simulation Setup

To ensure the service availability during switch failures, we made each pod connect to all the OCS-based switch groups, and assumed that the ports of each OCS-based switch group are evenly distributed to the pods to interconnect them [8]. The simulations considered two scenarios, *i.e.*, the small-scale and large-scale ones. In the small-scale simulations, we made sure that the problem was small-sized such that the MILP in Section III-C could be solved within reasonable time. Specifically, we considered one or two OCS-based switch groups, each composed of 1, 2 or 4 switches. The number of ports of each OCS-based switch could be $\{12, 18, 24\}$, and the number of pods varied in $[4, 8]$. The large-scale simulations addressed two or four OCS-based switch groups, with the number of switches in each group being $\{1, 2, 4, 6\}$. The number of ports on each switch varied in $[32, 512]$, while the number of pods varied in $[16, 256]$. In both scenarios, the pods were equipped with 100GbE optical ports. We made the inter-pod traffic in $\mathcal{R}$ consume 10%-50% of the DCN's total capacity, and changed the number of elements in $\mathcal{R}$ within $[1, |V|^2 - |V|]$.

The simulations assessed four schemes for TPE with hitless reconfiguration: 1) our MILP model formulated in Section III-C, and 2) Adapted-Gemini, 3) Min-Rerouting, and 4) Simp-TPChange presented in Section IV. The simulations run on a computer with 2.2 GHz Intel Xeon Silver 4210 CPU and 128 GB memory, and the software environment was PyCharm-2023.3.4 with Gurobi 9.5.1 [42] and MATLAB 2020b.

TABLE I
RESULTS OF SMALL-SCALE SIMULATIONS

	MILP		Adapted-Gemini			Min-Rerouting			Simp-TPChange		
$\|V\|$	S^*	Running Time (s)	S	Running Time (s)	δ (%)	S	Running Time (s)	δ(%)	S	Running Time (s)	δ (%)
4	1	0.080	1	0.004	0	1	0.004	0	1	0.005	0
6	1	738.630	1.7	0.020	70	1	0.008	0	1	0.013	0
8	1.3	4209.475	2.7	0.058	179	1.7	0.138	30.5	1.3	0.028	0

B. Small-Scale Simulations

We first conducted small-scale simulations to compare the performance of the algorithms, where the longest running time of MILP was set as three hours and the source and destination of each inter-pod flow in $\mathcal{R}$ were chosen randomly. Here, we denote the optimal result on reconfiguration stages from MILP as S^* and the corresponding result from a heuristic as S, and then the performance gap of the heuristic becomes,

$$\delta = \frac{S - S^*}{S^*}. \tag{24}$$

The simulation results are shown in Table I, where each value was obtained by averaging the results from 3 independent runs. In all the runs, the algorithms were able to obtain physical topologies that realize the target logical topologies exactly. From Table I, we can see that MILP always provides the fewest reconfiguration stages but it is very time-consuming, taking more than one hour to solve the joint optimization for the instance of 8 pods. All the three heuristics yield the same optimal solutions as output by MILP in the 4-pod scenario. However, as the number of pods increases, Adapted-Gemini fails to achieve the optimal performance. This indicates that simply minimizing rewiring in the first step of TPE does not necessarily reduce the number of reconfiguration stages because it does not address the impact of traffic rerouting. By minimizing traffic rerouting, Min-Rerouting still secures the optimal performance in the case of 6 pods, but its gap to MILP soars to 30.5% when the number of pods is further increased to 8. Simp-TPChange performs the best among the three heuristics and obtains solutions same to those from MILP consistently. Overall, the results suggest that the joint optimization of the two steps of TPE is beneficial and simplifying topology changes in the first step can accelerate the hitless reconfiguration better than the other two approaches.

We also investigated the impact of η_{th} (*i.e.*, the threshold on residual capacity ratio of each stage) on reconfiguration stages. Fig. 5 shows the results obtained when the number of pods and bandwidth usage are set to be 8 and 0.5, respectively. Again, Simp-TPChange performs the best in all settings. The number of reconfiguration stages increases with η_{th}, because a larger η_{th} reserves more bandwidth during each reconfiguration stage, making it harder to finish hitless reconfiguration toward the target physical topology within fewer stages.

C. Large-Scale Simulations

We then run large-scale simulations to further assess the proposed heuristic algorithms. Since the algorithms might not always realize the target logical topologies exactly, we quantify the distances between the obtained and target logical topologies by calculating the ratios of the number of missing connections in the obtained logical topologies to the total number of connections in the target ones. As shown in Fig. 6, the logical topologies obtained by the three algorithms have, on average, less than 5% missing connections compared with the target ones. The results manifest that the proposed algorithms can still obtain physical topologies to closely approximate target logical topologies even in large-scale settings.

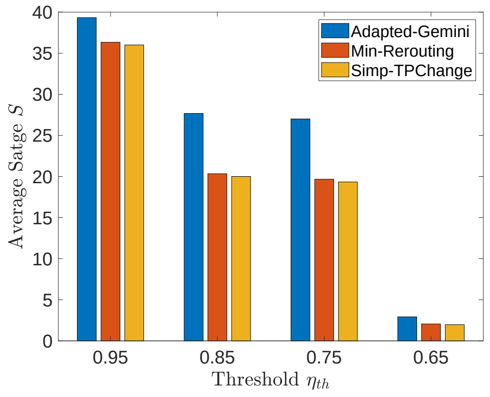


Fig. 5. Impact of threshold η_{th} on reconfiguration stages from algorithms.

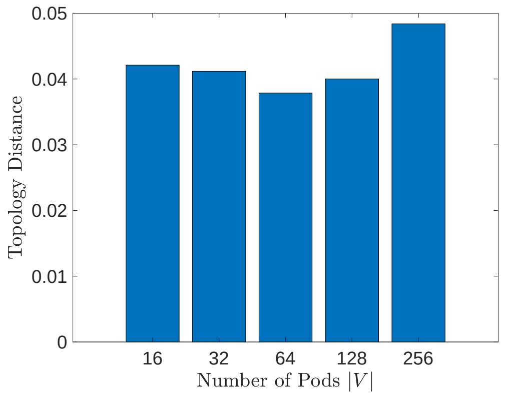


Fig. 6. Performance of approximating logical topologies with physical ones.

Fig. 7 shows the numbers of reconfiguration stages required by the three algorithms. The results coincide with those obtained from the small-scale simulations, showing that Simp-TPChange consistently outperforms the others while Adapted-Gemini requires the most stages. This confirms again the

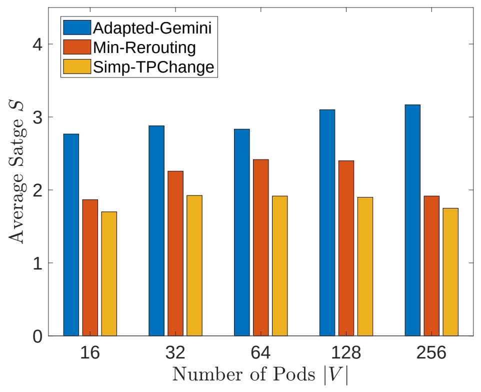


Fig. 7. Reconfiguration stages from heuristics in large-scale simulations.

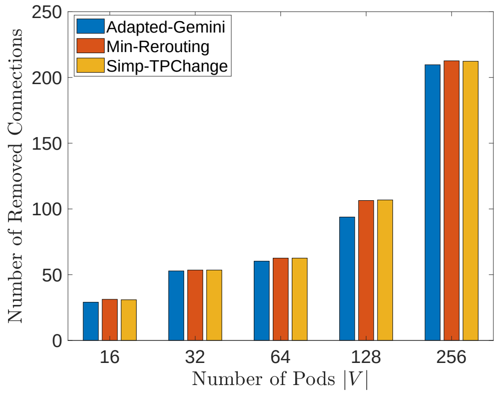


Fig. 8. Number of connections removed by algorithms.

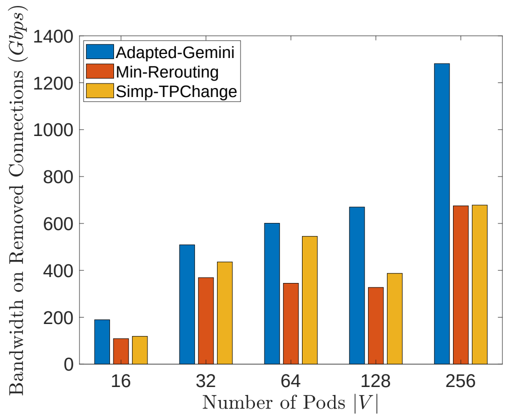


Fig. 9. Total bandwidth usage on connections removed by algorithms.

superiority of the proposed topology simplification strategy over minimum rewiring. We also notice that the number of reconfiguration stages does not vary evidently as the scale of DCNs enlarges. This phenomenon can be attributed to the fact that the number of reconfiguration stages is influenced primarily by the availability of network resources for traffic rerouting (aside from η_{th}), which is less correlated with the number of pods. We have justified the impact of η_{th} in Section V-B, and we will validate the presumption later by evaluating how bandwidth availability affects the number of reconfiguration stages.

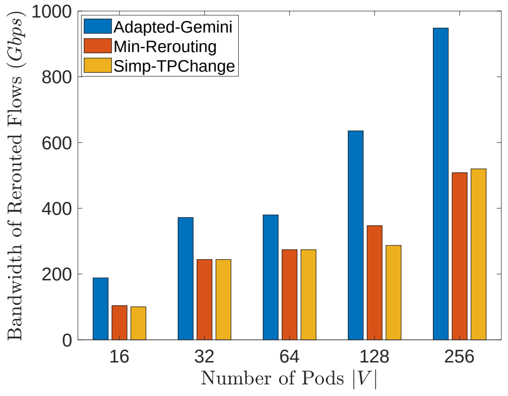


Fig. 10. Total bandwidth of traffic rerouted by algorithms.

To shed light on the rationale behind the performance gaps between the heuristics, we counted more statistics, including the number of removed connections, the total bandwidth utilization on removed connections, and the volume of rerouted traffic during hitless reconfigurations, and present their results in Figs. 8-10, respectively. In Fig. 8, the number of removed connections from Adapted-Gemini is slightly smaller than those of Min-Rerouting and Simp-TPChange. This is because Adapted-Gemini only tries to minimize rewiring connections in calculating target physical topologies, whereas Min-Rerouting and Simp-TPChange favor more reconfigurations (thus, get closer to the target logical topologies) by also considering the impact of bandwidth usage and traffic rerouting of the removed connections. For the same reason, Figs. 9-10 show better results on the other statistics from Min-Rerouting and Simp-TPChange. Besides, as Min-Rerouting pays attention to the bandwidth usage on removed connections when getting target physical topologies, it always gets the minimum bandwidth usage on removed connections.

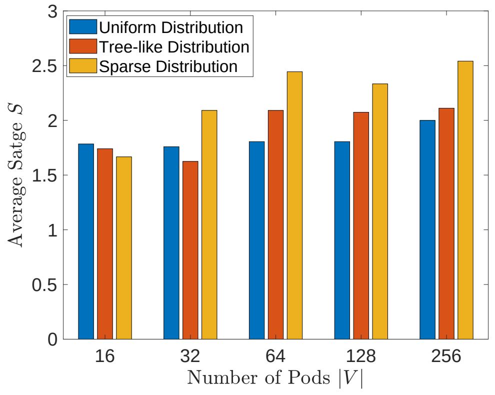


Fig. 11. Impact of traffic distribution on algorithms' performance.

Finally, we conducted simulations to study the impacts of different traffic characteristics on the performance of the three algorithms. We first assessed the algorithms with three traffic distributions, *i.e.*, uniform, tree-like, and sparse distributions. Fig. 11 shows the results on number of reconfiguration stages

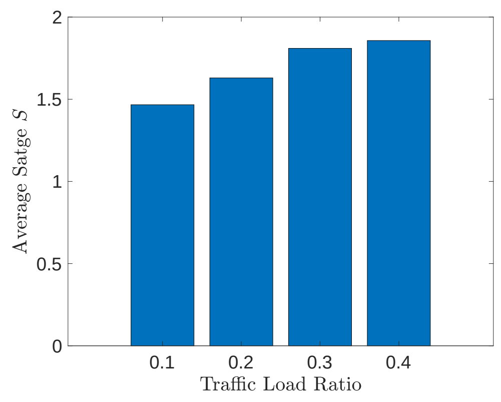


Fig. 12. Impact of traffic load ratio on algorithms' performance.

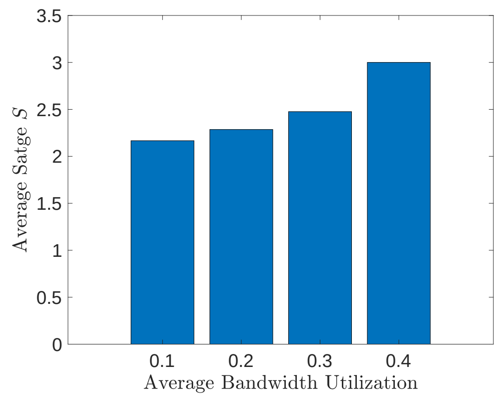


Fig. 13. Impact of bandwidth utilization on algorithms' performance.

when the traffic in $\mathcal{R}$ follows the distributions, where each bar denotes the average of the results from the three algorithms. We can see that the sparse distribution leads to more reconfiguration stages in most of the cases. This is because under this distribution, inter-pod traffic is rather dispersed and thus removing connections will likely implicate rerouting of more flows, causing more conflicts between the connections to be removed. In contrast, the tree-like distribution features a traffic matrix with flows concentrated on only a small portion of pods with fewer scattered flows. Therefore, less traffic rerouting and in turn fewer stages will be involved during reconfigurations. As for the uniform distribution, since traffic is evenly spread, more flows can be rerouted within one stage. This enables the removal of more connections, which frees up additional ports for establishing new connections in the same stage, and hereby, reduces the number of reconfiguration stages required.

Using the ratio of the number of pod pairs that carry traffic to the total number of pod pairs as the measure of traffic load ratio, we show the impact of traffic load on the number of reconfiguration stages in Fig. 12, where each bar is still for the average result of the algorithms. Clearly, the number of reconfiguration stages increases with the traffic load ratio. This is because more actively-communicating pods result in more traffic rerouting during reconfiguration, restricting the number of connections that can be concurrently removed in each stage. Similar to the observations drawn from Fig. 12, when we change traffic load by adjusting the average bandwidth utilization in the DCN, Fig. 13 indicates that increasing traffic load leads to more reconfiguration stages. The reason is also similar, as a larger bandwidth usage implies more traffic to be rerouted, and thereby, fewer connections can be removed in parallel due to the limited bandwidth availability.

VI. Conclusion

This work studied how to optimize the two steps of TPE jointly to efficiently accelerate the hitless reconfiguration of an OCS-based DCN. We formulated an MILP to solve the problem exactly, and proposed an approach that can optimize TPE design greedily according to various metrics to minimize the number of stages required in hitless reconfiguration for TPE. Simulations verified the effectiveness of our proposals. Specifically, the results suggested that simply focusing on minimizing rewirings in the first step of TPE cannot accelerate the hitless reconfiguration in its second step effectively.

Acknowledgments

Xiaoliang Chen and Zuqing Zhu are the co-corresponding authors. This work was supported by NSFC project 62371432.

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] H. Liu *et al.*, "Lightwave Fabrics: At-scale optical circuit switching for datacenter and machine learning systems," in *Proc. of ACM SIGCOMM 2023*, pp. 499–515, Aug. 2023.
[3] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.
[4] 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.
[5] A. Greenberg *et al.*, "VL2: a scalable and flexible data center network," in *Prof. of ACM SIGCOMM 2009*, pp. 51–62, Aug. 2009.
[6] N. Jouppi *et al.*, "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proc. of ISCA 2023*, pp. 1–14, Jun. 2023.
[7] A. Singh *et al.*, "Jupiter Rising: A decade of Clos topologies and centralized control in Google's datacenter network," in *Proc. of ACM SIGCOMM 2015*, pp. 183–197, Aug. 2015.
[8] M. Zhang *et al.*, "Gemini: Practical reconfigurable datacenter networks with topology and traffic engineering," *arXiv preprint arXiv:2110.08374*, Oct. 2021. [Online]. Available: https://arxiv.org/abs/2110.08374.
[9] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 498–511, Mar. 2013.
[10] 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.
[11] 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.
[12] 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.
[13] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.
[14] W. Lu, Z. Zhu, and B. Mukherjee, "On hybrid IR and AR service provisioning in elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 4659–4669, Nov. 2015.
[15] P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," *J. Lightw. Technol.*, vol. 35, pp. 5335–5346, Dec. 2017.

[16] J. Zerwas *et al.*, "Duo: A high-throughput reconfigurable datacenter network using local routing and control," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 7, pp. 1–25, Mar. 2023.

[17] N. Farrington *et al.*, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proc. of ACM SIGCOMM 2010*, pp. 339–350, Aug. 2010.

[18] G. Wang *et al.*, "c-Through: Part-time optics in data centers," in *Proc. of ACM SIGCOMM 2010*, pp. 327–338, Aug. 2010.

[19] L. Poutievski *et al.*, "Jupiter evolving: transforming Google's datacenter network via optical circuit switches and software-defined networking," in *Proc. of ACM SIGCOMM 2022*, pp. 66–85, Aug. 2022.

[20] H. Yang and Z. Zhu, "Traffic-aware configuration of all-optical data center networks based on Hyper-FleX-LION," *IEEE/ACM Trans. Netw., in Press*, 2024.

[21] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 63–74, Aug. 2008.

[22] J. Zerwas, W. Kellerer, and A. Blenk, "What you need to know about optical circuit reconfigurations in datacenter networks," in *Proc. of ITC 2021*, pp. 1–9, Aug. 2021.

[23] M. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," *J. Opt. Commun. Netw.*, vol. 12, pp. B44–B54, Feb. 2020.

[24] X. Chen *et al.*, "Machine-learning-aided cognitive reconfiguration for flexible-bandwidth HPC and data center networks," *J. Opt. Commun. Netw.*, vol. 13, pp. C10–C20, Jan. 2021.

[25] L. Schares *et al.*, "A reconfigurable interconnect fabric with optical circuit switch and software optimizer for stream computing systems," in *Proc. of OFC 2009*, pp. 1–3, Mar. 2009.

[26] J. Benjamin *et al.*, "PULSE: Optical circuit switched data center architecture operating at nanosecond timescales," *J. Lightw. Technol.*, vol. 38, pp. 4906–4921, May 2020.

[27] H. Ballani *et al.*, "Sirius: A flat datacenter network with nanosecond optical switching," in *Proc. of ACM SIGCOMM 2020*, pp. 782–797, Jul. 2020.

[28] G. Liu *et al.*, "Architecture and performance studies of 3D-Hyper-FleX-LION for reconfigurable All-to-All HPC networks," in *Proc. of SC 2020*, pp. 1–16, Nov. 2020.

[29] M. Khani *et al.*, "SiP-ML: high-bandwidth optical network interconnects for machine learning training," in *Proc. of ACM SIGCOMM 2021*, pp. 657–675, Aug. 2021.

[30] P. Cao *et al.*, "TROD: Evolving from electrical data center to optical data center," in *Proc. of ICNP 2021*, pp. 1–11, Dec. 2021.

[31] Y. Tang *et al.*, "Effectively reconfigure the optical circuit switching layer topology in data center network by OCBridge," *J. Lightw. Technol.*, vol. 37, pp. 897–908, Feb. 2019.

[32] 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.

[33] Z. Zhao, B. Guo, Y. Shang, and S. Huang, "Hierarchical and reconfigurable optical/electrical interconnection network for high-performance computing," *J. Opt. Commun. Netw.*, vol. 12, pp. 50–61, Mar. 2020.

[34] H. Yang and Z. Zhu, "Topology configuration scheme for accelerating coflows in a hyper-FleX-LION," *J. Opt. Commun. Netw.*, vol. 14, pp. 805–814, Sept. 2022.

[35] M. Teh, S. Zhao, P. Cao, and K. Bergman, "COUDER: robust topology engineering for optical circuit switched data center networks," *arXiv preprint arXiv:2010.00090*, 2020.

[36] Y. Zhao *et al.*, "Dynamic topology management in optical data center networks," *J. Lightw. Technol.*, vol. 33, pp. 4050–4062, Aug. 2015.

[37] W. Zheng, X. Chen, and Z. Li, "Fast and nondisruptive reconfiguration design for optical datacom networks," in *Proc. of PSC 2023*, pp. 1–3, Sept. 2023.

[38] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. of ACM SIGCOMM 2013*, pp. 447–458, Aug. 2013.

[39] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, 1979.

[40] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theor. Comput. Sci.*, vol. 1, no. 3, pp. 237–267, 1976.

[41] S. Zhao *et al.*, "Minimal rewiring: Efficient live expansion for Clos data center networks," in *Proc. of NSDI 2019*, pp. 221–234, Aug. 2019.

[42] "Gurobi." [Online]. Available: https://www.gurobi.com/.