Machine-Learning-Aided Cognitive Reconfiguration for Flexible-Bandwidth HPC and Data Center Networks [Invited]

Xiaoliang Chen[1,*], Roberto Proietti[1], Marjan Fariborz[1], Che-Yu Liu[2], and S. J. Ben Yoo[1,*]

[1]Department of Electrical and Computer Engineering, University of California, Davis, CA 95616 USA
[2]Department of Computer Science, University of California, Davis, Davis, CA 95616, USA
[*]Corresponding author: xlichen@ieee.org, sbyoo@ucdavis.edu

Compiled December 3, 2020

This paper proposes a machine learning-aided cognitive approach for effective bandwidth reconfiguration in optically interconnected datacenter/high-performance computing (HPC) systems. The proposed approach relies on a Hyper-X-like architecture augmented with flexible-bandwidth photonic interconnections at large scales using a hierarchical intra-/inter-POD photonic switching layout. We first formulate the problem of connectivity graph and routing scheme optimization as a mixed-integer linear programming model. A two-phase heuristic algorithm and a joint optimization approach are devised to solve the problem with low time complexity. Then, we propose a machine learning-based end-to-end performance estimator design to assist the network control plane with intelligent decision making for bandwidth reconfiguration. Numerical simulations using traffic distribution profiles extracted from HPC applications traces as well as random traffic matrices verify the accuracy performance of the ML design estimator ($<9\%$ error) and demonstrate up to $5\times$ throughput gain from the proposed approach compared with the baseline Hyper-X network using fixed all-to-all intra-/inter-POD interconnects.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

The rapid expansion of cloud services (e.g., web search, video streaming) and emerging high-performance computing (HPC) applications (e.g., deep learning workloads) entails advanced interconnect architectures supporting high-bandwidth, low-latency, and robust communications among tens of thousands of servers or computing nodes. Current hyper-scale data centers (DCs) and HPC systems are typically built on multi-hierarchy tree-based electronic packet switching (EPS) architectures with point-to-point optical links for inter-rack and inter-POD communications [1]. While these tree-based architectures provide rich interconnectivity, they suffer from high power consumptions and end-to-end latency as the systems scale to a large number of nodes (network diameter and number of switches to be traversed increases). As switch port data rates continue to increase (from 40G to 100G and beyond [2]), sustaining the scalability of such EPS-based hyperscale systems becomes increasingly challenging. In particular, the energy efficiency of electronics substantially degrades at such high data rates, and today's data centers and HPC systems are already consuming megawatts of power. Overall, scalability faces difficulties in both scaling up and scaling out due to the limitations in the bandwidth, radix, and switching capacity of the electronic switches.In this context, previous works have proposed a number of low-diameter directly connected topologies (e.g., Hyper-X [3], JellyFish [4], and Xpander [5]) and demonstrated improved performance over the tree-based architectures, with the first at-scale experimental demonstration of an Hyper-X HPC network reported in [6].

As coarse wavelength division multiplexing (CWDM) optical transceivers are penetrating the datacom market to sustain the ever increasing demand for compute node I/O bandwidth, [7], and as silicon-photonic (SiPh) technologies are becoming commercially viable, there has been a consensus in industry and academia that disruptive optical interconnect and switching technologies are essential for sustaining the scalability of cloud infrastructure networks [8]. In other words, there are opportunities to re-architect the DC/HPC systems leveraging photonic technologies not just for point-to-point interconnections but also for flattening the inter-rack networking architecture by direct inter-rack wavelength routing and switching. Following this consensus, the literature first reported several hybrid optical/electrical (HOE) switching architectures [9, 10], where optical circuit switching (OCS) fabrics are laid out to augment the traditional OPS structures for enabling transparent (and therefore, high-capacity and energy-efficient) switching of elephant flows. These hybrid architectures still cannot fully address the

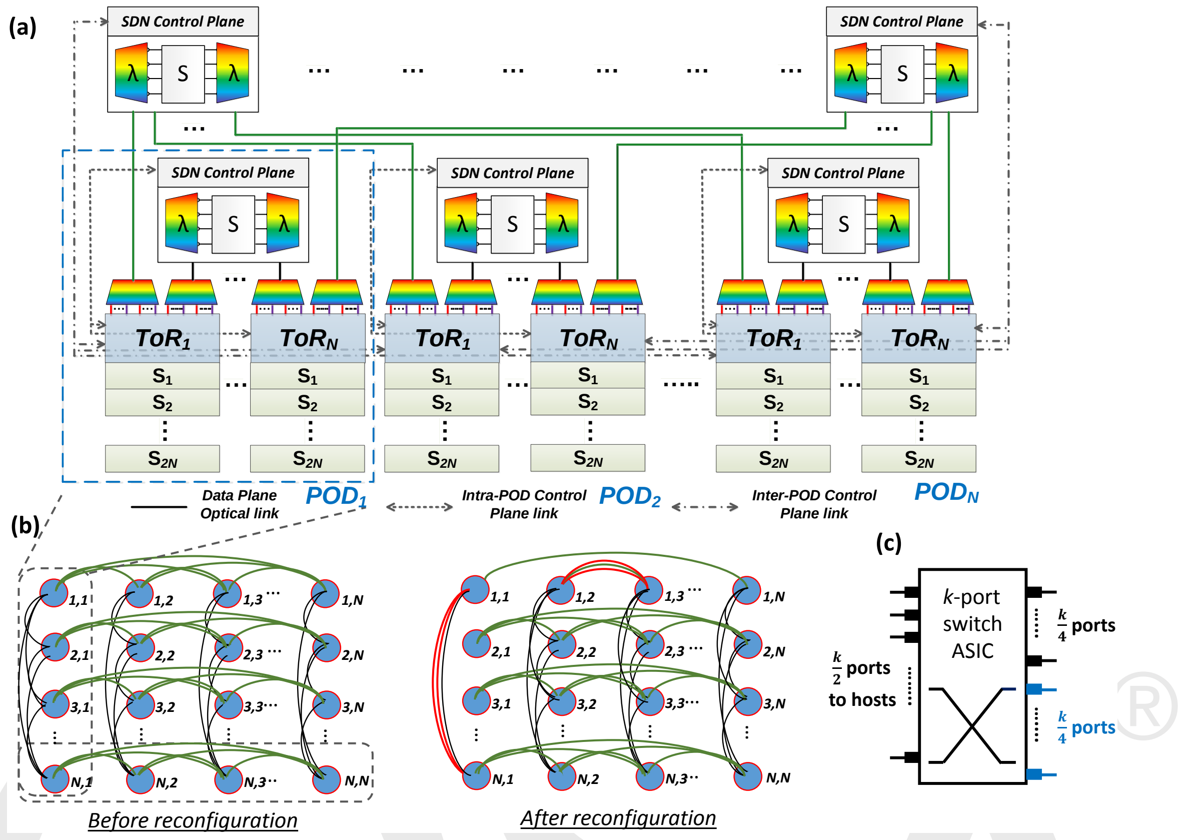

Fig. 1. (a) Directly connected architecture with space-wavelength selective switches for topology/bandwidth reconfiguration. There are $N = k/4$ PODs, with each PODs containing $2N$ servers and one k-port ToR. (b) Topological representation of the architecture before (left) and after (right) reconfiguration. Black links are intra-pod links. Green links are inter-pod links. Red links are intra or inter-pod links that have been reconfigured. (c) k-port ToR switch ASIC with $k/4$ ports for intra-POD interconnection and $k/4$ ports from inter-POD interconnection. The rest of the ports ($k/2$) are used for connection to the hosts.

scalability issue as they still largely rely on electronic switches and make use of microelectromechanical systems (MEMS)-based optical switches that are subject to slow switching speeds (milliseconds). Even more importantly, they require complex flow classification algorithms and a fully centralized control plane that need to orchestrate the switching operation across hundreds of top-of-rack (ToR) switches.

A different switching paradigm that has been considered in the past few years is the one where optical switching fabrics are used to dynamically adapt the inter-rack connectivity (also referred to as optical reconfiguration for bandwidth steering) bandwidth and topology to the traffic patterns. As the traffic distribution among racks and PODs is often non-uniform (often highly skewed) and evolves over time [11], optical reconfiguration can steer bandwidth resources where needed and ease link congestion due to hotspot traffic between specific source-destination racks. In this case, the switching operation does not need to be performed on a packet flow basis but whenever the traffic characteristic changes (e.g. different computation and communication phases of certain HPC applications). Nevertheless, we anticipate that even under this scenario, realizing effective and fast scheduling of reconfiguration is never a trivial task, which still requires a powerful control and management plane that can actively monitor the traffic among hundreds of ToR switches [9, 10, 12] and promptly react to traffic changes with fast and correct cross-layer reconfiguration operations of the optical and electronic switch layers. There have been several research efforts on optical reconfiguration design recently [13–17], which, will be discussed in Section 7. However, these works either just show proof-of-concept demonstrations with small-scale topologies [15, 16], or employ ML models [14]/simple heuristic designs [13, 17] targeting the reconfiguration of single OCS switches.

In this paper, we consider an Hyper-X-like optical interconnect architecture [3] that can provide scalable and flexible-bandwidth interconnections, and we benchmark its performance against a standard Hyper-X network. We propose a machine learning (ML)-aided cognitive approach for effective bandwidth reconfiguration under such an architecture. We first formulate the problem of connectivity graph and routing optimization as a mixed-integer linear programming (MILP) model. Two heuristic algorithms that optimize the connectivity graphs and routing schemes successively or jointly are developed for solving the problem in a time-efficient manner. We present an ML-based end-to-end performance (i.e., latency, packet loss rate) estimator design to assist in intelligent decision making for reconfiguration scheduling. Numerical results obtained using traces from existing HPC applications and under random traffic matrices verify the effectiveness of the proposed design.

The rest of the paper is organized as follows. In Section 2, we describe the 2D-Hyper-X architecture and discuss the control plane principle for supporting dynamic optical reconfiguration. Section 3 gives the MILP formulation. We elaborate on the heuristic and the ML designs in Section 4 and Section 5, respectively. In Section 6, we present the simulation results and the related discussions. Afterward, we discuss the related works in Section 7. Finally, Section 8 concludes the paper with remarks on potential future research directions and challenges yet to be resolved.

2. ARCHITECTURE

As discussed in Section 1, in this paper we consider a directly connected network where each switch connects to a certain number of servers (or computing nodes). Fig. 1(a) shows an architecture with $N = k/4$ clusters (often referred as portable data centers - PODs) with N ToR switches per POD. The intra-POD and inter-POD connectivity is augmented with wavelength-space selective optical switches [18, 19] to reconfigure the interconnect and increase the bandwidth (number of links between specific ToR pairs) based on the algorithms and policies discussed in Section 4.

As shown in Fig. 1(b) on the left, the ToRs can be seen as organized in columns, where each column represents a POD. The default connectivity inside a POD and between PODs (in each row) is all-to-all. This configuration is known as 2D-Hyper-X architecture [3, 6], where the ToR switches are fully connected in each dimension. While the standard Hyper-X configuration in Fig. 1(b) (on the left) is ideal for traffic that is evenly distributed across the ToRs and PODs, in the case of uneven traffic distributions with hotspots, it can be desirable to break the all-to-all connectivity to temporary give more bandwidth (links) where needed. This is shown in Fig. 1(b)-right where the red links represent the links/bandwidth that have been steered from other ToRs.

As shown in Fig. 1(c), each ToR switch uses $2N$ ports for servers (intra-rack) and $2N$ ports for inter-rack. This means that the oversubscription of the network is 1 : 1. Note that, different oversubscription ratios (e.g., 1 : 3 [20]) can be obtained within the same architecture by allocating more or fewer ToR ports for connections to the intra-rack servers. This is a design choice that represents a trade-off between costs and performance [1, 21]. We chose 1 : 1 as it allows to provide full bisection bandwidth and it is a common design choice in other recent studies as well as [6]. Among the $2N$ inter-rack ports, N are used for intra-POD and N are used for inter-POD. WDM TRXs with N wavelengths are used for inter-rack ports. The radix of the optical switches is N. If we assume state-of-the-art switch ASICs with $k = 4N = 128$ ports at 100 Gb/s, this 2D Hyper-X architecture can scale to $k^3/32 = 2N^3 = 65,536$ servers (more than what is required for Exascale computing [22]) while requiring optical switches with limited radix ($N = k/4 = 32$) and number of wavelengths ($N = k/4 = 32$). In terms of space-wavelength selective optical switch technology, we are not restricting this study to any specific technology implementation, but we are simply assuming to use a space-wavelength switch that can scale to $N = k/4 = 32$ ports and uses $N = k/4 = 32$ wavelengths at each input.

A key aspect of any application of optical switching is the control plane to orchestrate the cross-layer switching operations in the optical and electrical domains. As shown in Fig. 1(a), each switch has a software-defined networking (SDN) controller that connects to a subset of N ToRs. Each controller is responsible for running the reconfiguration algorithm (see Section 4), computing the new optical switch state, and updating the routing information in the ToRs. Fig. 2 sketches the layout of modular control plane functions for a POD. An SDN controller communicates with the ToR and optical switches using SDN protocols to

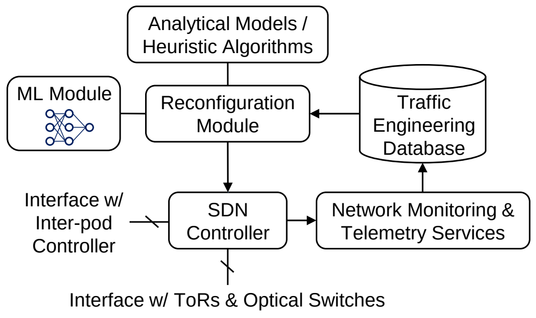
Analytical Models / Heuristic Algorithms
ML Module
Reconfiguration Module
Traffic Engineering Database
Interface w/ Inter-pod Controller
SDN Controller
Network Monitoring & Telemetry Services
Interface w/ ToRs & Optical Switches

Fig. 2. Layout of modular control plane functions for a POD.

Table 1. Decision variables used by the MILP model.

Variable	Description
$x_{u,v}$	number of ports configured for the direct communication from ToR u to v;
$p_{u,v}^{u',v'}$	Boolean, equals to 1 if the routing path from ToR u to v uses link u' to v', and 0 otherwise;
$b_{u,v}$	transmission capacity (in Gbps) allocated to the communication from ToR u to v.
$b_{u,v}^{u',v'}$	transmission capacity allocated on link u' to v' to the communication from ToR u to v.

collect network state data (e.g., traffic matrix [9, 10]) specified by the network monitoring and telemetry service module and distribute flow tables or configuration commands. The controller also interacts with the inter-POD controller for global bandwidth steering optimization. The traffic engineering database stores network state data related to switch state, routing scheme, and traffic distribution. Such data are constantly accessed by the reconfiguration module, which can employ both ML approaches and the traditional analytical models or heuristic algorithms to calculate reconfiguration scheduling for the controller.

3. PROBLEM FORMULATION

Let $V = \{V_i|_{i\in[1,N]}\}$ denote the set of N PODs in a bandwidth-reconfigurable datacenter/HPC system. Each POD V_i consists of $|V_i|$ ToR switches and the total number of ToR is $|V| = \sum_i |V_i|$. Every ToR $u \in V_i$ is equipped with $|V_i| - 1$ and $N - 1$ ports working at C Gbps for intra- and inter-POD communications, respectively. The traffic dynamicity of the system can be characterized by a demand matrix $D(t)$ evolving over time t (depending on the profile of the applications running), where $D_{u,v}(t) \in D(t)$ $(u, v \in V)$ represents the traffic demand in Gbps from ToR u to v. Our goal is to determine a connectivity graph/topology G for V (i.e., the interconnectivity of the ToR ports) and a set of routing schemes P at each time t such that the overall system performance (e.g., throughput, end-to-end latency) is optimized.

We formulate the optimization problem as an MILP model. Table 1 summarizes the decision variables used by the model.
Objective:

$$Maximize \quad z = \alpha_1 \cdot z_1 - \alpha_2 \cdot z_2 + \alpha_3 \cdot z_3. \tag{1}$$

$$z_1 = \frac{1}{\sum\limits_{u,v} D_{u,v}(t)} \cdot \sum_{u,v\in V} b_{u,v}. \tag{2}$$

$$z_2 = \frac{1}{|V|(|V|-1)} \cdot \sum_{u,v,u',v'} p_{u,v}^{u',v'}. \tag{3}$$

$$z_3 \leq \frac{1}{C}\cdot\left(x_{u',v'}\cdot C - \sum_{u,v} b_{u,v}^{u',v'}\right), \quad \forall u',v' \in V. \tag{4}$$

The objective is to maximize the normalized system throughput z_1 (i.e., the demand satisfaction ratio), while 1) minimizing the average path length z_2 (in terms of the number of hops) and 2) balancing the link loads. α_1, α_2, and α_3 are positive weighting factors. Here, load balancing is achieved by maximizing the minimal normalized link margin z_3 constrained by Formula 4. By optimizing z_2 and z_3, we aim at reducing the end-to-end communication latency and packet loss rate.

Constraints:

$$x_{u,v} \geq 0, \quad \forall u,v \in V. \tag{5}$$

$$\sum_{v\in V_i\setminus u} x_{u,v} \leq |V_i|-1, \quad \forall u \in V_i, V_i \in V. \tag{6}$$

$$\sum_{v\in V\setminus V_i} x_{u,v} \leq N-1, \quad \forall u \in V_i, V_i \in V. \tag{7}$$

$$\sum_{u\in V_i\setminus v} x_{u,v} \leq |V_i|-1, \quad \forall v \in V_i, V_i \in V. \tag{8}$$

$$\sum_{u\in V\setminus V_i} x_{u,v} \leq N-1, \quad \forall v \in V_i, V_i \in V. \tag{9}$$

Formula 5 ensures that $x_{u,v}$ is non-negative. Formulas 6-9 restrict the numbers of ports each ToR can use for intra- and inter-POD communications to be $|V_i|-1$ and $N-1$, respectively.

$$p_{u,v}^{u',v'} \leq x_{u',v'}, \quad \forall u,v,u',v' \in V. \tag{10}$$

Formula 10 ensures that traffic can only be routed on valid links, i.e., on $u' \rightarrow v'$, $x_{u',v'} > 0$.

$$\sum_{v'\in V} p_{u,v}^{u',v'} - \sum_{v'\in V} p_{u,v}^{v',u'} = \begin{cases} 1, & u'=u, \\ -1, & u'=v, \\ 0, & u' \in V\setminus\{u,v\}, \end{cases} \quad \forall u,v \in V. \tag{11}$$

$$\sum_{v'\in V} p_{u,v}^{u',v'} \leq 1, \quad \forall u,v \in V. \tag{12}$$

$$\sum_{v'\in V} p_{u,v}^{v',u'} \leq 1, \quad \forall u,v \in V. \tag{13}$$

Formulas 11-13 are flow conservation constraints [23], making sure that a single routing path is used for each ToR pair.

$$0 \leq b_{u,v} \leq D_{u,v}(t), \quad \forall u,v \in V. \tag{14}$$

Formula 14 ensures that the capacities allocated are bounded by 0 and the actual demands.

$$b_{u,v}^{u',v'} \leq b_{u,v} + (1-p_{u,v}^{u',v'})\cdot M, \quad \forall u,v,u',v' \in V. \tag{15}$$

$$b_{u,v}^{u',v'} \geq b_{u,v} - (1-p_{u,v}^{u',v'})\cdot M, \quad \forall u,v,u',v' \in V. \tag{16}$$

$$b_{u,v}^{u',v'} \leq p_{u,v}^{u',v'}\cdot M, \quad \forall u,v,u',v' \in V. \tag{17}$$

$$b_{u,v}^{u',v'} \geq -p_{u,v}^{u',v'}\cdot M, \quad \forall u,v,u',v' \in V. \tag{18}$$

With Formulas 15-18, where M is a large constant, we can derive $b_{u,v}^{u',v'}$ from $p_{u,v}^{u',v'}$ and $b_{u,v}$. In particular, $b_{u,v}^{u',v'}$ is equal to $b_{u,v}$ if $p_{u,v}^{u',v'}=1$, i.e., the traffic is routed on $u' \rightarrow v'$. Otherwise, $b_{u,v}^{u',v'}$ takes 0.

$$\sum_{u,v} b_{u,v}^{u',v'} \leq x_{u',v'}\cdot C, \quad \forall u'v' \in V. \tag{19}$$

Finally, Formula 19 ensures that the amount of traffic routed on a link does not exceed the configured capacity.

The MILP model contains $2|V|^2(|V|-1)^2 + 2|V|(|V|-1)$ variables and $5|V|^2(|V|-1)^2 + 8|V|(|V|-1) + 4|V|$ inequality constraints. As $|V|$ can take a large value in practice (e.g., 256), solving the problem exactly becomes intractable.

4. HEURISTIC ALGORITHMS

In this section, we present two time-efficient heuristic designs for the optimization problem described in Section 3. Specifically, we focus on the optimization of connectivity graphs and routing schemes for a set of ToRs controlled by a single controller (i.e., either for an intra-POD or an inter-POD system), while reconfiguration operations for an entire system can be done by applying the proposed algorithms to intra- and inter-POD systems successively. More comprehensive designs performing joint intra-/inter-POD optimizations will be left as a future research task.

An intuitive idea to optimize the objective function defined by Formulas 1-4 is that we boost the bandwidth between ToRs with higher demands and perform balanced-load routing. Following the idea, we first design a two-phase optimization approach for flexible bandwidth reconfiguration (namely, Flex-TP), which optimizes the connectivity graph and routing schemes successively. Algorithm 1 shows the procedures of Flex-TP. *Lines* 1-2 are for initialization, where we *i*) set the numbers of viable sending (s_u) and receiving ports (r_u) in each ToR, *ii*) make all the ports disconnected (i.e., $x_{u,v}=0$), and *iii*) create an empty routing set P. Here, for the sake of clarity, we overuse the symbol V to denote the set of ToRs in a POD. The for-loop covering *Lines* 3-18 performs the first-phase optimization to determine the connectivity graph. In particular, Flex-TP iterates through the ToRs $|V|-1$ times and each time attempts to configure a communication link for each ToR. With *Lines* 5-12, we assign every ToR except the current one (ToR u) a weight as a function of the amount of unsatisfied traffic demand between the ToRs and the number of available receiving ports. M is a large positive constant introduced to eliminate ToRs whose ports have all been allocated. *Lines* 13-16 add a link from the current ToR to the ToR with the largest weight and update the demand satisfaction and port utilization information accordingly. After the connectivity graph has been determined, we initialize the link capacities with *Line* 19. Next, in the second optimization phase (*Lines* 20-27), Flex-TP calculates a routing path for each ToR pair and allocates transmission capacities on the paths, giving priorities to ToR pairs with higher demands. Specifically, *Line* 22 constructs a cost matrix, where the cost of each link is equal to the reciprocal of the link's residual capacity. *Lines* 23-24 calculate the shortest path with the cost matrix as the routing path for each ToR pair. With *Line* 25, we perform best-effort transmission capacity allocation on the routing path, i.e., allocating a capacity equal to the traffic demand or the maximal available capacity, whichever is smaller. The link capacity state is updated in *Line* 26, after which, the algorithm proceeds to the provisioning for the next ToR pair. The time complexity of Flex-TP is $O(|V|^4)$.

Flex-TP optimizes the connectivity graphs and routing schemes separately, which can lead to compromised performance. To address this issue, we further devise a joint optimization approach (namely, Flex-JO) whose procedures are summarized in Algorithm 2. *Lines* 1-2 initialize the related variables. With *Lines* 3-14, the algorithm iterates through all the ToR pairs in the descending order of traffic demands and attempts to service as much demand as possible with direct communication links. Therein, *Line* 5 calculates the number of links (or wavelengths) needed. If there are enough ports available, *Lines* 7-9 interconnect the related ports and record the routing and capacity allocation solution for the ToR pair. *Lines* 10-12 update the information of link capacity, pending demands, and port utilization. Next, the while-loop from *Lines* 15-35 aims at provisioning

Algorithm 1. Procedures of Flex-TP.

```
1: $s_u \leftarrow |V|-1, r_u \leftarrow |V|-1, \forall u \in V$
2: $x_{u,v} \leftarrow 0, \forall u,v \in V, W \leftarrow D(t), P \leftarrow \emptyset$
3: for each $k \in [1, |V|-1]$ do
4:     for each ToR $u \in V$ do
5:         $\omega_v \leftarrow \begin{cases} W_{u,v}, & r_v > 0 \\ -M, & r_v = 0 \end{cases}, \quad \forall v \in V \setminus u$
6:         if $\max_v \omega_v > 0$ then
7:             $\omega_v = \omega_v \times r_v, \forall r_v > 0$
8:         else if $\max_v \omega_v \neq -M$ then
9:             $\omega_v = \frac{\omega_v}{r_v}, \forall r_v > 0$
10:        else
11:            continue
12:        end if
13:        $v^* \leftarrow \arg\max_v \omega_v$
14:        $x_{u,v^*} \leftarrow x_{u,v^*} + 1$
15:        $s_u \leftarrow s_u - 1, r_{v^*} \leftarrow r_{v^*} - 1$
16:        $W_{u,v^*} \leftarrow W_{u,v^*} - C$
17:    end for
18: end for
19: $U_{u,v} \leftarrow x_{u,v} \times C, \forall u,v \in V$
20: $\hat{D} \leftarrow$ sort $D(t)$ in the descending order
21: for each $\hat{D}_{u,v}$ do
22:     $\omega_{u',v'} \leftarrow \frac{1}{U_{u',v'}}, \forall u',v' \in V$
23:     calculate the shortest path $p_{u,v}$ with $\omega$
24:     store $p_{u,v}$ in $P$
25:     $b_{u,v} \leftarrow \min\{\hat{D}_{u,v}, \min_{(u',v') \in p_{u,v}} U_{u',v'}\}$
26:     $U_{u',v'} \leftarrow U_{u',v'} - b_{u,v}, \forall (u',v') \in p_{u,v}$
27: end for
```

Algorithm 2. Procedures of Flex-JO.

```
1: $s_u \leftarrow |V|-1, r_u \leftarrow |V|-1, \forall u \in V$
2: $x_{u,v} \leftarrow 0, \forall u,v \in V, P \leftarrow \emptyset$
3: $\hat{D} \leftarrow$ sort $D(t)$ in the descending order
4: for each $\hat{D}_{u,v}$ do
5:     $n \leftarrow \left\lceil \frac{\hat{D}_{u,v}}{C} \right\rceil$
6:     if $s_u \geq n$ & $r_v \geq n$ then
7:         $x_{u,v} \leftarrow n$
8:         store $p_{u,v} \leftarrow (u,v)$ in $P$
9:         $b_{u,v} \leftarrow \hat{D}_{u,v}$
10:        $U_{u,v} \leftarrow n \times C - b_{u,v}$
11:        $\hat{D}_{u,v} \leftarrow 0$
12:        $s_u \leftarrow s_u - n, r_v \leftarrow r_v - n$
13:    end if
14: end for
15: for each $\hat{D}_{u,v} > 0$ do
16:     $\omega_{u',v'} \leftarrow \begin{cases} \frac{1}{U_{u',v'}}, & U_{u',v'} \geq \hat{D}_{u,v} \\ \infty, & U_{u',v'} < \hat{D}_{u,v} \end{cases}, \quad \forall u',v' \in V$
17:     calculate the shortest path $p_{u,v}$ with $\omega$
18:     if $p_{u,v} \neq \emptyset$ then
19:         store $p_{u,v}$ in $P$
20:         $b_{u,v} \leftarrow \hat{D}_{u,v}$
21:         $U_{u',v'} \leftarrow U_{u',v'} - b_{u,v}, \forall (u',v') \in p_{u,v}$
22:     else
23:         $U'_{u',v'} \leftarrow U_{u',v'} + \min\{s_{u'}, r_{v'}\} \times C, \forall u',v' \in V$
24:         $\omega_{u',v'} \leftarrow \frac{1}{U'_{u',v'}}, \forall u',v' \in V$
25:         calculate the shortest path $p_{u,v}$ with $\omega$
26:         store $p_{u,v}$ in $P$
27:         $b_{u,v} \leftarrow \min\{\hat{D}_{u,v}, \min_{(u',v') \in p_{u,v}} U'_{u',v'}\}$
28:         for each $(u',v') \in p_{u,v}$ do
29:             $n \leftarrow \begin{cases} \left\lceil \frac{b_{u,v} - U_{u',v'}}{C} \right\rceil, & b_{u,v} - U_{u',v'} > 0 \\ 0, & b_{u,v} - U_{u',v'} \leq 0 \end{cases}$
30:             $x_{u',v'} \leftarrow x_{u',v'} + n$
31:             $U_{u',v'} \leftarrow U_{u',v'} + n \times C - b_{u,v}$
32:             $s_{u'} \leftarrow s_{u'} - n, r_{v'} \leftarrow r_{v'} - n$
33:         end for
34:     end if
35: end for
36: while $s_{u'} > 0, r_{v'} > 0, \exists u',v' \in V$ do
37:     $u \leftarrow \arg\max_{u'} s_{u'}$
38:     $\omega_{v'} \leftarrow \begin{cases} \frac{U_{u,v'}}{x_{u,v'}}, & r_{v'} > 0 \\ M, & r_{v'} = 0 \end{cases}, \quad \forall v' \in V \setminus u$
39:     $v \leftarrow \arg\min_{v'} \omega_{v'}$
40:     $x_{u,v} \leftarrow x_{u,v} + 1$
41:     $U_{u,v} \leftarrow U_{u,v} + C$
42:     $s_u \leftarrow s_u - 1, r_v \leftarrow r_v - 1$
43: end while
```

the pending demands with communication paths of more than one hop. First, in *Lines* 16-21, we try to fully service the demand between two ToRs with the existing bandwidth configured. *Line* 16 builds a cost matrix similar to those used by Flex-TP but removes links that cannot support the full demand. If the shortest path can be found with the cost matrix, *Lines* 19-21 record this path as the routing paths and update the capacity allocation and link capacity utilization information accordingly. Otherwise, with *Lines* 23-33, Flex-JO seeks solutions by adding more links into the connectivity graph. More concretely, *Line* 23 virtually adds all the possible links based on the port availability. *Lines* 24-27 construct a cost matrix with the obtained virtual graph and provision the demand with the best effort. Then, *Lines* 28-33 configure the links actually involved in the routing and capacity allocation solution. In particular, *Line* 29 computes the number of links required to fill up the gaps between the allocated and the currently available capacities. Finally, the while-loop from *Lines* 36-43 interconnects the unassigned ports. In particular, *Lines* 38-39 boost the capacities of highly loaded links (i.e., links with the minimal capacity margins) to mitigate congestion. The time complexity of Flex-JO is also $O(|V|^4)$.

5. ML DESIGNS

While topology reconfiguration enhances the system adaptability, frequent reconfiguration operations can add non-negligible control-plane overheads or cause traffic disruptions. Ideally, reconfiguration should be invoked only when valuable performance enhancement could be anticipated. While throughput can be estimated relatively easily given the routing schemes and the link capacities, end-to-end latency and packet loss rate are affected by multiple factors (e.g., routing path, link load, and equipment condition) and cannot be accurately modeled by simple formulas. In this section, we present ML-based latency and packet loss rate estimator designs to assist in intelligent decision making for topology reconfiguration.

Latency estimator: we model the total time a packet stays in a ToR u by a neural network $\mathcal{F}_{\theta_l}(\chi_{u,v})$, where v is the next-hop

ToR of the packet, θ_l and $\chi_{u,v}$ are the sets of trainable weights and input features, respectively. In particular, $\chi_{u,v}$ contains the information of *i*) the total amount of traffic injected into u, *ii*) the transmission capacity from u to v, and *iii)* the amount of traffic routed through u to v to each destination. A detailed implementation of $\mathcal{F}_{\theta_l}$ will be discussed in Section 6-B. Since latency accumulates along a routing path, we can obtain the estimation of end-to-end latency from u to v as,

$$\tilde{l}_{u,v} = \sum_{u',v'} \mathcal{F}_{\theta_l}(\chi_{u',v'}) \cdot p_{u,v}^{u',v'}, \tag{20}$$

where $p_{u,v}^{u',v'}$ represents the routing scheme (see Table 1). Finally, we can train θ_l by collecting a large data set $\mathcal{S} = \{(\chi, p, l)\}$ and by minimizing the mean prediction error on $\mathcal{S}$. Note that, in the case where ToRs of different characteristics are deployed, a proprietary set of θ_l can be learned for each ToR.

Packet loss rate estimator: the design of the packet loss rate estimator is similar to that of the latency estimator but with the following modifications. First, we model the successful transmission rate ρ of a packet instead of estimating the packet loss rate $(1-\rho)$ directly. This is because end-to-end successful transmission rate can be modeled as the product of the successful transmission rate by each ToR along a routing path, whereas there is not such a simple rule holding for end-to-end packet loss rate. Second, we take $\log_a(\rho)$ as the training labels, where a is a constant. This way, we can achieve the following linear operations,

$$\begin{aligned} \log_a(\tilde{\rho}_{u,v}) &= \sum_{u',v'} \log_a(\tilde{\rho}_{u',v'}) \cdot p_{u,v}^{u',v'} \\ &= \sum_{u',v'} \mathcal{F}_{\theta_\rho}(\chi_{u',v'}) \cdot p_{u,v}^{u',v'}, \end{aligned} \tag{21}$$

where θ_ρ is the set of neural network weights. In other words, we can model the *log*-scale successful transmission rate by each ToR individually with a neural network $\mathcal{F}_{\theta_\rho}(\chi_{u,v})$. Therefore, the scalability of the proposed design can be justified.

With the performance estimators properly trained, we can assess the values of topology reconfiguration proactively. Specifically, given a prospective traffic matrix, we can calculate a reconfiguration scheme with the proposed algorithms and estimate the corresponding latency and packet loss performance with the trained ML models in a relatively short time (e.g., within microseconds). Then, a value function taking into account the performance metrics and the reconfiguration costs can be used to guide the reconfiguration decision. For instance, the value function could suggest maintaining the current system configuration if the anticipated performance gain is only marginal.

6. PERFORMANCE EVALUATION

We evaluated the performance of the proposed designs with numerical simulations for the provisioning of aggregation traffic at the ToR level. Each ToR port was assumed to work at $C = 50$ Gbps with a buffer size of 15 packets. The packet length was set to be 296 Bytes. We assessed the end-to-end latency and packet loss rates using a queuing model discussed in [24]. Note that, although the queuing model adopted does not provide the assessments of the exact latency and packet loss rate values, it incorporates the main factors affecting the system performance and thereby can provide useful insight for evaluating the performance of the proposed designs.

A. Results from the Heuristic Algorithms

We first evaluated the performance of the proposed heuristic algorithms with traffic matrices derived from different application traces, i.e., AMG, CESAR, FFT, and LULESH [25]. We visualize the traffic matrices with heatmaps shown in the first column of Fig. 3, which indicates the traffic intensities among 64 switches. The proposed algorithms are compared with a baseline that always adopts all-to-all interconnects (denoted as All2All in Fig. 3), where each ToR pair communicates through a direct link and is allocated the full link capacity. Figures in columns two through four of Fig. 3 show the results of system throughput, end-to-end latency, and packet loss rate as a function of normalized traffic demand, respectively. Here, throughput is defined as the ratio of the successfully provisioned data rate to the total data rate given by a traffic matrix. The latency and packet loss results are obtained as the average results of per ToR pair for the fairness consideration. Normalized traffic demand γ is the ratio of the total demand (i.e., $\sum_{u,v} D_{u,v}$) to the upper-bound system capacity (i.e., $C|V|(|V|-1)$, which can be reached only when $D_{u,v} = C, \forall u, v$). The setup of γ for each traffic matrix was determined separately so that the performance metrics fall in reasonable ranges. We can see that both of the reconfiguration designs facilitate much higher throughput under different traffic matrices compared with the baseline. In particular, Flex-JO can provision 100% of the demand in most of the cases, even under the heaviest loads. The throughput gain from Flex-JO against the baseline is as much as 5× (for CESAR and FFT under the heaviest loads), demonstrating a clear benefit of the proposed reconfiguration scheme. Meanwhile, by performing joint optimization on connectivity graphs and routing schemes, Flex-JO achieves higher throughput than Flex-TP, especially for LULESH (as much as 24%). The latency results, on the other hand, show "better" performance from the baseline. This is because with all-to-all interconnects, all the ToRs communicate with direct links, whereas with the proposed reconfiguration schemes, certain traffic can be routed on multi-hop paths. Moreover, the proposed schemes enable us to accommodate much more traffic in the systems, which can lead to high-loaded links. The latency performance from Flex-JO is close to that of the baseline under light loads while Flex-TP performs the worst. Considering the fact that Flex-JO facilitates much higher throughput, the slight cost of end-to-end latency is tolerable. A similar observation can be drawn from the results of packet loss rate. Flex-JO achieves packet loss rates comparable to or even lower than those from the baseline under light loads, while the baseline performs better with the increase of load. Again, the performance of Flex-TP is the worst. Note that, the latency and packet loss performance of the proposed schemes would get much better if we evaluate on a per-packet base, because with reconfiguration, we make high-volume traffic transmitted on direct links while using longer routing paths merely for ToR pairs with relatively low demands.

We further evaluated the performance of the proposed designs using random traffic matrices of different scales. Fig. 4 shows the results of throughput, end-to-end latency, and packet loss rate, with $|V|$ equal to 16 or 64. We generated 1,000 random traffic matrices and averaged the results to obtain each data point in the figures. For each traffic matrix, we randomly picked 20% of the ToR pairs to generate high demands ($D_{u,v} \in [0.8, 1]$) while the rest were assigned with low demands ($D_{u,v} \in [0, 0.4]$). Then, the traffic matrix was derived as $D = \gamma C|V|(|V|-1)\frac{D}{\sum_{u,v} D_{u,v}}$. The results show that Flex-JO achieves the highest throughput, with a slightly worse latency performance under heavy loads

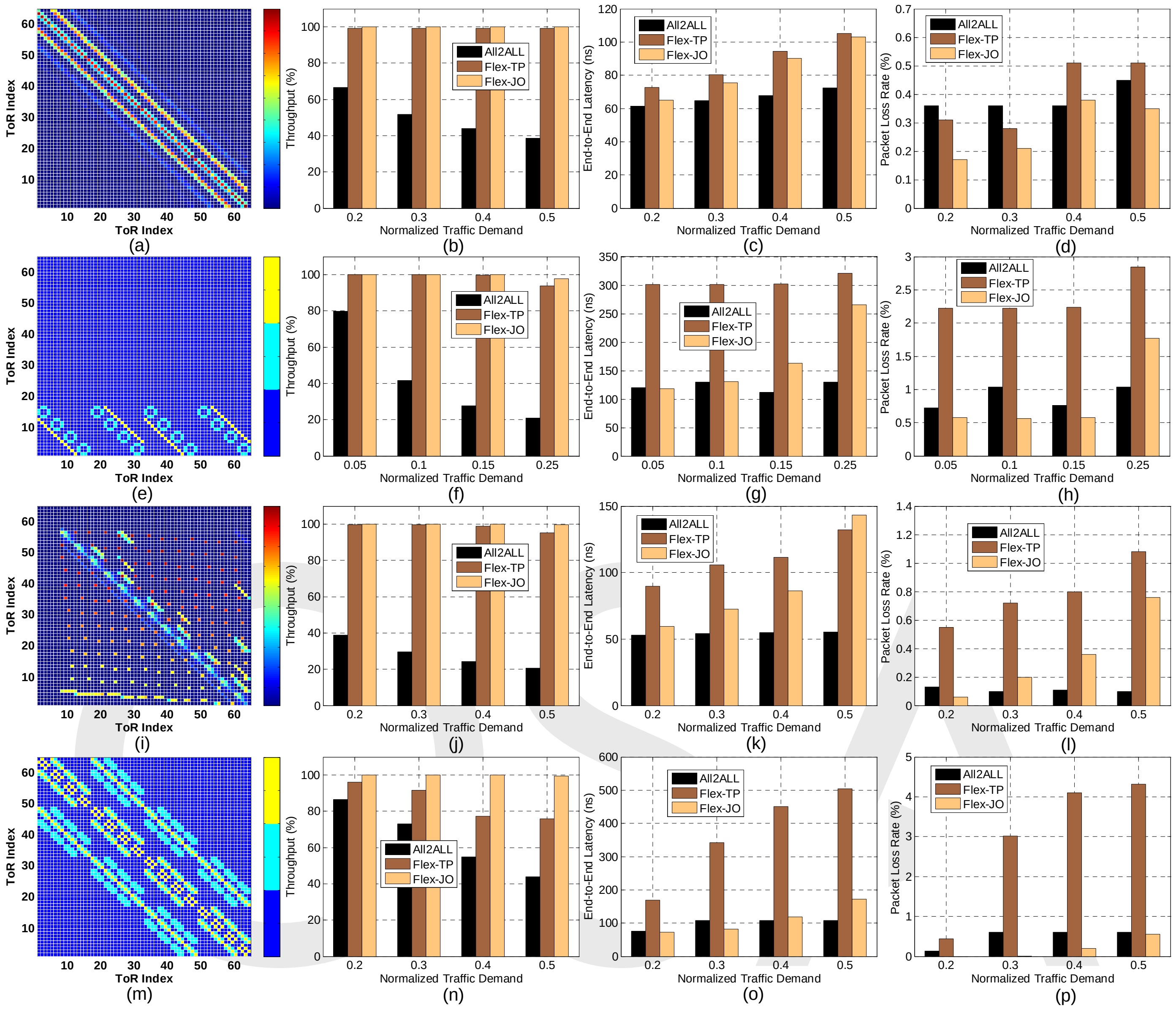

Fig. 3. Results of system throughput (second column), end-to-end latency (third column) and packet loss rate (fourth column) evaluated under four traffic matrices: AMG ((a)-(d)), CESAR ((e)-(h)), FFT ((i)-(l)), and LULESH ((m)-(p)). The latency and packet loss results are obtained as the average results of per ToR pair.

compared with the baseline. The throughput gain from Flex-JO is 21% when the normalized traffic demand is 0.6. Besides, we can see that under such a random traffic setting, Flex-JO can achieve even better latency performance under light loads and much lower packet loss rates than the baseline. This is because Flex-JO can effectively balance the link capacity utilization, which can facilitate better latency and packet loss performance when the total amount of traffic accommodated in a system is not significantly larger compared with the case of the baseline. Meanwhile, the results show similar trends for $|V| = 16$ and $|V| = 64$, verifying the robustness of the proposed reconfiguration designs.

The results in Figs. 3 and 4 have suggested that the skewness of traffic matrices can dramatically influence the benefit of topology reconfiguration. In particular, we can observe up to 5× throughput gains from Flex-JO for CESAR and FFT, where only small portions of ToRs have communication demands. Whereas under the aforementioned random traffic matrices, where all ToRs communicate with each other, the largest throughput gain from Flex-JO is only 21%. To confirm this inference, we performed simulations with random traffic matrices of different skewness. Three cases were evaluated: *Case 1*, $D_{u,v} \in [0,1]$ for every ToR pair; *Case 2*, $D_{u,v} \in [0,0.4]$ for 80% of the ToR pairs, and $D_{u,v} \in [0.8,1]$ for the rest; *Case 3*, $D_{u,v} \in [0,1]$ for 40% of the ToR pairs, and $D_{u,v} = 0$ for the rest. The skewness of traffic increases from *Case 1* to *Case 3*. Fig. 5 shows the comparison between the three cases, where each data point is the average of the evaluations under 1,000 traffic matrices. The normalized traffic demand was set to be 0.6. We can see that the throughput gain from Flex-JO under *Case 3* is 10× to that under *Case 1*, which coincides with the previous inference. The latency and packet loss results also show trends consistent with those

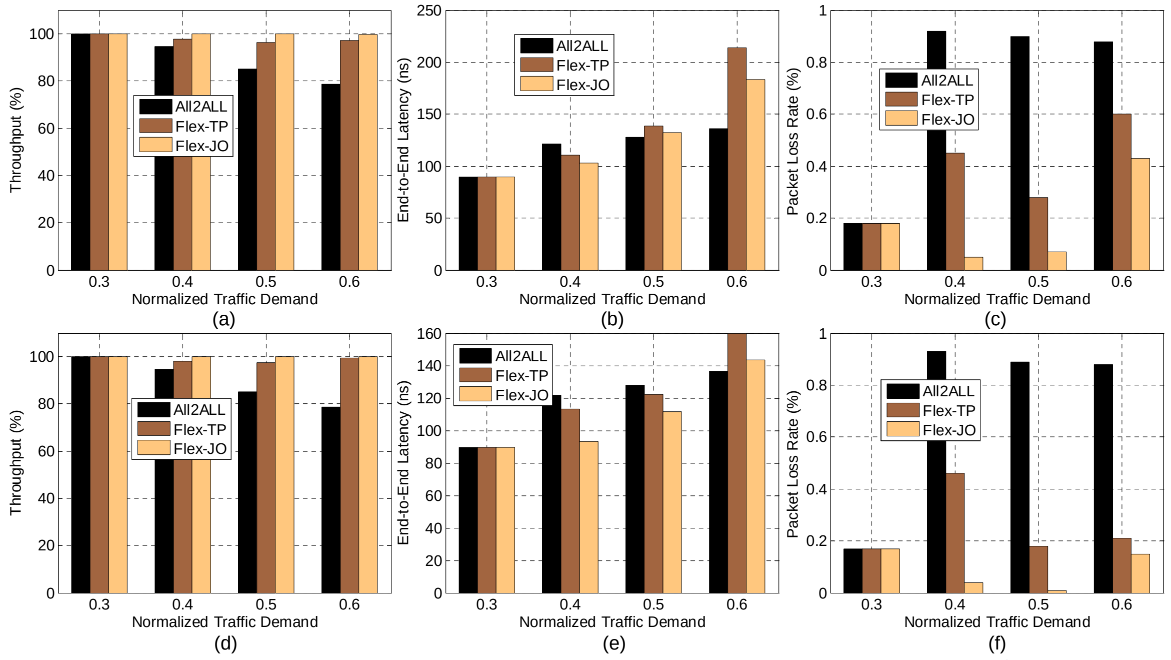

Fig. 4. Results of system throughput (first column), end-to-end latency (second column) and packet loss rate (third column) evaluated under random traffic matrices. (a)-(c): number of ToRs equal to 16. (d)-(f): number of ToRs equal to 64.

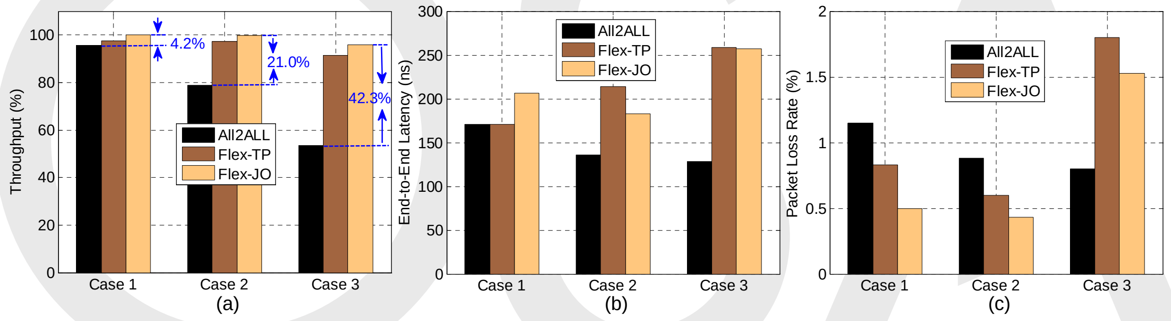

Fig. 5. Performance comparison between different algorithms under random traffic of different skewness.

seen from Figs. 3 and 4. Overall, the results confirm that the more skewed traffic matrices are, the larger is the advantage of topology reconfiguration over the all-to-all interconnects.

B. Results from the ML Designs

Next, we evaluated the performance of the proposed ML-aided estimator designs. We implemented $\mathcal{F}_{\theta_l}(\cdot)$ and $\mathcal{F}_{\theta_\rho}(\cdot)$ with a fully-connected neural network architecture of two hidden layers (16 neurons each layer). Based on the evaluation of different activation functions (e.g., *ReLU* and *ELU*) for the hidden layers, we selected *ReLU* as it provided lower prediction errors. For each normalized traffic demand, we generated 3,000 random traffic matrices according to *Case 2* (with $|V| = 16$) and applied Flex-JO to obtain the connectivity graphs, routing schemes, and the evaluations of end-to-end latency and packet loss rate. The obtained data were divided into training and testing sets following a ratio of 9 : 1. For the packet loss rate estimator, we set the base of the log operations to be $a = 0.99$. Figs. 6(a) and (c) show the evolution of training and validation losses during the training of $\mathcal{F}_{\theta_l}(\cdot)$ and $\mathcal{F}_{\theta_\rho}(\cdot)$, with the normalized traffic demand set to be 0.6. In both cases, the training and validation losses converge to very close values after training of ~ 150 epochs, indicating successful training processes without evident overfitting. Figs. 6(b) and (d) show the results of average prediction error on the testing sets. The errors from both estimator designs increase with the traffic demand because the absolute values of the labels increase (refer to the results in Fig. 4). The largest prediction errors on latency and packet loss rate are 2.25 ns and 3.7×10^{-4} when the normalized traffic demand is 0.6. Such errors are relatively small considering that the average latency and packet loss rate in this case are 183 ns and 4.3×10^{-3}, respectively.

After verifying the accuracy performance of the proposed estimator designs, we evaluated their application in assisting cognitive reconfiguration decision making. In particular, we generated random traffic matrices according to *Case 2* defined in Section 6-A and added random but zero-sum perturbations to each of the obtained matrices. The ML-aided cognitive approach triggers a reconfiguration operation when the prediction results indicate at least a 20% reduction in average end-to-end latency or packet loss rate can be achieved by reconfiguration. We compare the cognitive approach with a traffic-driven one, where reconfiguration is invoked whenever changes in traffic matrices are observed (in practical and more complicated scenarios, a proper threshold of change should be applied). Table 2

Table 2. Performance comparison between the ML-aided and the traffic-driven approaches under different perturbation intensities.

Perturbation Intensity	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3	0.1	0.2	0.3
Traffic-Driven	*1.00*	*1.00*	*1.00*	*183.1*	*185.9*	*193.2*	*0.44*	*0.48*	*0.55*	*99.77*	*99.73*	*99.74*
ML-Aided	*0.16*	*0.70*	*0.90*	*184.9*	*186.5*	*192.8*	*0.47*	*0.51*	*0.56*	*99.71*	*99.62*	*99.67*
	Reconfiguration Frequency			*End-to-End Latency (ns)*			*Packet Loss Rate (%)*			*Throughput (%)*		

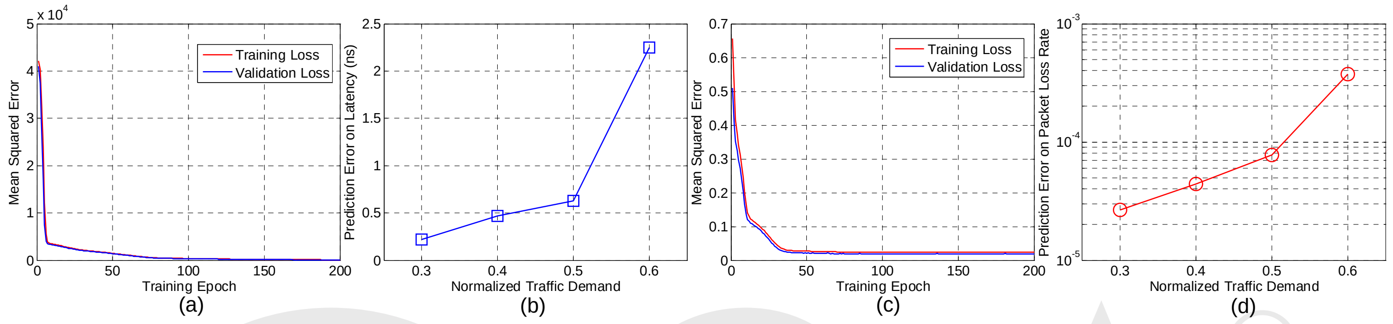

Fig. 6. Results of training and validation losses and average prediction error for: (a)-(b) the end-to-end latency estimator, and (c)-(d) the packet loss rate estimator. The training and validation losses were evaluated with the normalized traffic demand set as 0.6.

summarizes the results of the key performance metrics from the two approaches under different perturbation intensities. Here, perturbation intensity is defined as the maximum proportion of the total demand being moved around. Each value in the table is the average of the results from simulations with 1,000 traffic matrices. We can see the cognitive approach achieves performance of end-to-end latency, packet loss rate, and throughput very close to that of the traffic-driven approach in all the cases but requires lower reconfiguration frequencies. Under low traffic perturbations (e.g., 0.1), only a small portion of reconfiguration operations (16%) can bring notable performance gains. On the other hand, when the changes in traffic distribution become more significant, the cognitive approach advocates reconfiguration most of the time. Note that, the cognitive approach we evaluated only adopts a simple threshold-based policy while more comprehensive designs will be left as one of our future research tasks. In the meantime, while the absolute values of the reconfiguration frequencies (in terms of time scale) would depend on how fast and frequently the traffic matrix varies, in practice, the proposed technology is aiming at performing reconfiguration over time scales larger than milliseconds. In fact, the reconfiguration procedure can take from hundreds of microseconds to few milliseconds (depending on the specific hardware implementations) as it involves the reconfiguration of the optical switch fabric and the ToR switches connected to it (ToRs flow tables needs to be updated).

7. RELATED WORK

There has been a significant amount of architectural and experimental works in the context of optical switching and interconnects for DC and HPC systems [26–28]. The common goal of these solutions is to improve the scalability and performance of large-scale cloud computing systems while reducing their cost and power consumption. Most of the proposed solutions aim at replacing either partially or completely the electronic switches (in the aggregation and core layers) of legacy fat-tree and spine-leaf architectures, with various optical switching and interconnect solutions. Hybrid approaches [9, 10] assume the use of OCS to supplement a legacy EPS network and offload certain type of traffic (either based on traffic-driven or application-driven approaches) to the high-bandwidth optical circuits with reconfiguration times ranging from hundreds of microseconds [29] to tens of milliseconds [9]. Other approaches like in [13, 30–32] assume that the EPS is performed only at the edge or aggregation switches which are optically interconnected through either optical circuits based on spatial and wavelength-domain switches [13, 30] or fast optical packet switches based on wavelength and/or spatial routing [31, 32].

In particular, the works in [13, 30] introduced the concept of topology reconfiguration and bandwidth steering to match the interconnects with specific applications and traffic patterns. In [30], the authors proposed an architecture and demonstrated in a hardware testbed how to interconnect aggregation switches optically in a hierarchical all-to-all topology (like in a dragonfly topology [33]) by using AWGs. By using transceivers with tunable lasers, the bandwidth of some links could be temporarily augmented to solve congestion due to hotspots in the network. More recent works have demonstrated inter-POD bandwidth reconfiguration in a dragonfly topology [33] using SiPh integrated circuits (PICs) switches based on MRRs [34], MZI devices [35], or Flex-LIONS [18]. However, all these works are demonstrated only for small-scale systems and do not tackle or demonstrate scheduling of optical reconfiguration.

Finally, there have been a few works investigating optical reconfiguration for HOE switching architectures, leveraging either heuristic [13, 17] or ML-based [14–16, 36] approaches. In [13, 17], the authors modeled the problems of determining the optimal configuration of an OCS switch as matching problems and developed heuristic algorithms based on existing algorithms in graph theory. In [14], the authors explored a supervised learning approach and trained deep neural networks to predict the most appropriate OCS configuration for each observed traffic distribution. Other ML-based approaches apply deep reinforcement learning to learn reconfiguration policies autonomically from a large amount of trial and error, either for gradual topology augmentation [15] or for selecting the connectivity graphs to use [16, 36]. However, the effectiveness of these cognitive designs were only demonstrated with oversimplified examples.

8. CONCLUSIONS

In this paper, we proposed an ML-aided cognitive approach for effective bandwidth reconfiguration under a 2D-Hyper-X optical interconnect architecture. We formulated the related optimization problem as an MILP model and devised time-efficient heuristic algorithms assisted by an ML-based end-to-end performance estimator design. Simulation results demonstrate up to 5× throughput gain from the proposed approach compared with the baseline. Future research directions include: 1) investigating more comprehensive designs that perform joint optimization of reconfiguration scheduling for intra-/inter-POD systems while providing additional flexibility in routing selection, for instance, incorporating multipathing; 2) studying low-cost or hitless migration schemes for bandwidth steering; 3) developing ML approaches that can learn the optimal reconfiguration policies directly, eliminating the use of heuristic algorithms.

ACKNOWLEDGMENTS

This work was supported in part by ARO award # W911NF1910470, DoD award # H98230-19-C-0209, NSF ECCS award # 1611560, and by DoE UAI consortium award # DE-SC0019582, DE-SC0019526, and DE-SC001969.

REFERENCES

1. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," SIGCOMM Comput. Commun. Rev. **38**, 63–74 (2008).
2. "broadcom-s-tomahawk-3," https://www.broadcom.com/blog/broadcom-s-tomahawk-3-ethernet-switch-chip-delivers-12-8-tbps-of-\speed-in-a-single-16-nm-device.
3. J. Ahn, N. Binkert, A. Davis, M. McLaren, and R. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis,* (2009), pp. 1–11.
4. A. Singla, C. Hong, L. Popa, and P. Godfrey, "Jellyfish: Networking data centers randomly," in *Proc. USENIX Symposium on Networked Systems Design and Implementation,* (2012), pp. 225–238.
5. A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards optimal-performance datacenters," in *Proc. International on Conference on emerging Networking Experiments and Technologies,* (2016), p. 205–219.
6. J. Domke, S. Matsuoka, R. Ivanov, Y. Tsushima, T. Yuki, A. Nomura, S. Miura, N. McDonald, L. Floyd, and N. Dubé, "HyperX topology: First at-scale implementation and comparison to the fat-tree," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis,* (2019), pp. 1–23.
7. H. Isono, "Latest standardization status and its future directions for high speed optical transceivers," in *Metro and Data Center Optical Networks and Short-Reach Links II,* vol. 10946 (2019), pp. 11–16.
8. P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," IEEE Netw. **29**, 36–42 (2015).
9. 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," SIGCOMM Comput. Commun. Rev. **40**, 339–350 (2010).
10. G. Wang, D. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "C-through: Part-time optics in data centers," SIGCOMM Comput. Commun. Rev. **40**, 327–338 (2010).
11. T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," SIGCOMM Comput. Commun. Rev. **40**, 92–99 (2010).
12. M. Balanici and S. Pachnicke, "Machine learning-based traffic prediction for optical switching resource allocation in hybrid intra-data center networks," in *Proc. Optical Fiber Communication Conference,* (2019). Paper Th1H.4.
13. K. Chen, K. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, Y. Wen, and Y. Chen, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," IEEE/ACM Trans. Netw. **22**, 498–511 (2014).
14. M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, and J. Zhu, "Neural network meets DCN: Traffic-driven topology adaptation with deep learning," Proc. ACM Meas. Anal. Comput. Syst. **2** (2018).
15. S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "DeepConf: Automating data center network topologies management with machine learning," in *Proc. Workshop on Network Meets AI & ML,* (2018), pp. 8–14.
16. 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. Light. Technol. **37**, 4921–4934 (2019).
17. M. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," J. Opt. Commun. Netw. **12**, B44–B54 (2020).
18. X. Xiao, X. Proietti, G. Liu, H. Lu, H. Fotouhi, S. Werner, S. Zhang, and S. J. B. Yoo, "Silicon photonic flex-lions for bandwidth-reconfigurable optical interconnects," J. Sel. Top. Quantum Electron. **26**, 1–10 (2020).
19. T. Seok, J. Luo, Z. Huang, K. Kwon, J. Henriksson, J. Jacobs, L. Ochikubo, R. Muller, and M. Wu, "MEMS-actuated 8×8 silicon photonic wavelength-selective switches with 8 wavelength channels," in *Proc. Conference on Lasers and Electro-Optics,* (2018). Paper STu4B.1.
20. S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proc. ACM SIGCOMM,* (2017), p. 281–294.
21. F. Yao, J. Wu, G. Venkataramani, and S. Subramaniam, "A comparative analysis of data center network architectures," in *IEEE International Conference on Communications (ICC),* (2014), pp. 3106–3111.
22. J. A. Kahle, J. Moreno, and D. Dreps, "2.1 summit and sierra: Designing AI/HPC supercomputers," in *Proc. IEEE International Solid-State Circuits Conference,* (2019), pp. 42–43.
23. Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," J. Light. Technol. **31**, 15–22 (2013).
24. K. Kosek-Szott, "Throughput, delay, and frame loss probability analysis of IEEE 802.11 DCF with M/M/1/K queues," in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications,* (2013), pp. 2234–2238.
25. "Characterization of DOE Mini-apps," https://crd.lbl.gov/departments/computer-science/cag/research/past-research/characterization-of-doe-mini-apps-draft/.
26. C. Kachris and I. Tomkos, "A survey on optical interconnects for data centers," IEEE Commu. Surv. Tutor. **14**, 1021–1036 (2012).
27. Q. Cheng, S. Rumley, M. Bahadori, and K. Bergman, "Photonic switching in high performance datacenters," Opt. Express **26**, 16022–16043 (2018).
28. S. J. B. Yoo, R. Proietti, and P. Grani, *Photonics in Data Centers. Optical Switching in Next Generation Data Centers* (Springer, Cham, 2018).
29. N. Farrington, A. Forencich, P. Sun, S. Fainman, J. Ford, A. Vahdat, G. Porter, and G. Papen, "A 10 μs hybrid optical-circuit/electrical-packet

network for datacenters," in *Proc. Optical Fiber Communication Conference,* (2013). Paper OW3H.3.

30. Z. Cao, R. Proietti, M. Clements, and S. J. B. Yoo, "Experimental demonstration of flexible bandwidth optical data center core network with all-to-all interconnectivity," J. Light. Technol. **33**, 1578–1585 (2015).
31. R. Proietti, Z. Cao, C. Nitta, Y. Li, and S. J. B. Yoo, "A scalable, low-latency, high-throughput, optical interconnect architecture based on arrayed waveguide grating routers," J. Light. Technol. **33**, 911–920 (2015).
32. F. Yan, X. Xue, and N. Calabretta, "HiFOST: a scalable and low-latency hybrid data center network architecture based on flow-controlled fast optical switches," J. Opt. Commun. Netw. **10**, 1–14 (2018).
33. J. Kim, J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proc. International Symposium on Computer Architecture,* (2008), pp. 77–88.
34. Y. Shen, A. Gazman, Z. Zhu, M. Teh, M. Hattink, S. Rumley, P. Samadi, and K. Bergman, "Autonomous dynamic bandwidth steering with silicon photonic-based wavelength and spatial switching for datacom networks," in *Proc. Optical Fiber Communication Conference,* (2018). Paper Tu3F.2.
35. K. Wen, P. Samadi, S. Rumley, C. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis,* (2016), pp. 166–177.
36. Y. Shang, X. Chen, R. Proietti, B. Guo, S. Huang, and S. J. B. Yoo, "DeepAutonet: Self-driving reconfigurable HPC system with deep reinforcement learning," in *Proc. Asia Communications and Photonics Conference,* (2019). Paper S3C. 4.