Noname manuscript No.
(will be inserted by the editor)

Virtualized Controller Placement for Multi-Domain Optical Transport Networks using Machine Learning

Sabidur Rahman* · Tanjila Ahmed · Sifat Ferdousi · Partha Bhaumik · Pulak Chowdhury · Massimo Tornatore · Goutam Das · Biswanath Mukherjee

the date of receipt and acceptance should be inserted later

Abstract Optical multi-domain transport networks are often controlled by a hierarchical distributed architecture of controllers. Optimal placement of these controllers is very important for efficient management and control. Traditional SDN controller placement methods focus mostly on controller placement in datacenter networks. But the problem of virtualized controller placement for multi-domain transport networks needs to be solved in the context of geographically-distributed heterogeneous multi-domain networks. In this context, Edge datacenters have enabled network operators to place virtualized controller instances closer to users, besides providing more candidate locations for controller placement. In this study, we propose a dynamic controller placement method for optical transport networks that considers the heterogeneity of optical controllers, resource limitations at edge hosting locations, and latency requirements. We also propose a machine-learning framework that helps the controller placement algorithm with proactive prediction (instead of traditional *reactive* threshold-based approach). Simulation studies, considering practical scenarios and temporal variation of load, show significant cost savings compared to traditional placement approaches.

Keywords optical transport network · optical network controller · cost savings · network virtualization · edge computing · machine learning.

S. Rahman*, T. Ahmed, S. Ferdousi, P. Bhaumik, P. Chowdhury, and B. Mukherjee
University of California, Davis, USA
E-mail: {krahman*,tanahmed,sferdousi,pbhaumik,pchowdhury,bmukherjee}@ucdavis.edu

M. Tornatore
University of California, Davis, USA and Politecnico di Milano, Italy
E-mail: massimo.tornatore@polimi.it

G. Das
Indian Institute of Technology, Kharagpur, India
E-mail: gdas@gssst.iitkgp.ac.in

1 Introduction

Existing proposals for controller placement [1] have focused mostly on packet-switched Software-Defined Networks (SDNs) and they often ignore the complexity, heterogeneity, and vendor specificity of a transport-network control plane. Current technical solutions for transport-network control planes (e.g., Transport SDN, T-SDN) are designed for circuit-switched optical layer and SONET/SDH/OTN layer. T-SDN supports multi-layer, multi-vendor, circuit-oriented networks that are different from packet-based SDN-controlled networks [2]. In our study, we consider a heterogeneous optical transport networks, i.e., optical networks where the devices are heterogeneous (e.g., from different vendors, implementing different technologies flexi-gird vs. fixed-grid, etc.). Hence, to control such a network, we need a heterogeneous set of optical network controllers, with different vendor-specific or technology-specific capabilities. Moreover, the control plane for optical transport networks employs a hierarchical distributed architecture [3] comprising of such heterogeneous (often vendor-specific) Optical Network (ON) Controllers (ONCs).

Our study considers that T-SDN controllers can be deployed as virtualized controller instances (as in [4]). Virtualized controller placement has many benefits. First, manually deploying ONCs in traditional 'hardware boxes' can take several days, compared to few minutes in case of virtualized instances (hosted on Virtual Machines (VMs), docker containers, etc.). Such virtual instances are hosted in cloud datecenter (DCs) or in computing nodes at edge datacenters (Edge-DCs) (such as Network Function Virtualization Infrastructure Points of Presence (NFVI-PoPs), metro datacenters (DCs), or Central Offices Re-architected as Datacenters (CORDs), etc.). Second, virtualized controllers can be easily recovered from failures or disasters using the backed-up/replicated virtual copy of the controllers. These instances can be easily moved from one location to another and can be redeployed [5] without significant down time. Third, operational cost savings for network operators and leasing cost savings for network leasers are other important motivations toward virtualization.

Prior studies exploring static [6] [7], and dynamic [9] controller-placement problems mostly focused on packet-based SDN controllers and DC networks [10]. But, as we discuss in Section 2, methods proposed in SDN and DC scenarios are often not applicable and not optimal for heterogeneous optical transport networks.

To the best of our knowledge, our study is the first to propose dynamic placement of controllers for heterogeneous, multi-domain transport networks comprising heterogeneous ONCs, considering the complexity due to virtual instances hosted jointly on DCs and Edge-DCs (e.g., NFVi-PoPs). We explore the technical details of the dynamic controller-placement problem (e.g., latency requirements, resource limitations at Edge-DCs, controller capacity limitations, etc.), propose the Virtualized Controller Deployment Algorithm (VCDA), and report illustrative results.

In addition, we observe that the *reactive* threshold-based approach used, e.g., in our work in [11] and others' [9], do not benefit from knowledge of historical traffic patterns and data. Hence, we propose a machine learning (ML) based method that leverages knowledge of historical data and can be used to forecast the required number of controllers during next time interval. As we will elaborate in this study, the ML method for controller-placement problem is not trivial, especially due to complexity in modeling domain-specific controller traffic and in devising a conversion from controller traffic to required number of controller.

Significant contributions of our study in the controller placement problem are as follows:

1. To the best of our knowledge, our study is the first to propose dynamic placement of controllers for heterogeneous, multi-domain transport networks comprising heterogeneous ONCs.
2. We propose a ML framework that identifies the key parameters, and models the conversion of traffic data to control traffic for the controller-placement problem.
3. We demonstrate that dynamic load-aware controller placement (powered by the proposed ML framework) can achieve significant accuracy, and enables relevant cost savings and QoS improvement.

This study is organized as follows. Section 2 reviews prior work on controller placement problems. Section 3 discusses the control-plane architecture. Section 4 provides a formal problem statement and describes the proposed solution method, including the ML framework and cost models. Section 5 discusses numerical results on accuracy of the ML framework, on cost savings and QoS improvement. Section 6 concludes the study.

2 Background and Related Work

The controller-placement problem is known to be NP-hard [1]. In the context of SDN controllers, both static [6] [7], and dynamic [9] placement problems have been explored. But most studies on SDN Controller Placement Problems (CPPs), e.g., [6]- [8], consider placement of only controller 'middle-boxes', not virtualized instances. Ref. [15] considers recovery of SDN controllers in a disaster scenario. These early studies do not consider the additional complexities due to virtualization, delay constraints, hosting location constraints, etc.

Recent studies [9] [10] on Elastic Control Placement (ECP) for SDN controllers discuss threshold-based methods to dynamically resize the ECPs. Ref. [9] focuses on DC networks managed through homogeneous SDN controllers, to minimize control-plane resizing delay. Ref. [9] focuses on the 'switch-to-controller' mapping, ensuring that each switch (forwarding plane) is connected to at least one controller (control plane). Incoming traffic-flow requests originated inside a domain will be served by the same controller; and controller capacity limit is preserved. When a new traffic-flow request arrives, the 'switch' depends on the controller for routing and path-computation decisions.

Recently, control plane architectures [12] [16] in T-SDN have been proposed to accommodate multiple heterogeneous network domains and associated domain-specific ON controllers. However, the design of T-SDN control plane architecture is still evolving and is an active research area [2] [16]. Ref. [4] was among the first to propose a virtualized control-plane architecture for transport networks, but not from a placement perspective.

Refs. [9] [10] consider DC placement (of controller instances), which is practical for a DC network scenario. But, for a transport network with distributed heterogeneous domains, we also consider joint deployment in DCs and Edge-DCs. This introduces new constraints such as host location capacity, and inter-domain and intra-domain communication delays (which [9] [10] do not consider). To the

best of our knowledge, none of the prior studies on controller placement explores impact of *proactive* prediction of number of controllers using machine learning or other methods.

3 Control Plane Architecture

Fig. 1 shows an example of a hierarchical control plane for heterogeneous transport networks. Domain-specific controllers are connected with 'parent controller(s)', which are connected to the 'application plane' (e.g., Transport Network Orchestrator (TNO), Operations Support Systems (OSS), etc.).

'Domain Controllers' (ONCs in the figure) are responsible for the communication between 'control plane' and 'data plane'. Different autonomous domains, depending on the underlying ONC, use specific type of controllers and protocols to control the 'data plane' switches. Fig. 1 shows two types of domains, i.e., Fixed-grid and Flex-grid domains. This example architecture can be extended to support more variations of domain controllers and associated technologies. In this study, we consider two levels of hierarchy and assume that the parent controller location is already determined. Hence, our proposed method focuses solely on the placement of domain controllers.

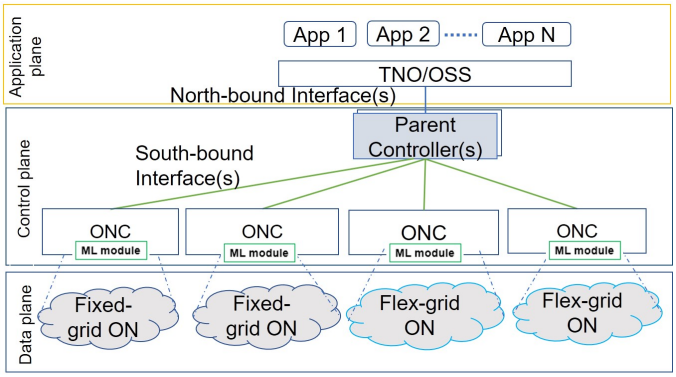

Fig. 1 Control-plane architecture for heterogeneous transport networks with ML module.

In the considered control-plane architecture, dedicated ML modules are associated to each domain. In fact, each domain can have separate load patterns, load types, and variation trend in load. To capture this knowledge, we propose to use these distributed ML modules to power the controller placement algorithm proposed in the next section. The choice of per-domain ML is not trivial. In some scenarios, one ML module is enough to predict network-wide results. Then, in other cases, deploying ML modules in a more distributed fashion than per-domain level (e.g., per device level, per city/population center level) can be useful. In our use-case, 'per-device ML module' will be a overkill, as control traffic from that domain will arrive to controllers (making per-device control-traffic intelligence redundant).

4 Problem Statement and Solution Method

4.1 Problem Statement

The dynamic 'on-demand' controller-placement problem can be defined as follows: Given a topology, a set of controller hosting locations with limited capacity, arrival rate of traffic flows, a set of heterogeneous network domains, controller capacity, and constraints, deploy optimal number of controllers to satisfy all the domains, minimizing the leasing costs.

In order to decide on how many number of controllers we need in a given time interval, in Ref. [11], we proposed a traditional threshold-based approach (also explored in [9] for datacenter network scenarios). In this study, we explore if ML-assisted prediction can help to optimize the performance of our proposed solution method. Hence, we propose a new ML-based a prediction mechanism, demonstrating its superior performance with respect to threshold-based approaches.

4.2 Input Parameters and Variables

- $G(V, E)$: Optical transport network topology where V is set of domains and E is set of inter-domain links.
- M_v: set of controllers serving domain v.
- S_v: set of switches in domain v serving traffic.
- T_v: domain-specific controller types (flexi-grid or fixed-grid).
- H_v: controller hosting location where $H_v \subseteq V$.
- χ_v^t: total compute capacity at v.
- χ_v^u: used compute capacity at v.
- ω_v^t: total memory capacity at v.
- ω_v^u: used memory capacity at v.
- T_v^μ: service capacity limit (i.e., maximum number of new routing requests served per second) for controller type T_v.
- T_v^χ: compute resource requirement for controller type T_v.
- T_v^ω: memory resource requirement for controller type T_v.
- $\lambda(v)$: arrival rate of new traffic flows for a given domain (v), where r_v represents new request for flow routing (to be processed by the controller). S_r is the switch at which the request has arrived, and M_v gives the set of domain controllers the data traffic switch (S_r) is connected.
- α: latency constraint (maximum allowed delay from switch to controller).

In this study, switch stands for optical data plane switch, where the new routing requests arrives. Then, the switch depends on the domain controller for making the routing decision for the new request [2]. If the domain controller is placed far away from the switch, the decision making will be delayed. Hence, we need a constraint (α) to enforce that.

4.3 Constraints

We consider the following constraints:

1. **Latency constraints**: Controllers must be placed within the allowed latency limit, i.e., switch-to-controller and controller-to-switch delay, including processing delay must not exceed the allowed delay limit:

$$D(s,h) + D_p + D(h,s) \leq \alpha; \forall s \epsilon S_v, \ \forall h \epsilon V \tag{1}$$

   where function $D(x,y)$ represents transmission, propagation, and processing delay between points x and y, s is switch where the flow originated, h is controller hosting location, and D_p is processing delay at controller instances M_v.
2. **Controller type constraint**: To reflect domain heterogeneity, we enforce the following controller type constraints:

$$T_v == M_v^t; \ \forall v \epsilon V \tag{2}$$

   where the constraint enforces that all controller instances of v (M_v^t) match required controller type T_v.
3. **Controller capacity constraints**: Deployed controllers must have enough capacity to support domain switches:

$$\sum_{s \epsilon S_v} \lambda(s) \leq T_v^{\mu} * |M_v|; \ \forall v \epsilon V \tag{3}$$

4. **Controller host capacity limit**: Hosting location ($v.h$) must have both compute and memory capacity:

$$\sum_{g \epsilon V} |M_g| * T_g^{\chi} \leq \chi_v^t; \ H_g == v; \forall v \epsilon V \tag{4}$$

$$\sum_{g \epsilon V} |M_g| * T_g^{\omega} \leq \omega_v^t; \ H_g == v; \forall v \epsilon V \tag{5}$$

4.4 Cost Models

4.4.1 Leasing cost for virtual controller instances

We consider that network operators lease capacity from DC operators. Virtual instance leasing cost C_C can be stated as:

$$C_C = \sum_{v \epsilon V} (|M_v| * (\gamma_h + \gamma_s) * d) \tag{6}$$

where γ_h is per-unit hardware cost for per unit time, γ_s is per-unit software cost for per unit time, and d is duration of operation.

4.4.2 QoS degradation penalty

The quality of service is negatively impacted when the controller-placement method fails to deploy enough number of controller instances. We model this cost by converting the duration (d_q) in degraded QoS to a penalty cost:

$$C_q = \sum_{v \epsilon V} (|M_v| * \gamma_q * d_q) \tag{7}$$

where γ_q is the per unit-time penalty for degraded service.

4.5 Proposed Machine-Learning Framework to Forecast Controller Demand

Fig. 2 a) shows the components inside the 'ML module' (in Fig. 1). A brief description of the components follows:

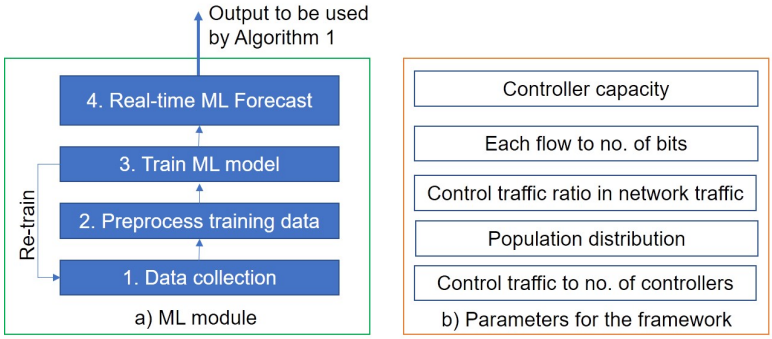

Fig. 2 Proposed ML framework: a) ML module deployed per domain; and b) key parameters used in designing the ML module.

- **Data collection:** For each domain, data is collected from network traffic. For this study, we use realistic traffic load traces from [17]. This traffic load data (in bits) was collected at every five-minute intervals over a 1.5-month period from a private ISP and on a trans-Atlantic link.
- **Preprocess training data:** In many cases the collected data is not immediately applicable for training the ML model. The data needs to go through certain steps to be ready as training data. In this study, to make the data useful for the controller-placement problem, it has to go through two major conversion: 1) domain-specific controller traffic and 2) controller traffic to number of controller conversion. We discuss these two procedures later in this subsection.
- **Train ML model:** In this study, we have used supervised ML techniques to train the models. In supervised ML, we can associate each *instance* (set of *features*) of the problem to a *class*. In our training model, we have used the 22 features defined in [14], which capture the variation of traffic in collected data. As *class* (output) that associates with the *features* (input), we have used 'number of controllers' for a given time interval. This means that we train the ML model to learn from the mapping between features/patterns inside the traffic data and how many 'number of controllers' can handle that traffic from future time interval. Intuitively, 'maximum traffic flow (Mbps)' can be an option for class/output. But, traffic flow in Mbps would require a very high number of classes, leading to poor performance. Hence, we use less granular output (i.e., number of controllers), with less number of classes, leading to good prediction performance.
- **Real-time ML recommendation:** Finally, in this step, the domain controllers can start receiving the predictions made by the ML module (see Algorithm 1).

In order to keep the ML model updated, steps 1 to 3 are re-executed in an interval (e.g., every 7 days) suitable for the domain and the operator.

Fig. 2 b) shows the key parameters used by this framework:

- **Domain-specific controller traffic generation:** Ideally, the 'ML module' will use control traffic data collected from respective domain controller. But the datasets we have access to consist of only network (data-plane) traffic data. To mimic domain specific control traffic, we converted the dataset [17] using these two parameters: *i)* control traffic in percentage (%) of network traffic and *ii)* population distribution. It is not trivial to answer the question: 'what percentage of the network traffic is control traffic?' Based on existing studies, we can estimate that, depending on the network size, control traffic overhead can be 1%-10% [18]. Also, the population distribution varies from domain to domain, impacting the control traffic generated. To incorporate this aspect in our study, we use population density of four time zones in USA, to create different sets of control traffic at respective domains. In the results section, we report our findings on impacts of these two parameters.
- **Controller traffic to number of controller conversion:** Conversion from controller traffic (Γ in bps) to number of controllers is not trivial. The controller traffic (measured or converted as we did) is in bits per second (bps) unit. On the other hand, the controller capacity (T_v^μ) is often reported in number of flows [8] (not in bps). In order to know how many number of controller instances are required to serve a certain bps traffic, we need these two parameters: controller capacity in number of flows and how many bits are typically communicated in each flow request (b). Based on the controller communication steps described in [4], we can identify the number of bits used by each control flow (b). Hence, number of controllers (c) can be derived from the following equation:

$$c = \Gamma/(T_v^\mu * b) \tag{8}$$

Once the ML framework is complete, we train different modules using domain-specific set of data. Then, the ML modules are ready to be used by Algorithm 1 (proposed in next subsection).

Fig. 2 depends on the control traffic ratio in network data traffic. Now, it is safe to consider that the optical network operator will know this ratio in their networks and will be able to use this parameter for our proposed method. This ratio will vary depending on the network. Hence, for this study, we explored the literature to gather some insights on the range of this ratio, instead of an exact number. Our search yielded in this study [18] which reports the ratio can be 1%-10%. We also did sensitivity analysis over this range and ensured that the method performs well even after changes in this parameter.

4.6 Algorithm

We propose a polynomial-time heuristic, called Virtualized Controller Deployment Algorithm (VCDA), as a scalable solution for a heterogeneous optical transport network (see Algorithm 1). Since turning controllers on/off too often may lead to instabilities, we introduce a decision epoch (e), i.e., a dynamic variable allowing network operators to tune the decision frequency. We also use two management entities: Network Management and Orchestration (NMO) and Distributed Cloud Management (DCM). NMO takes care of load balancing and assignment

of switches and traffic with the controllers. DCM takes care of the cloud resource management for placement of controller instances.

Algorithm 1 Virtualized Controller Deployment Algorithm (VCDA)

1: **Input:** $G(V, E), \lambda(v), \alpha, e$;
2: **for each** domain v **in** V **do**
3: ▷ Forecast/calculate required number of controllers
4: $c \leftarrow$ predicted by ML or calculated using threshold-based method;
5: **if** $|M_v| == c$ **then**
6: Consolidate and load balance traffic flows among the M_vs using Eqns. (1-5);
7: ▷ deploy more controllers
8: **else if** $c > |M_v|$ **then**
9: $h \leftarrow H_v$;
10: $\delta \leftarrow c - M_v$;
11: ▷ enough resources at h
12: **if** $(\chi_h^t - \chi_h^u) > \delta * T_v^\chi$ & $(\omega_h^t - \omega_h^u) > \delta * T_v^\omega$ **then**
13: Turn on additional δ controllers (T_v type) at location h;
14: Load balance and re-route among the M_vs using Eqns. (1-5);
15: ▷ not enough resources at h
16: **else**
17: $h^{'} \leftarrow$ find optimum location to host c using Eqns. (1-5);
18: Allocate c controllers (T_v type) at $h^{'}$ via **DCM**;
19: Migrate all M_v instances to $h^{'}$ via **DCM**;
20: Turn on δ additional controller instances via **DCM**;
21: Load balance and re-route flows among the controller instances;
22: Turn off M_v controllers at h via **DCM**;
23: **end if**
24: ▷ remove extra resources
25: **else if** $c < |M_v|$ **then**
26: $\delta \leftarrow c - M_v$;
27: DCM finds optimum δ controllers to turn off;
28: Re-route and load balance among the M_v using Eqns. (1-5);
29: Turn off δ controllers;
30: **end if**
31: **end for**
32: **if** e is expired **then**
33: go to line 2;
34: **end if**

Our algorithm ensures that, for each domain, enough controllers are deployed to serve current load, observing the constraints. In a threshold-based approach, we use the current threshold value to calculate required number of controllers for next epoch. In an ML-assisted approach, we consult the 'ML module' which forecasts the number of controllers we will require for next epoch e. At a given epoch, if the controller capacity constraint holds, it means that we do not need additional controller instances (line 5). But, if the controller capacity constraint fails (line 9), the algorithm checks if the current hosting location (h) has enough compute and memory capacity to host the additional δ controllers (line 13). If yes, we turn on additional controllers and load balance the switches and traffic flows (line 14-17). If host location h does not have enough resources, the algorithm finds the next optimal location to host all the instances (line 20) following constraints as in Eqns. (1-5) and minimizing Eqn. (14). In this step, we utilize the benefits of consolidation in computing. Placing controllers from the same domain closer to

each other will reduce delay cost (Eqn. (12)). We consider live VM migration (line 18) to relocate the already-running controller instances to the new location with least interruption of services.

The algorithm turns off the extra controllers (line 25-29) to save operational cost. In line 27, we keep at least one controller instance running for each domain. Installation of a new virtual machine hosting the controller instance may take time in the range of 100 seconds [5]. However, it does not mean that we do not have any controller capability during the installation of novel VM instances, as we consider that at least one instance of a domain controller is always present. After each iteration, the algorithm waits for the epoch e to expire.

The run-time complexity of VCDA depends on number of domains ($|V|$), maximum number of controllers ($max(|M_v|)$), maximum number of switches in a domain ($max(|S_v|)$), and number of host locations ($|H_v|$). The run-time complexity of VCDA can be expressed as $O(|V|*max(|M_v|)*max(|S_v|)+|V|*|H_v|*max(|M_v|)*max(|S_v|)$.

5 Illustrative Numerical Examples

We present illustrative results on a US-wide topology (see Fig. 3), with heterogeneous domains and Edge-DCs/DCs. Each network domain requires domain-specific controller(s), that are connected to other domains via backbone optical links. We consider two DC locations (2 and 13) and three Edge-DCs (5, 7, 10) to host controller instances. Capacities of DCs, racks, and servers vary significantly in practice. For Edge-DCs, we assume total compute capacity (χ_v^t) is 64 units and total memory capacity (ω_v^t) is 256 GB. For DCs, we consider 15000 compute and 30000 memory capacity (to represent significantly large capacity). For illustrative examples in this section, let α = 15 ms, T_v^{μ} = 2500 requests per second [8], per-controller instance compute requirement (T_v^{χ}) = 2 compute units [9], memory requirement (T_v^{ω}) = 4 GB, γ_h = \$0.01 per controller per hour, γ_s = \$0.02 per controller per hour and γ_q = \$0.05 for every 10 minutes of degraded service [14].

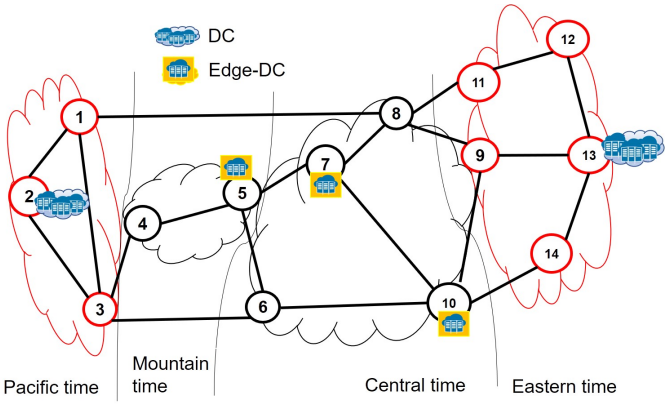

Fig. 3 Example optical network topology with controller host locations (DCs and Edge-DCs) and heterogeneous domains (color red and black denote separate controller types).

5.1 Accuracy of ML Framework

In this part of the study, we report the performance of the proposed ML framework for different scenarios. As the ML algorithm, we use Random Forest (best-performing algorithm in [14]). We consider $e = 10$, i.e., VCDA algorithm asks for ML forecast every 10 minutes, and we use 40 days of data ($40 * 144 = 5760$ instances), of which 50% is training and rest 50% is used for testing. As the data was collected every 5 minutes, we have 288 instances of data in 24 hours. Then, the prediction decision is taken every 10 minutes, leading to 144 instances in a day.

We consider 47% of US population is in Eastern, 29% in Central, 7% in Mountain, and 17% in Pacific time zone domains. Our study uses this data to generate domain-specific controller traffic (discussed in Section 4.5). We report the ML accuracy for 1% control traffic (Table 1) and 2% control traffic (Table 2).

In Tables 1 and 2, we present three different performance metrics: a) **Precision (%)**: corresponds to the fraction of predicted positives which are in fact positive. Precision is given by percentage of: $TP/(TP+FP)$. b) **False Positive (FP)(%)**: FP is an important indication of ML classifiers as lower FP indicates less classification mistakes. c) **ROC area (%)**: Receiver Operating Characteristic (ROC) curve is a graphical plot in which true positive rate ($TP/((TP + FN))$ is plotted as function of the false positive rate ($FP/(FP + TN)$). Here, TP implies true positive, FN is false negative, and TN is true negative. ROC area is a robust metric for ML classifier performance evaluation. A higher ROC area value means more robust ML prediction model.

Table 1 Accuracy of the proposed ML framework (1% control traffic).

Domain	Precision (%)	False Positives (%)	ROC Area (%)
Eastern	96.7	1.2	99.7
Central	96.8	1.0	99.8
Mountain	100.0	0.0	100.0
Pacific	97.8	1.4	99.9

Table 1 shows promising accuracy of our method. The lowest Precision is achieved for Eastern (96.7%), going up to 100.0% for Mountain. This can be explained with the population ratio. As Eastern has 47% of total population, it generates more dynamic traffic, resulting in lower forecasting accuracy. In comparison, Mountain with only 7% of population, generates less dynamic traffic, resulting in higher accuracy. This argument is further supported by Table 2 where 2% control traffic (i.e., twice as much control traffic for all domains). This increase in control traffic brings down the prediction accuracy to 88.7% (for Eastern). Domains with lower population (producing less dynamic traffic) show higher precision: Mountain (99.2%) and Pacific (95.7%). Other performance metrics (false positives and ROC area) show similar trends in accuracy. Specially, ROC area consistently stays over 99%, showing the robustness of the proposed model.

Fig. 4 a) shows the controller variation over 24 hours for the sample scenario of the Eastern domain. We compare our ML method with threshold-based (TH) method [9]. We observe that ML method is faster to capture the future changes

Table 2 Accuracy of the proposed ML prediction framework (2% control traffic).

Domain	Precision (%)	False Positives (%)	ROC Area (%)
Eastern	88.7	1.5	99.4
Central	93.3	1.3	99.5
Mountain	99.2	1.2	99.9
Pacific	95.7	1.4	99.7

than the TH method (orange line with crosses trails the blue line). We also indicate the situations where TH does not provide enough controllers to support QoS (yellow dashed circles). We also indicate situations where TH deploys more resources than required (using purple dashed circles). Fig. 4 b) shows the controller variation over 24 hours for Pacific domain. As discussed earlier, due to population, the load variation is less, resulting in less variation in number of controllers as well.

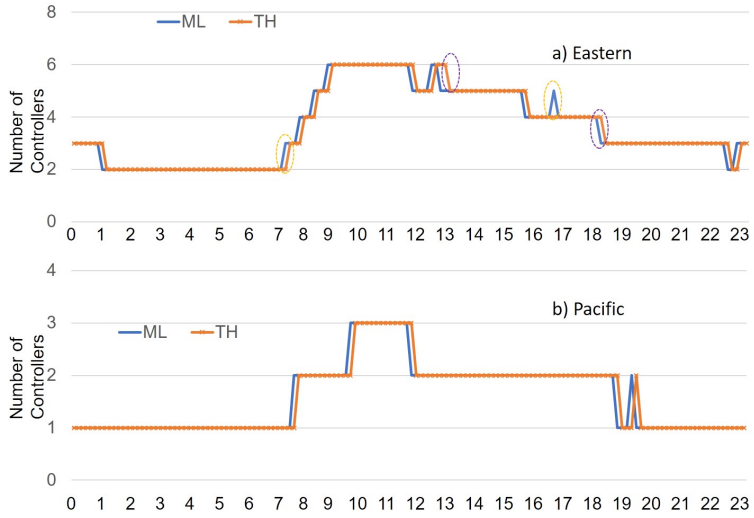

Fig. 4 Illustrative 24-hour variation of number of controllers using machine learning (ML) and threshold-based (TH) approaches with 1% control traffic.

In Fig. 5 a), we present 24-hour variation of controllers for Eastern domain considering 2% control traffic. With the increased traffic, we observe more variation and higher number of controllers. Similarly in Fig. 5 b), Pacific domain experiences more traffic compared to Fig. 4 b).

5.2 Impact on Cost Savings, Electricity Consumption, and QoS

Fig. 6 explores the sum of hardware cost and software leasing cost (using Eqn. 6) and QoS penalty cost (using Eqn. 7) for ML and TH approaches. We compare three methods: a) Static controller placement (realizing prior study [7] in

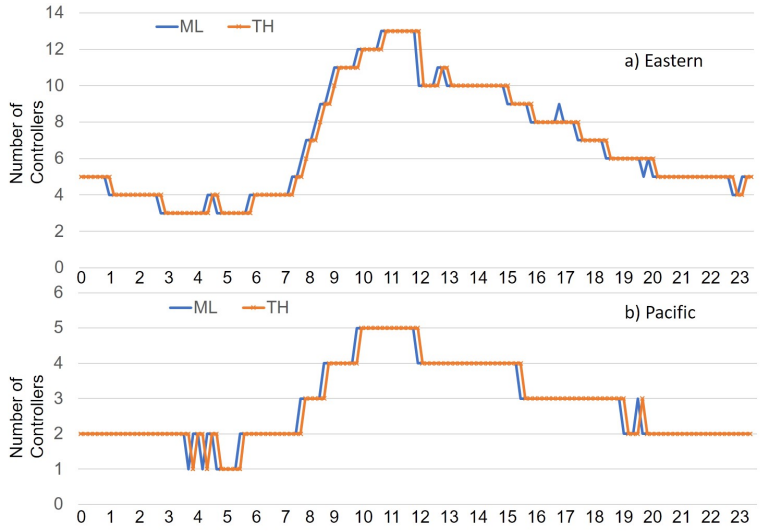

Fig. 5 Illustrative 24-hour variation of number of controllers using machine learning (ML) and threshold-based (TH) approaches with 2% control traffic.

transport network scenario), b) VCDA-TH (mimicking prior study [9] in transport network scenario), and c) VCDA-ML (implementing VCDA powered by the ML framework). VCDA-ML shows the highest cost savings (45.54%) compared to the VCDA-TH approach (39.32%). In a large network where cost is in millions of dollars, our proposed methods can help significantly.

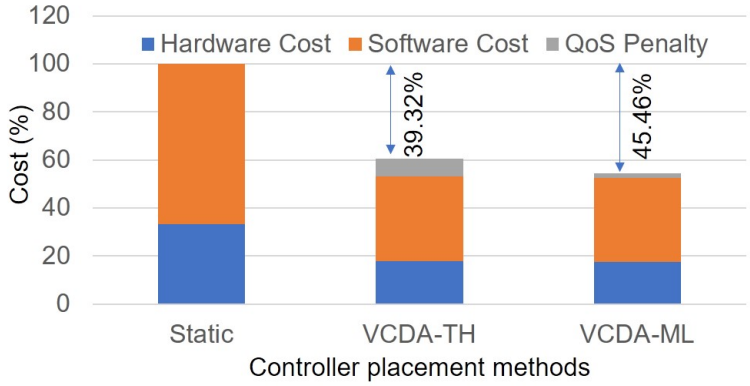

Fig. 6 Cost (%) from VCDA with threshold-based and ML-assisted methods, compared to the static controller placement.

We also consider electricity consumption (and related cost) and report the results in Fig. 7. A large portion of DC/Edge-DC and network operational costs come from electricity cost. For controller power consumption, we borrow electric-

ity consumption model for servers from Ref. [19]. We also consider four different virtualization technologies (Xen, KVM, Docker, and LXC) and use practical power consumption values reported in Ref. [20]. We observe that the ML-assisted method yields least electricity consumption compared to VCDA-TH and Static placement. We also observe that container-based technologies (Docker and LXC) use lower electricity compared to Xen and KVM.

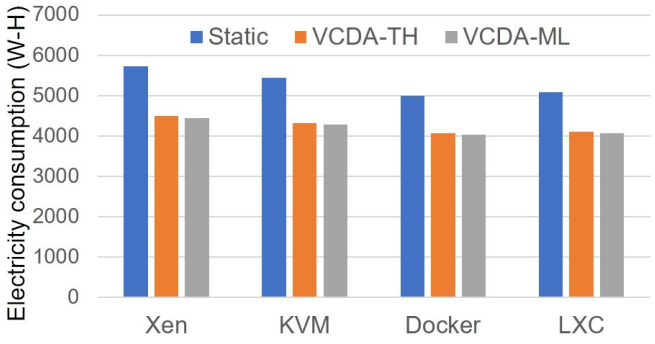

Fig. 7 Electricity consumption saving (Watt-Hour) from VCDA with threshold-based and ML-assisted methods, compared to the static controller placement.

5.3 Daily Shift of 'Center of Gravity' of Controllers

As the sun moves from East to West, due to dynamic controller placement, we observe that the 'center of gravity' of the controllers also shifts. Fig. 8 shows this shifting phenomena. In the early morning (6AM Eastern time), Eastern domain has 2 controllers and the rest of the domains each has one controller, i.e., the 'center of gravity' is clearly on the Eastern domain. But, as the day progresses, more controllers 'light-up' in Central and Pacific domains, shifting the 'center of gravity' towards the middle of the country. In the afternoon (3PM), the Eastern domain has 5 controllers, Central has 3, Mountain 1, and Pacific 3, shifting the 'center' to the Central domain. As Eastern (47%) population ratio is very high compared to Mountain (7%) and Pacific (17%), the 'center' does not move past the Central domain. We observe similar trends for 2% control traffic (with higher number of controllers).

6 Conclusion

Virtualized controller placement in multi-domain heterogeneous optical transport networks introduces new challenges for network management. Our proposed method for controller placement considers transport-network-specific properties and constraints such as heterogeneous controller types, resource limitations at

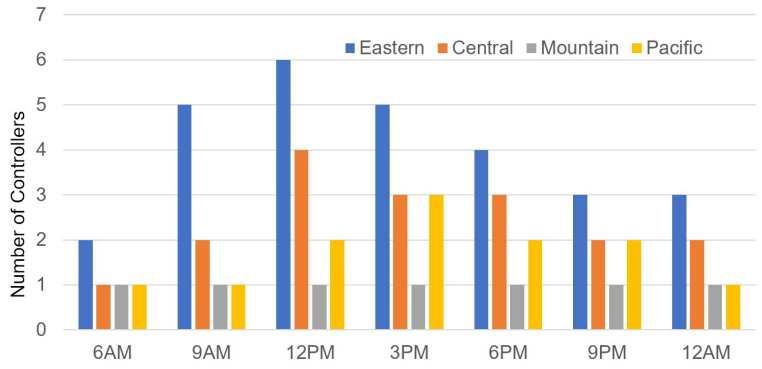

Fig. 8 'Center of gravity' for controllers shifting as the day progresses (1% control traffic).

edge-hosting locations, etc. In addition, ML-based prediction helps to improve the performance of the proposed algorithm. Illustrative examples show that our proposed method saves cost and reduces delays significantly, compared to traditional approaches. Future studies should explore variation of compute/memory requirements, variation of Edge-DC capacities, variation of Edge-DC and DC locations, different prediction methods, and more detailed cost models.

Acknowledgement

This work was supported by NSF Grant No. 1716945.

References

1. G. Wang et al., "The controller placement problem in software defined networking: a survey," *IEEE Network*, vol. 31, no. 5, pp. 21-27, 2017.
2. R. Alvizu et al., "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Comms. Surveys & Tutorials*, vol. 19, no. 4, pp. 2232-2283, 2017.
3. V. Lopez et al., "Control plane architectures for elastic optical networks," *J. of Optical Comms. and Net.*, vol. 10, no. 2, pp. 241-249, Feb. 2018.
4. R. Muoz et al., "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks," Journal of Optical Comms. and Net., vol. 7, no. 11, pp. 62-70, 2015.
5. S. Rahman et al., "Dynamic Workload Migration over Optical Backbone Network to Minimize Data Center Electricity Cost," *IEEE Trans. Green Comms. and Net.*, vol. 2. no. 2, pp. 570-579, Dec. 2017.
6. B. Heller et al., "The Controller Placement Problem," *Proc. 1st Wksp. Hot Topics in Software Defined Networks*, pp. 712, 2012.
7. G. Yao et al., "On the Capacitated Controller Placement Problem in Software Defined Networks," *IEEE Commun. Letters*, vol. 18, no. 8, pp. 1339-1342, 2014.
8. A. Sallahi et al., "Optimal Model for the Controller Placement Problem in Software Defined Networks," *IEEE Comms. Letters*, vol. 19, no. 1, pp. 30-33, 2015.
9. W. Kim et al., "T-DCORAL: A Threshold-based Dynamic Controller Resource Allocation for Elastic Control Plane in Software-Defined Data Center Networks," *IEEE Comms. Letters,* Nov. 2018.
10. A. Potluri et al., "An efficient DHT-based elastic SDN controller," *Proc., 9th Intl. Conf. on Comm. Sys. and Net.*, pp. 267-273, Jan. 2017.
11. S. Rahman et al., "Virtualized Controller Placement for Multi-Domain Optical Transport Networks," *ONDM*, 2019.

12. A. Aguado et al., "ABNO: A feasible SDN approach for multivendor IP and optical networks," *IEEE/OSA J. Opt. Comms. Net.*, vol. 7, no. 2, pp. A356-A362, Feb. 2015.
13. S. Rahman et al., "Dynamic Controller Deployment for Mixed-Grid Optical Networks," *Proc. Asia Comms. and Phots. Conf.*, 2018.
14. S. Rahman et al., "Auto-Scaling VNFs Using Machine Learning to Improve QoS and Reduce Cost," *Proc., IEEE Intl. Conf. on Commun.*, May 2018.
15. S. S. Savas et al., "Disaster-resilient control plane design and mapping in software-defined networks," *Proc., 16th IEEE Intl. Conf. on High Performance Switching and Routing,* July, 2015.
16. R. B. Lourenco et al., "Robust hierarchical control plane for Transport Software-Defined Networks," *Optical Switching and Net.,* vol. 30, pp. 10-22, Nov. 2018.
17. T. P. Oliveira et al., "Computer network traffic prediction: a comparison between traditional and deep learning neural networks," *Int. J. Big Data Intelligence*, vol. 3, no. 1, pp. 28-37, 2016.
18. D. Bhamare et al., "Models and algorithms for centralized control planes to optimize control traffic overhead," Computer Communications, vol. 70, no. 1, pp. 68-78, 2015.
19. S. Rahman et al., "Dynamic Workload Migration over Optical Backbone Network to Minimize Data Center Electricity Cost," *IEEE Transactions on Green Comms. and Net.*, vol. 2, no. 2, pp. 570-579, 2018.
20. R. Morabito, "Power Consumption of Virtualization Technologies: an Empirical Investigation," *8th IEEE/ACM Intl. Conf. on Utility and Cloud Computing*, pp. 522-527, 2015.