Machine-Learning-Aided Dynamic Reconfiguration in Optical DC/HPC Networks (Invited)

Sandeep Kumar Singh[1], Che-Yu Liu[2], S. J. Ben Yoo[1], and Roberto Proietti[1, 3, *]
[1]*Department of Electrical and Computer Engineering, University of California, Davis, CA, USA, 95616*
[2]*Department of Computer Science, University of California, Davis, CA, USA, 95616*
[3]*Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy*
** Corresponding author: roberto.proietti@polito.it*

***Abstract*—The high bandwidth and low latency requirements of modern computing applications with their dynamic and non-uniform traffic patterns impose severe challenges to current data center (DC) and high performance computing (HPC) networks. Therefore, we present a dynamic network reconfiguration mechanism that could satisfy the time-varying applications' demands in an optical DC/HPC network. We propose a direct and an indirect topology extraction methods based on a machine learning-aided traffic prediction approach under multi-application scenario. The traffic prediction for topology extraction and bandwidth reconfiguration (PredicTER) method could lead to frequent topology and bandwidth reconfiguration. In contrast, the indirect approach, namely traffic prediction with clustering for topology extraction and bandwidth reconfiguration (PrediCLUSTER), utilizes an unsupervised learning-based clustering model to first associate the predicted traffic to one of possible traffic clusters, and then extracts a common topology for the cluster. This restricts the reconfigured topology set to the number of traffic clusters. Our simulation results show that the time-average of mean packet latencies (and total dropped packets) over 60 seconds of time-varying traffic under the PredicTER, PrediCLUSTER and a static topology are $37.7\mu s, 41.2\mu s$, and $50.2\mu s$ (and $37,967$, $12,305$, and $36,836$), respectively. Overall, the PredicTER (and PrediCLUSTER) method(s) can improve the end-to-end packet latency by 24.9% (and 17.8%), and the packet loss rate by -3.1% (and 66.6%), as compared to the static flat Hyper-X-like topology.**

I. Introduction

Modern high performance computing (HPC) and data center (DC) workloads, including distributed machine learning and graph analytics, exhibit dynamic change in communication patterns with non-uniform spatial and temporal distributions [1]. Thus, these heterogeneous applications exhibit traffic profiles that might not match with fixed multi-stage electrical DC/HPC architectures, resulting in low resource utilization and performance bottleneck [2]. To overcome this, the switching and computing systems should reconfigure themselves to adapt to the changes in data flow patterns for a given set of workloads. Recently, there have been a few proposals and studies looking at the benefits of using silicon photonic switches to enable topology and bandwidth reconfiguration of direct optical interconnect topologies in both spectral and spatial domains upon demand [3], [4]. By allocating interconnection resources where and when is needed and eliminating intermediate electronic switches and optical TRXs, it is possible to deliver superior performance at a fraction of the cost and energy required in legacy architectures [5].

In reconfigurable optical DC/HPC networks, a critical aspect is the codesign of control and management plane architecture to orchestrate when, where and how to perform the reconfiguration operation to adapt the interconnection topology and links' bandwidth to the traffic characteristics. Machine learning (ML) techniques, especially neural networks, have been applied in various stages (i.e., when, where and how) of the reconfiguration process for traffic engineering, resource allocation, and service provisioning tasks in DC/HPC networks [6]. These ML-based approaches can provide, in general, better scalability, adaptability, and performance when compared to optimal and heuristic methodologies.

Recurrent neural network (RNN) architectures, in particular long short-term memory (LSTM), with their ability to learn long-term spatio-temporal correlation of traffic data have been widely investigated for traffic prediction in order to know when is time to perform certain resource allocation tasks [7]–[9]. The deep reinforcement learning can learn instead reconfiguration policies from repeated trials and errors [10]. The use of supervised learning, e.g., neural networks [11] together with a traffic clustering approach is also promising as the traffic characteristics can be learned without being labeled manually to generate the topology needed [12]. Additionally, deep neural networks (DNN) are useful in the estimation or prediction of network performance metrics, such as packet loss rate, latency, and job completion time, which can be used to trigger the network reconfiguration operation [13]. Nevertheless, the labeling and training of the DNN models require some sort of efforts and collecting a large amount of performance data. Meanwhile, these approaches mostly use a fixed-threshold-based policy applied to the performance estimations. While the reconfiguration operation can lead to performance gain under skewed and non-uniform traffic, it is important not to reconfigure electrical and optical switches frequently, as these operations involve updating routing tables and can cause traffic disruption [14]. Hence, it is desirable to implement effective reconfiguration policies.

In this paper we investigate the trade-off between the network performance and reconfiguration using two different ML-aided traffic-topology characterization methods. We propose a traffic prediction for topology extraction and band-

978-3-903176-44-7

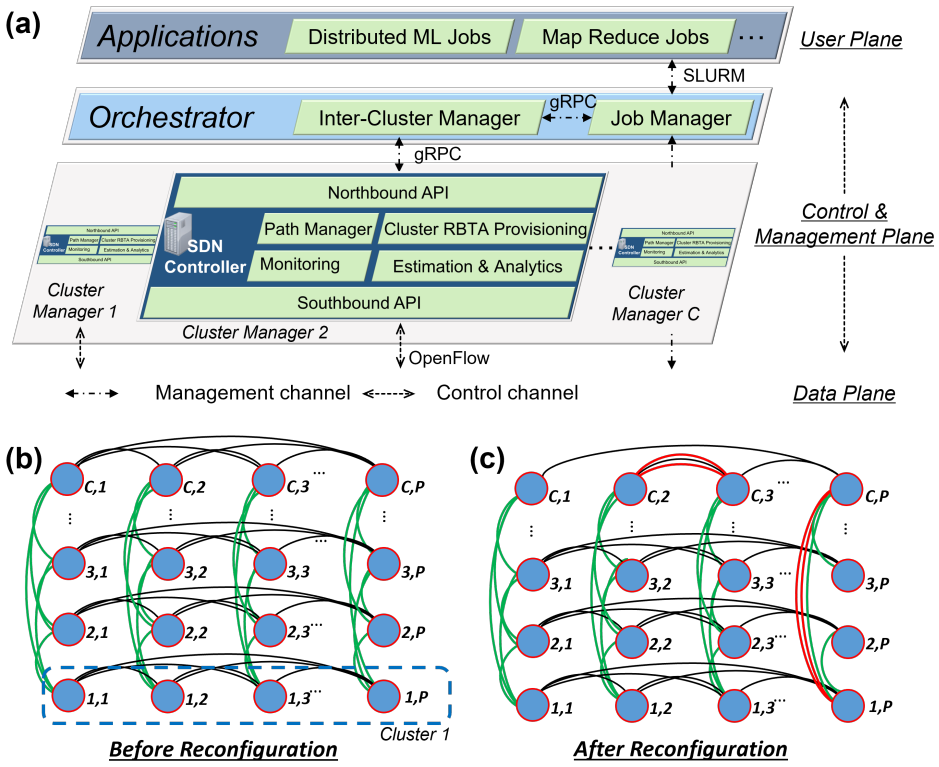

Fig. 1. (a) Software-Controlled Reconfigurable HPC/DC Network Architecture. (b) 2D Hyper-X interconnect architecture before reconfiguration, where clusters are organized into rows with P ToRs per cluster. (c) 2D Hyper-X interconnect architecture after reconfiguration.

width reconfiguration method, in short PredicTER, under a multi-application scenario. We use a long-short term memory (LSTM)-based encoder-decoder RNN model to train time-varying top-of-rack (ToR)-to-ToR traffic matrix, and utilize it to extract topology and reconfigure the wavelengths over fiber links connecting ToRs of reconfigurable flat DC/HPC architecture [4]. We also propose a traffic prediction with clustering for topology extraction and bandwidth reconfiguration method (PrediCLUSTER), utilizing an unsupervised learning approach. We evaluate these methods against an all-to-all network without reconfiguration. Our simulation results show that PrediCLUSTER reduces the number of reconfigurations and packet loss rate at the cost of the increase in average packet latency as compared to the PredicTER method.

The rest of the paper is organised as follows. Section II briefly describes the software-controlled reconfigurable optical HPC/DC networks. Section III discusses the details of reconfiguration algorithm. Section IV discusses the evaluation results. Finally, Section V concludes the paper.

II. Software-Controlled Reconfigurable HPC/DC Network Architecture

Fig. 1(a) shows the architecture of the control and management plane (CMP) that drives the reconfiguration operations. Note that the codesign of the data plane (hardware) and control and management plane (software and algorithms) is key for using any optical switching paradigm. The CMP is centralized at the cluster level but distributed between clusters. For each cluster, a software-defined networking (SDN) controller interfaces with a cluster-level photonic switch and P interconnected ToRs.

The architecture comprises the user plane layer, CMP layer, and data plane layer. The user plane layer communicates about its job's resource requirements, communication patterns, etc. to a Job Manager of the CMP layer via an open-source user interface, for example, simple Linux utility for resource management (SLURM) [15]. Some examples of workloads are scientific computing and distributed machine learning, particularly distributed ML jobs such as recommender, translator, image processing, and map-reduce jobs. The distributed jobs could share information and parameters among themselves using the popular message passing interface (MPI) API.

The job manager places the workloads into the servers and informs an Inter-Cluster Manager about the new job mapping and its communication requirements over a request-response protocol, for instance Google remote procedure calls (gRPC). The Inter-cluster manager disseminates the job placement information to relevant Cluster Managers through the northbound API (e.g., gRPC) to reconfigure the underlining network topology to suit the new and other existing jobs inside relevant clusters. The SDN controller calls the cluster routing, bandwidth and topology assignment (RBTA) provisioning module to compute reconfiguration schemes, including the target connectivity graph, routing schemes, stepwise reconfig-

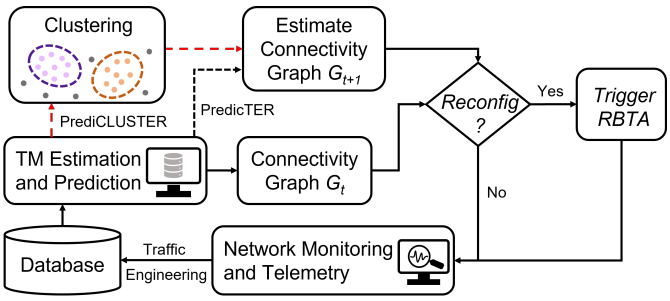

Fig. 2. Reconfiguration for RBTA triggering mechanism involving traffic monitoring, estimation, and prediction with connectivity graph computation without (PredicTER) and with clustering (PrediCLUSTER) methods.

uration operations (which device/routing tables to reconfigure in each step), to minimize traffic disruption. The RBTA provisioning calls the Path Manager to provide the current topology and connectivity (paths) information, and uses the traffic monitoring, estimation and data analytics to compute the optimum topology for the given workloads. The controller reconfigures the related ToRs (via, for instance, OpenFlow [16], by distributing new flow entries) and Flex-LIONS (via, for instance, OpenFlow with optical extension). Note that the off-the-shelf ToR electrical switches can be controlled using these standard protocols. However, the softwarized control of optical switching would require implementing specialized control flow programs and SDN agents [6].

For the studies in this paper we consider an Hyper-X network that can be built by interconnecting N servers in a rack connected with a k port ToR switch (shown by a circle in Fig. 1(b)), where $N \leq k/2$. Multiple racks are organized into clusters, and the ToR switches are interconnected in a 2D flat Hyper-x architecture as shown in Fig. 1(b). In each row cluster, P ToRs are interconnected by a photonic switch, shown as black links. These row clusters are connected using additional layer of inter-cluster photonic switches, shown in green links. Thus, these clusters are arranged into C rows and P columns. Fig. 1(c) shows an example of reconfigured topology and bandwidth assignment. Notice that some ToRs are interconnected with multiple parallel links (in red) to meet the applications' traffic and communication patterns.

III. ML-Aided Reconfiguration Approach

Fig. 2 shows the reconfiguration mechanism involving *when, where* (i.e., which cluster), and *how* to trigger the reconfiguration by the Inter-cluster manager. It periodically monitors, estimates and predicts the traffic matrix for next time step and invokes the cluster manager to determine whether a reconfiguration is required. It uses the traditional graph and ML tools to facilitate the decision making process. One approach is to compare the estimated connectivity graph G_{t+1} with the current connectivity graph G_t, and when they are sufficiently different, the reconfiguration could be triggered. However, the frequent reconfiguration process might be disruptive and costly due to live traffic flows. Therefore, a clustering method is essential in limiting the traffic mapping to a fixed set of

Algorithm 1 Multi-cluster connectivity graph computation.

input: weight $W_{t+1} \leftarrow$ normalized predicted traffic $\widehat{\mathbf{D}}_{t+1}$, connectivity graph G_0
output: connectivity graph topology G_{t+1}
$\mathbb{P} \leftarrow$ a set of shortest paths from all-to-all source-destination $(s-d)$ pairs based on the graph G_0
if PrediCLUSTER **then**
    compute cluster id for $\widehat{\mathbf{D}}_{t+1}$, and the average cluster traffic $\widetilde{\mathbf{D}}_{t+1} \leftarrow \frac{1}{|\mathbb{C}_i|}\sum_{D_t \in \mathbb{C}_i} D_t$, $W_{t+1} \leftarrow \widetilde{\mathbf{D}}_{t+1}$
end if
while ToRs' port-pairs are free, or $\max W_{t+1} \neq -\infty$ **do**
    select a $s-d$ pair (i,j) which maximizes the product of weight vector $w^{i,:} \in W$ and available ports
    for each hop on a shortest path $p_{i,j} \in \mathbb{P}$ **do**
        if all hops have available port-pairs **then**
            add a link between each ToR-pair on $p_{i,j}$
            update G_{t+1}, $w^{i,j} \leftarrow w^{i,j} - C$
        else
            assign $w^{i,j} \leftarrow -\infty$.
        end if
    end for
end while
Connect remaining available port-pairs in each cluster of G_{t+1} based on the decreasing order of traffic $\widehat{\mathbf{D}}_{t+1}$.

topology clusters using unsupervised learning. It can minimize the reconfiguration frequency and achieve nearly the same performance gain. We describe below four key modules of the reconfiguration mechanism for RBTA.

A. Traffic Estimation and Prediction

Let $D_t = \{d\}_t^{i,j}, i,j = 1,2,\ldots,N$ be an $N \times N$ estimated ToR-to-ToR traffic matrix at time t. The aim of a traffic prediction (or regression) model is to forecast the subsequent L $N \times N$ traffic instances, given the last T time instances prior. Thus, a regression model estimates $\widehat{\mathbf{D}}_{t+T:t+T+L} = f\left(\mathbf{X}^i_{t:t+T}, \mathbf{W}\right)$, with $\widehat{\mathbf{D}}_{t+T:t+T+L}$ the traffic forecast for L time instances, $\mathbf{W}$ the model weights optimized during the training process and $f(\cdot)$ the estimator *per se*. Among the recurrent networks, the LSTM encoder-decoder is known for its superior performance at temporal prediction and it constitutes the traffic prediction algorithm for our work.

B. Traffic-Topology Mapping with Unsupervised Learning

An unsupervised learning approach for clustering discovers the patterns of the traffic matrices in $\mathbb{D}$ by segregating them into different clusters based on their spatial distance or density. We adopt the density-based spatial clustering of applications with noise (DBSCAN) algorithm [17] to cluster the data points into an arbitrary number of clusters determined by its parameters ε representing the neighbourhood of a core point, and $MinPts$ the minimum number of density-reachable points. For the similar traffic matrices belonging to a cluster, a common connectivity graph can be generated. In other words, the reconfiguration will only be triggered when the predicted

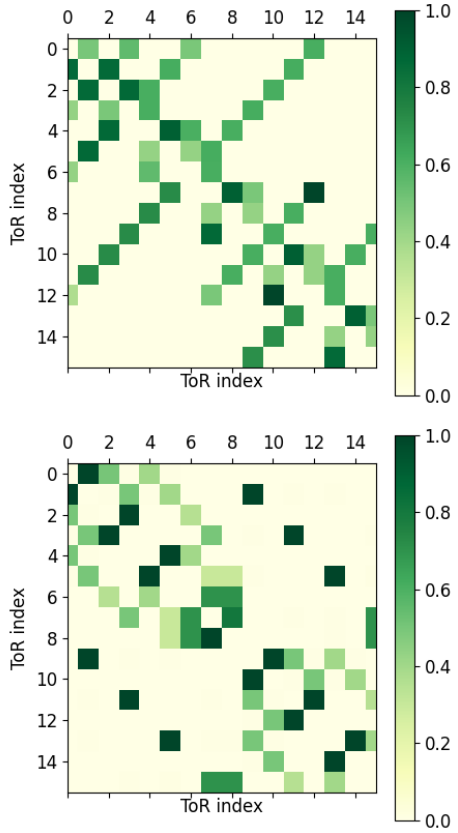

Fig. 3. Heatmaps of two applications showing the spatial distribution of traffic among ToRs. Crystal Router (top), and MiniFE (bottom).

traffic matrix ($\widehat{\mathbf{D}}_{t+1}$) is clustered differently from the previous one ($\widehat{\mathbf{D}}_t$). Thus, by exploiting the inherent structure of traffic data, we can largely avoid unnecessary reconfiguration operations while sustaining the desired performance gains.

C. Connectivity Graph Optimization

Given the predicted ToR-to-ToR traffic matrix (TM) $\widehat{\mathbf{D}}_{t+1}$, the number of ports per ToR k for interconnecting k_h (and k_v) ToRs in a horizontal (and vertical) cluster(s), where $k_h + k_v \leq k/2$, and the topology connectivity graph G_0 with each cluster interconnected in an all-to-all fashion (See Fig. 1(b)), Algorithm 1 summarizes how to compute the connectivity graph at time t for the next time interval, i.e., G_{t+1}. The basic idea is to iteratively interconnect the largest number of available ToR ports on the shortest paths between ToR-pairs that potentially carry larger traffic.

In Algorithm 1, *Step* 3 precomputes the shortest paths between all source-destination ToR pairs. In *Step* 4-6, if the reconfiguration method is *PrediCLUSTER*, we assign the average TM of a cluster (w.r.t. the number of TMs in the cluster $\mathbb{C}$), that the predicted TM $\widehat{\mathbf{D}}_{t+1}$ belongs, to the weight W_{t+1}. In *Step* 7-17, we iteratively interconnect available ToR ports on each hop of a shortest path between a ToR pair with larger amounts of traffic pending to be provisioned and larger number of ports not assigned yet. When all hops on a path $p_{i,j}$ have available ports to connect, the connectivity

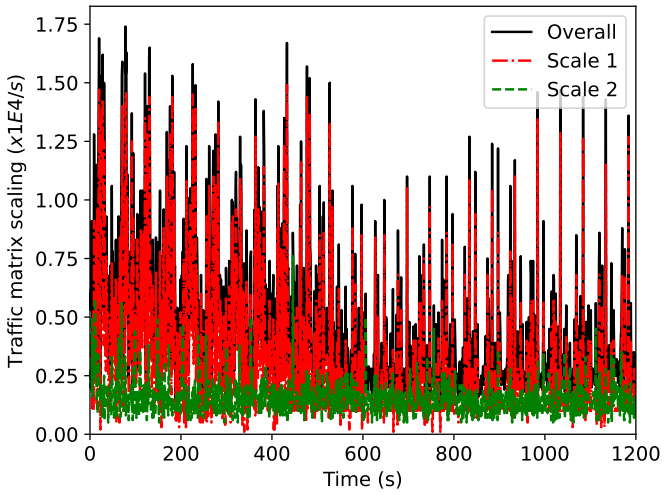

Fig. 4. Time-varying traffic matrix scaling for two different applications. The overall TM(t) is the sum of scaled TMs of two applications.

graph G_{t+1} is updated and the weight $w^{i,j}$ is decreased by a wavelength capacity C (*Steps* 10-12). Otherwise, the weight $w^{i,j}$ is updated to negative infinite (*Steps* 13-14). When all weights are negative infinite and still some ports are available, we connect ToR-pairs in decreasing order of the predicted TM $\widehat{\mathbf{D}}_{t+1}$ in *Step* 18.

D. Reconfiguration process for RBTA

When the reconfiguration process for RBTA is triggered, by comparing G_{t+1} and G_t, we identify ports and corresponding links to be added into or removed from the current network topology. More importantly, during a reconfiguration period we adopt an *offload-before-reconfigure* approach to stop accepting new packets to ports to be reconfigured to reduce packet loss during the reconfiguration phase. Furthermore, we apply the equal-cost multipath (ECMP) routing [18] with flow splitting over parallel next hops for its capability of increasing bandwidth utilization by load-balancing traffic over multiple paths.

IV. Performance Evaluation

We evaluated the proposed reconfiguration design performance using the Netbench packet simulator [19] with an extension of the routing table update mechanism for the time-varying traffic and reconfiguration evaluation. We consider 16 ToRs, where each ToR connects to 16 servers. The 16 ToRs are arranged into a 4×4 flat (horizontal and vertical) architecture. The servers generate packets following the Poisson processes. The upper bound of Alizadeh Web Search distribution [20] was used to emulate the sizes of the flows injected into the network. The source-destination pairs of the packet flows were selected according to the traffic distributions derived from the time-varying traffic traces. We consider two real HPC applications, i.e., Crystal Router, and MiniFE [21], with different scaling over time. Fig. 3 shows the heatmaps of the traffic matrices. To emulate their time-varying traffic

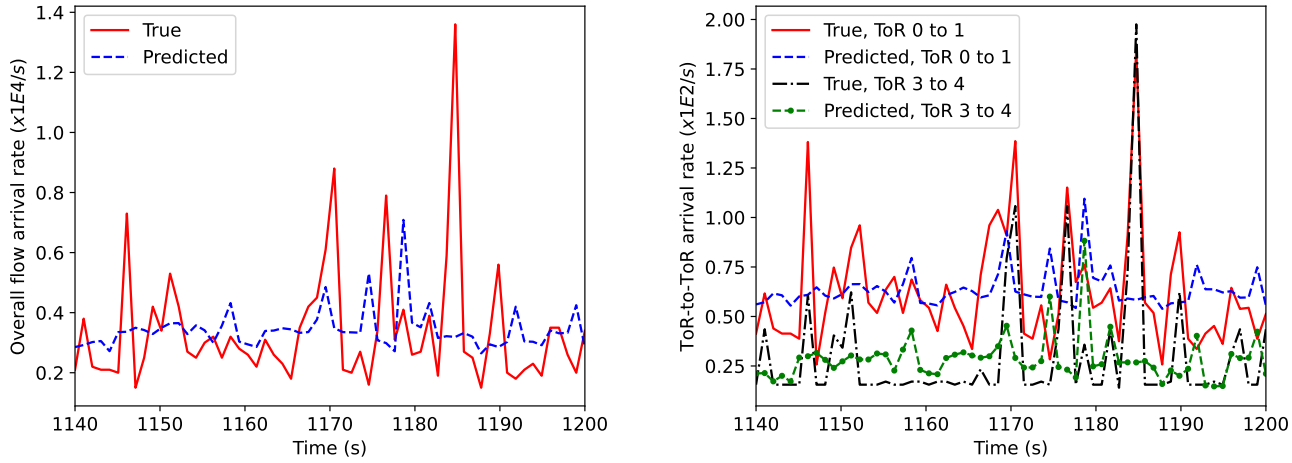

Fig. 5. True and predicted data are shown for the overall traffic flow rate (left), and some ToR-to-ToR traffic flow rate (right).

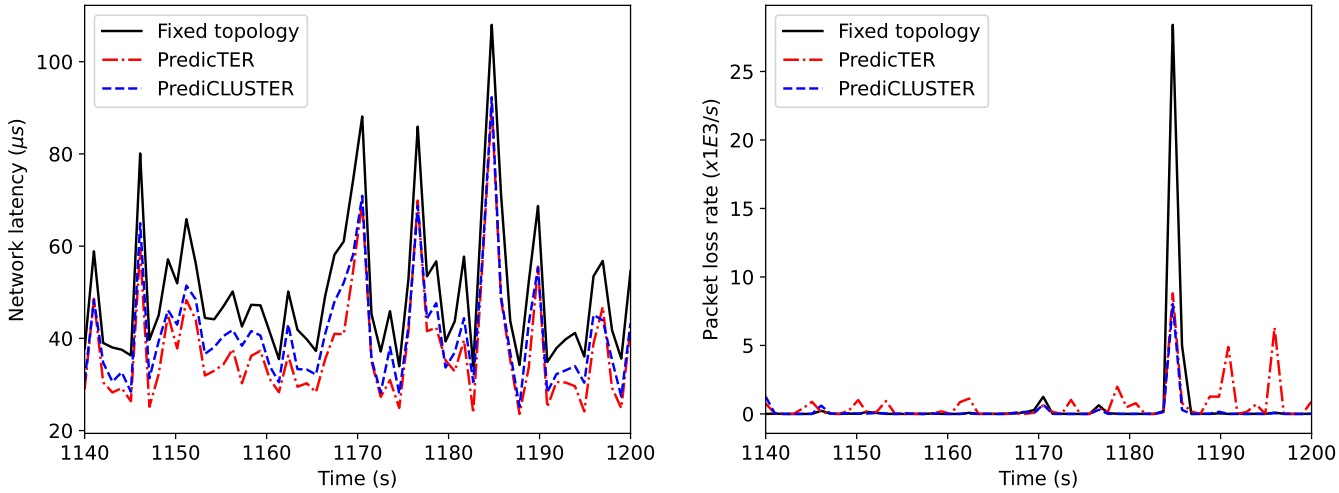

Fig. 6. Evaluation of the average end-to-end packet latency (left) and the average packet loss rate (right) in a fixed Hyper-X topology (i.e., no reconfiguration), and two reconfiguration (PredicTER and PrediCLUSTER) design methods.

patterns, we combine them with different scaling over time, as shown in Fig. 4. They reflect the changes in overall traffic bandwidth utilization during the run-time of applications [9]. We assumed an identical wavelength/link capacity of 10 Gbps and a link delay of 20 ns. We adopted data center TCP [22] as the transport protocol responsible for the communications between two specific network devices. We set the buffer size of each ToR port as $300,000$ Bytes and congestion threshold is $270,000$ bytes. ECMP routing was used to decide the forwarding of packets. We also evaluate the algorithm under fixed topology, i.e., no reconfiguration scenario.

Parameters and Settings: A TM consists of 16×16 data entries. An input sample to train the LSTM encoder-decoder model is formed by 10 consecutive TMs measured at every time unit (second), and it learns to output an 11th TM. The model is trained (90%) and validated (5%) with the first 1130 sequence of TMs to predict traffic for next timestamp. Each encoder and decoder has 100 LSTM cells, and the model is trained with an Adam optimizer with a learning rate 0.001. We use the early stopping criteria to overcome the overfitting. The test dataset is formed on the last 60 sequences of consecutive TM instances (see Fig. 4). We monitor and predict traffic every time unit, which is second, as each second of real traffic takes several minutes in the discrete event simulation depending on the load. The RBTA reconfiguration time is set to 100 ms. For the PrediCLUSTER method, the DBSCAN parameters are set as $MinPts = 4$ and $\varepsilon = 2.5$. Although the individual traffic matrix of both applications has time-varying spatial traffic distribution, the overall traffic matrix varies only slightly. Thus, the DBSCAN results in only one cluster, which leads to one topology extracted by the average TM of the cluster irrespective of the traffic variation. We believe that a larger variation in traffic distributions of multiple applications would result in more clusters, which we leave as a future work.

Fig. 5 (left) shows the overall flow arrival rates and their predicted values for the test dataset. The average mean square error for the overall rate prediction is ~ 0.04, and ToR-to-ToR prediction is ~ 0.001. Fig. 5 (right) shows the flow arrival rate from ToR 0 to 1, and from ToR 3 to 4 and their predicted values. Although the sum of prediction of TM as an overall

rate shows an average of ~ 0.04 deviation, the model is able to predict the spatial traffic variation among ToR pairs with their average values over time. Note that the accuracy of predicted ToR-to-ToR traffic impacts the topology generation and the network performance under the reconfiguration methods.

Fig. 6(left) shows network latency, i.e., average end-to-end data packet latency for the time-varying test dataset. We observe that both reconfiguration methods reduce the packet latency. The percentage improvement in latency by the PredicTER and PrediCLUSTER schemes over the fixed topology is 24.9% and 17.8%, respectively. Furthermore, the packet latency increases when the load increases due to the queuing latency. Fig. 6(right) shows the packet loss rate under the fixed topology and reconfiguration methods. When the load is low or medium, the PredicTER method exhibits momentarily higher packet loss rate compared to other methods due to a larger number (44) of the reconfiguration processes. Interestingly, the PrediCLUSTER method exhibits no or lower loss than the PredicTER method. The reason is that it reconfigures only once in the beginning of the simulation in contrast to the PredicTER method which reconfigures 44 times out of 60 possible instances. Thus, the PrediCLUSTER shows a trade-off in reducing the packet loss at the cost of slightly higher packet latency than the PredicTER. Nevertheless, both methods show better loss performance under the high load, at time $t \sim 1185$s. Notably, there are only 12,305 (37,967) dropped packets under the PrediCLUSTER (PredicTER), which is 33.3% (103.1%) as compared to 36,836 dropped packets under the no reconfiguration in a fixed topology scenario.

V. CONCLUSION

We presented an ML-aided software-defined control plane architecture for dynamic network reconfiguration mechanism in an optical DC/HPC network. We proposed two reconfiguration methods utilizing the predicted traffic under time-varying traffic scenarios. The traffic prediction with the topology extraction and reconfiguration method, in short PredicTER, leads to frequent topology and bandwidth reconfiguration. In contrast, the traffic prediction with clustering for topology extraction and reconfiguration (PrediCLUSTER) reduces the number of reconfigurations at the cost of slightly higher average packet latency. Both methods show that the network reconfiguration is useful in the optical data center and computing networks. However, the reconfiguration does lead to packet loss. As a future work, we plan to implement and investigate the proposed reconfiguration mechanisms on an experimental testbed running one or multiple distributed applications.

ACKNOWLEDGMENT

This work was supported by the NSF ECCS Award #1611560. We thank authors from [9] for sharing traffic data from their testbed.

REFERENCES

[1] R. Cheveresan, M. Ramsay, C. Feucht, and I. Sharapov, "Characteristics of workloads used in high performance and technical computing," in *Proceedings of the 21st annual international conference on Supercomputing*, 2007, pp. 73–82.

[2] X. Guo, X. Xue, F. Yan, B. Pan, G. Exarchakos, and N. Calabretta, "Dacon: a reconfigurable application-centric optical network for disaggregated data center infrastructures," *Journal of Optical Communications and Networking*, vol. 14, no. 1, pp. A69–A80, 2022.

[3] M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," *J. Opt. Commun. Netw.*, vol. 12, no. 4, pp. B44–B54, 2020.

[4] G. Liu, R. Proietti, M. Fariborz, P. Fotouhi, X. Xiao, and S. Ben Yoo, "Architecture and performance studies of 3d-hyper-flex-lion for reconfigurable all-to-all hpc networks," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–16.

[5] R. Proietti, Y. Shang, X. Xiao, X. Chen, Y. Zhang, and S. B. Yoo, "Self-driving reconfigurable silicon photonic interconnects (flex-lions) with deep reinforcement learning," *Supercomputing, Poster*, vol. 118, 2019.

[6] Y. Shen, M. H. Hattink, P. Samadi, Q. Cheng, Z. Hu, A. Gazman, and K. Bergman, "Software-defined networking control plane for seamless integration of multiple silicon photonic switches in datacom networks," *Optics Express*, vol. 26, no. 8, pp. 10 914–10 929, 2018.

[7] S. K. Singh and A. Jukan, "Machine-learning-based prediction for resource (re) allocation in optical data center networks," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D12–D28, 2018.

[8] M. Balanici and S. Pachnicke, "Machine learning-based traffic prediction for optical switching resource allocation in hybrid intra-data center networks," in *Optical Fiber Communication Conference*. Optical Society of America, 2019, pp. Th1H–4.

[9] X. Guo, X. Xue, F. Yan, B. Pan, G. Exarchakos, and N. Calabretta, "Experimental assessment of traffic prediction assisted data center network reconfiguration method," in *2021 European Conference on Optical Communication (ECOC)*. IEEE, 2021, pp. 1–4.

[10] X. Chen, B. Li, R. Proietti, H. Lu, Z. Zhu, and S. B. Yoo, "Deeprmsa: a deep reinforcement learning framework for routing, modulation and spectrum assignment in elastic optical networks," *Journal of Lightwave Technology*, vol. 37, no. 16, pp. 4155–4163, 2019.

[11] M. Wang *et al.*, "Neural network meets dcn: Traffic-driven topology adaptation with deep learning," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 2, 2018.

[12] C.-Y. Liu, X. Chen, Z. Li, R. Proietti, and S. B. Yoo, "Sl-hyper-flex: a cognitive and flexible-bandwidth optical datacom network by self-supervised learning," *Journal of Optical Communications and Networking*, vol. 14, no. 2, pp. A113–A121, 2022.

[13] A. Giannakou, D. Dwivedi, and S. Peisert, "A machine learning approach for packet loss prediction in science flows," *Future Generation Computer Systems*, vol. 102, pp. 190–197, 2020.

[14] M. Zhang, J. Zhang, R. Wang, R. Govindan, J. C. Mogul, and A. Vahdat, "Gemini: Practical reconfigurable datacenter networks with topology and traffic engineering," *arXiv preprint arXiv:2110.08374*, 2021.

[15] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*. Springer, 2003, pp. 44–60.

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[17] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of KDD*, Aug. 1996, pp. 226–231.

[18] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992, Nov. 2000.

[19] S. Kassing, A. Valadarsky, and A. Singla, "Netbench," https://github.com/ndal-eth/netbench, 2016.

[20] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Pfabric: Minimal near-optimal datacenter transport," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, p. 435–446, 2013.

[21] NERSC, "Characterization of the doe mini-apps. https://portal.nersc.gov/project/CAL/doe-miniapps.htm."

[22] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," vol. 40, no. 4, p. 63–74, 2010. [Online]. Available: https://doi.org/10.1145/1851275.1851192