



Virtualized controller placement for multi-domain optical transport networks using machine learning

Sabidur Rahman¹ · Tanjila Ahmed¹ · Sifat Ferdousi¹ · Partha Bhaumik¹ · Pulak Chowdhury¹ · Massimo Tornatore^{1,2} · Goutam Das³ · Biswanath Mukherjee¹

Received: 4 November 2019 / Accepted: 17 June 2020 / Published online: 10 July 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

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

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 multilayer, 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-grid 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]

✉ Sabidur Rahman
krahman@ucdavis.edu

Tanjila Ahmed
tanahmed@ucdavis.edu

Sifat Ferdousi
sferdousi@ucdavis.edu

Partha Bhaumik
pbhaumik@ucdavis.edu

Pulak Chowdhury
pchowdhury@ucdavis.edu

Massimo Tornatore
massimo.tornatore@polimi.it

Goutam Das
gdas@gssst.iitkgp.ac.in

Biswanath Mukherjee
bmukherjee@ucdavis.edu

¹ University of California, Davis, USA

² Politecnico di Milano, Milan, Italy

³ Indian Institute of Technology, Kharagpur, Kharagpur, India

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 datacenters (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 downtime. 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 [8] controller placement problems mostly focused on packet-based SDN controllers and DC networks [9]. But, as we discuss in Sect. 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 [10] and others’ [8], 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 hetero-

geneous, 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 [8] placement problems have been explored. But most studies on SDN Controller Placement Problems (CPPs), e.g., [6, 7, 11], consider placement of only controller ‘middle-boxes’, not virtualized instances. Reference [12] 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 [8, 9] on Elastic Control Placement (ECP) for SDN controllers discuss threshold-based methods to dynamically resize the ECPs. Reference [8] focuses on DC networks managed through homogeneous SDN controllers, to minimize control-plane resizing delay. Reference [8] 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 [13, 14] 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, 14]. Ref. [4] was among the first to propose a virtualized control-plane architecture for transport networks, but not from a placement perspective.

References [8, 9] 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 [8, 9] 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

Figure 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. Figure 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.

In the considered control plane architecture, dedicated ML modules are associated with 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

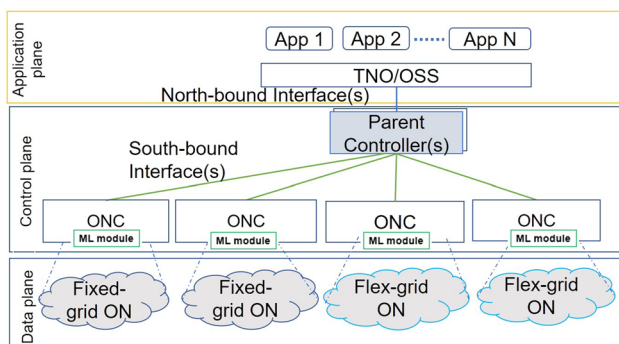


Fig. 1 Control plane architecture for heterogeneous transport networks with ML module

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. [10], we proposed a traditional threshold-based approach (also explored in [8] 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:

$$D(s, h) + D_p + D(h, s) \leq \alpha; \forall s \in S_v, \forall h \in V \quad (1)$$

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:

$$T_v = M_v^t; \forall v \in V \quad (2)$$

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:

$$\sum_{s \in S_v} \lambda(s) \leq T_v^u * |M_v|; \forall v \in V \quad (3)$$

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

$$\sum_{g \in V} |M_g| * T_g^x \leq \chi_v^t; H_g == v; \forall v \in V \quad (4)$$

$$\sum_{g \in V} |M_g| * T_g^w \leq \omega_v^t; H_g == v; \forall v \in V \quad (5)$$

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:

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

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:

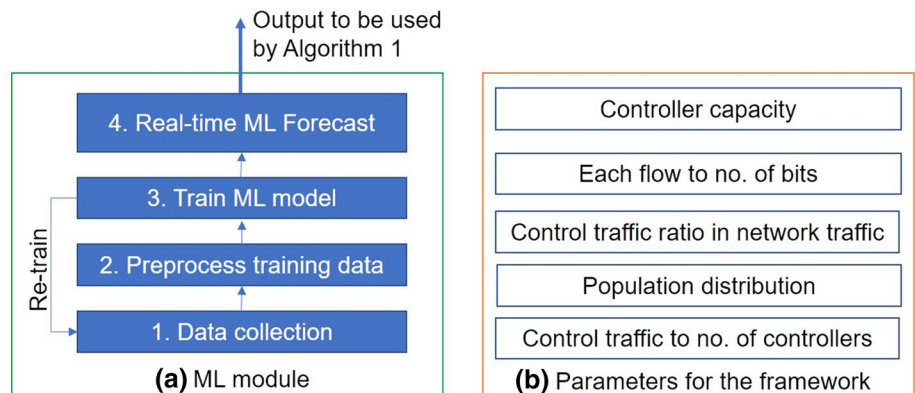
$$C_q = \sum_{v \in V} (|M_v| * \gamma_q * d_q) \quad (7)$$

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

4.5 Proposed machine learning framework to forecast controller demand

Figure 2a 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 [15]. 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 [16], 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.

Figure 2b 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 [15] 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% [17]. Also, the population distribution var-

ies 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 [11] (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:

$$c = \Gamma / (T_v^\mu * b) \quad (8)$$

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).

Figure 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 into the range of this ratio, instead of an exact number. Our search yielded in this study [17] 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.

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 Eqs. (1–5) and minimizing Eqs. (6) and (7). In this step, we utilize the benefits of consolidation in computing. Placing controllers from the same domain closer to each other will reduce penalty cost (Eq. 7). 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 runtime 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 $O(|V| * \max(|M_v|) * \max(|S_v|) + |V| * |H_v| * \max(|M_v|) * \max(|S_v|)$.

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  $M_v$ s using Eqs. (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  $(\chi_h^t - \chi_h^u) > \delta * T_v^X$  &  $(\omega_h^t - \omega_h^u) > \delta * T_v^w$  then
13:      Turn on additional  $\delta$  controllers ( $T_v$  type) at location  $h$ ;
14:      Load balance and re-route among the  $M_v$ s using Eqs. (1-5);
15:                                     ▷ not enough resources at  $h$ 
16:    else
17:       $h' \leftarrow$  find optimum location to host  $c$  using Eqs. (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  $\delta$  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  $\delta$  controllers to turn off;
28:     Re-route and load balance among the  $M_v$  using Eqs. (1-5);
29:     Turn off  $\delta$  controllers;
30:   end if
31: end for
32: if  $e$  is expired then
33:   go to line 2;
34: end if

```

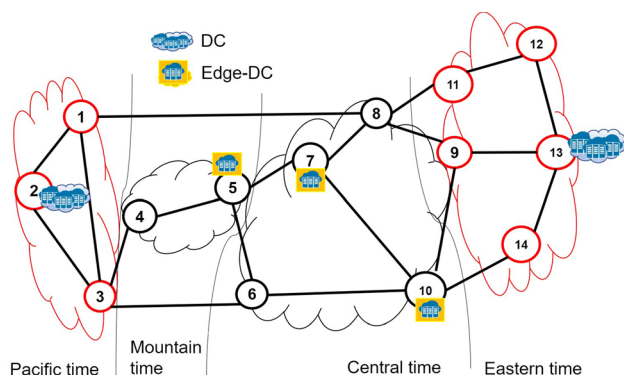


Fig. 3 Example optical network topology with controller host locations (DCs and Edge-DCs) and heterogeneous domains (color red and black denote separate controller types)

5 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^c) is 64 units and total memory capacity (ω_v^c) 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 $\alpha = 15$ ms, $T_v^\mu = 2500$ requests per second [11], per-controller instance compute requirement (T_v^x) = 2 compute units [8], memory requirement (T_v^ω) = 4 GB, $\gamma_h = \$0.01$ per controller per hour, $\gamma_s = \$0.02$ per controller per hour and $\gamma_q = \$0.05$ for every 10 minutes of degraded service [16].

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

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 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

algorithm in [16]). 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 Sect. 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 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

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

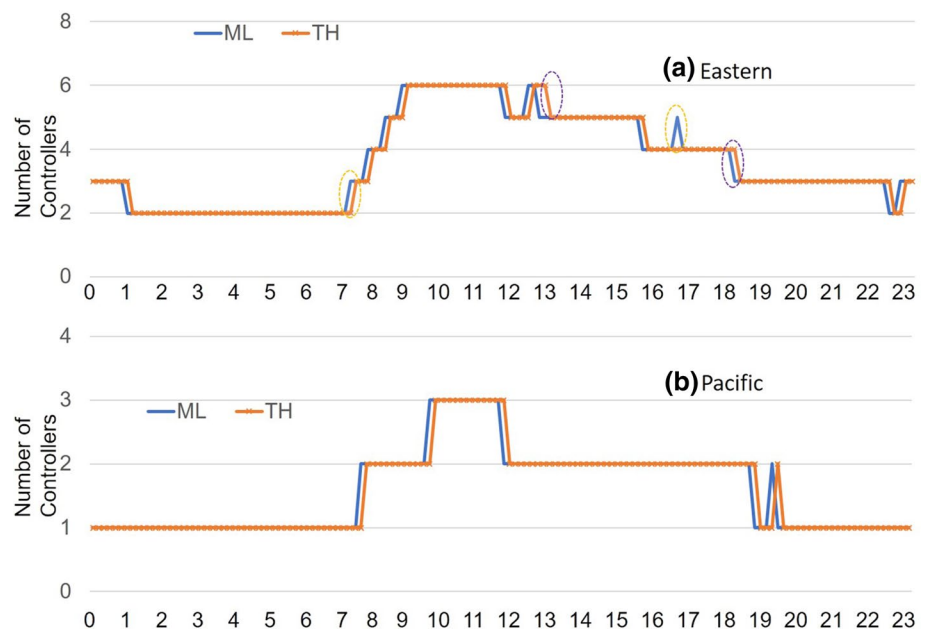
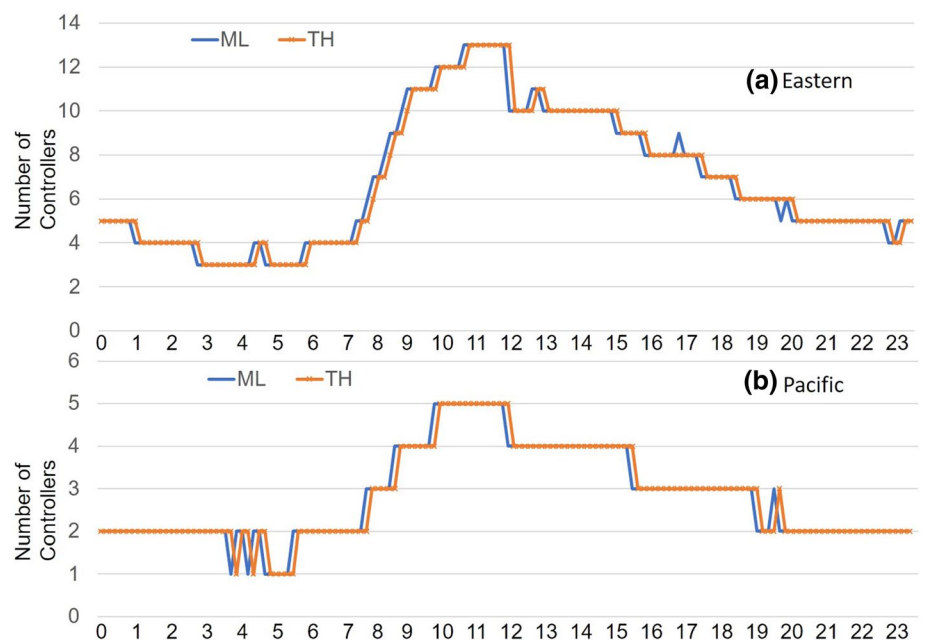


Fig. 5 Illustrative 24-hour variation of number of controllers using machine learning (ML) and threshold-based (TH) approaches with 2% control traffic



area consistently stays over 99%, showing the robustness of the proposed model.

Figure 4a 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 [8]. We observe that ML method is faster to capture the future changes 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). Figure 4b 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.

In Fig. 5a, 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. 5b, Pacific domain experiences more traffic compared to Fig. 4b.

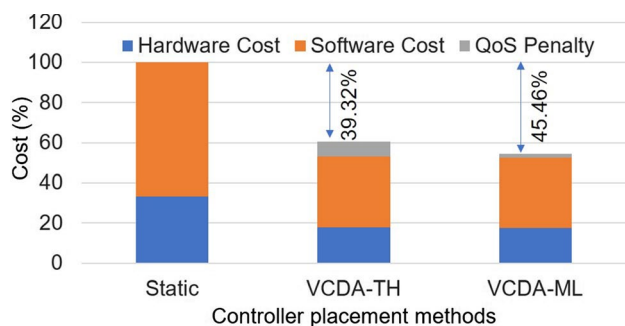


Fig. 6 Cost (%) from VCDA with threshold-based and ML-assisted methods, compared to the static controller placement

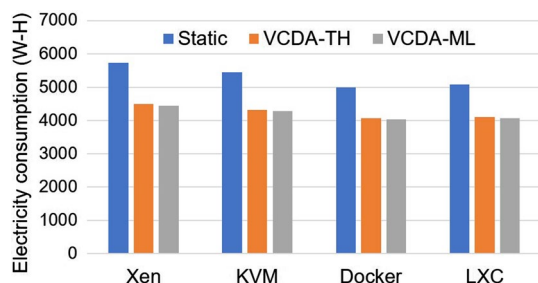


Fig. 7 Electricity consumption saving (Watt-Hour) from VCDA with threshold-based and ML-assisted methods, compared to the static controller placement

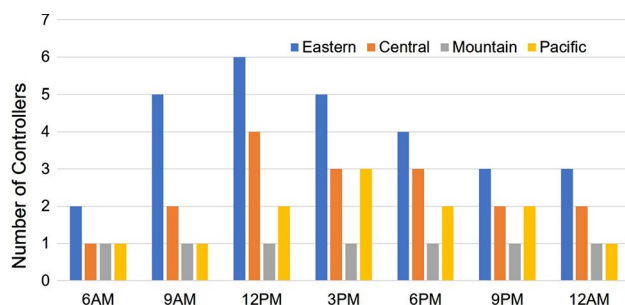


Fig. 8 ‘Center of gravity’ for controllers shifting as the day progresses (1% control traffic)

5.2 Impact on cost savings, electricity consumption, and QoS

Figure 6 explores the sum of hardware cost and software leasing cost (using Eq. 6) and QoS penalty cost (using Eq. 7) for ML and TH approaches. We compare three methods: (a) Static controller placement (realizing prior study [7] in transport network scenario), (b) VCDA-TH (mimicking prior study [8] 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.

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 electricity consumption model for servers from Ref. [18]. We also consider four different virtualization technologies (Xen, KVM, Docker, and LXC) and use practical power consumption values reported in Ref. [19]. 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.

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. Figure 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 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,

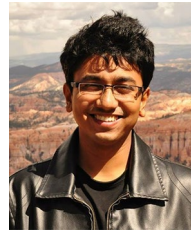
variation of Edge-DC and DC locations, different prediction methods, and more detailed cost models.

Acknowledgements This work was supported by NSF Grant No. 1716945.

References

1. Wang, G., et al.: The controller placement problem in software defined networking: a survey. *IEEE Netw.* **31**(5), 21–27 (2017)
2. Alvizu, R., et al.: Comprehensive survey on T-SDN: software-defined networking for transport networks. *IEEE Commun. Surv. Tutor.* **19**(4), 2232–2283 (2017)
3. Lopez, V., et al.: Control plane architectures for elastic optical networks. *J. Opt. Commun. Netw.* **10**(2), 241–249 (2018)
4. Muñoz, R., et al.: Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks. *J. Opt. Commun. Netw.* **7**(11), 62–70 (2015)
5. Rahman, S., et al.: Dynamic workload migration over optical backbone network to minimize data center electricity cost. *IEEE Trans. Green Commun. Netw.* **2**(2), 570–579 (2017)
6. Heller, B. et al.: The controller placement problem. In: *Proc. 1st Wksp. Hot topics in software defined networks*, pp. 7–12 (2012)
7. Yao, G., et al.: On the capacitated controller placement problem in software defined networks. *IEEE Commun. Lett.* **18**(8), 1339–1342 (2014)
8. Kim, W., et al.: T-DCORAL: a threshold-based dynamic controller resource allocation for elastic control plane in software-defined data center networks. *IEEE Commun. Lett.* **23**(2), 198–201 (2018)
9. Potluri, A., et al.: An efficient DHT-based elastic SDN controller. In: *Proceedings of 9th International Conference on Communication Systems and Networks*, pp. 267–273 (2017)
10. Rahman, S., et al.: Virtualized controller placement for multi-domain optical transport networks. In: *ONDM* (2019)
11. Sallahi, A., et al.: Optimal model for the controller placement problem in software defined networks. *IEEE Commun. Lett.* **19**(1), 30–33 (2015)
12. Savas, S.S., et al.: Disaster-resilient control plane design and mapping in software-defined networks. In: *Proceedings of the 16th IEEE International Conference on High Performance Switching and Routing* (2015)
13. Aguado, A., et al.: ABNO: a feasible SDN approach for multi-vendor IP and optical networks. *IEEE/OSA J. Opt. Commun. Netw.* **7**(2), A356–A362 (2015)
14. Lourenco, R.B., et al.: Robust hierarchical control plane for transport software-defined networks. *Opt. Switch. Netw.* **30**, 10–22 (2018)
15. Oliveira, T.P., et al.: Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *Int. J. Big Data Intell.* **3**(1), 28–37 (2016)
16. Rahman, S., et al.: Auto-scaling VNFs using machine learning to improve QoS and reduce cost. In: *Proceedings of the IEEE International Conference on Communication* (2018)
17. Bhamare, D., et al.: Models and algorithms for centralized control planes to optimize control traffic overhead. *Comput. Commun.* **70**(1), 68–78 (2015)
18. Rahman, S., et al.: Dynamic workload migration over optical backbone network to minimize data center electricity cost. *IEEE Trans. Green Commun. Netw.* **2**(2), 570–579 (2018)
19. Morabito, R.: Power consumption of virtualization technologies: an empirical investigation. In: *8th IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 522–527 (2015)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Sabidur Rahman is currently a Ph.D. student in Computer Science at University of California, Davis. He received his B.S. and M.S. in Computer Science from Bangladesh University of Engineering and Technology (2011) and University of Texas at San Antonio (2014), respectively. He also has research and development experience with AT&T Labs and Samsung R&D. His research interests include computer network virtualization, use of machine learning and data analytics to solve network research problems, cost-effective networking, etc.



Tanjila Ahmed is currently pursuing her Ph.D. degree in Electrical and Computer Engineering from the University of California, Davis. She received her B.S. in Electrical Electronics and Communication Engineering from Military Institute of Science and Technology, Dhaka, Bangladesh in 2010 and her M.S. in Electrical Engineering from University of Texas at San Antonio in 2015. She worked as a research intern at AT&T

labs and software engineer intern at Honeywell Inc in their respective Machine Learning Development team. Her research interests include datacenter networks, optical backbone network, and Internet-Of-Things (IoT) networks.



Sifat Ferdousi is currently a postdoc researcher in the Department of Computer Science at University of California, Davis. She received her B.S. in Electronics and Telecommunication Engineering from North South University, Dhaka, Bangladesh, in 2007 and her M.S. and Ph.D. in Electrical and Computer Engineering from University of California, Davis in 2011 and 2017, respectively. His research interests include

optical network, disaster resiliency, and cloud network.



Partha Bhaumik received his Ph.D. degree in computer science from the University of California at Davis in 2016, his M.S. in computer science from UC Davis in 2012, and his B.E. in computer science and engineering from Jadavpur University, Kolkata, India in 2009. Currently, he is a Senior R&D Engineer at Ennetix Inc. Prior to his M.S., he worked as an assistant system engineer at Tata Consultancy Services Ltd. His research interests include AI and ML powered analytics technologies for next-gen DevOps and AIOps, real-time streaming architectures, cloud-native applications, and software-defined networking (SDN).



Pulak Chowdhury is an author of more than 35 peer-reviewed scholarly papers in international journal and conference proceedings, mostly in resource-efficient, next-generation networking domain. He was an associate editor of IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on Emerging Technologies in Communications. He has also served as a Technical Program Committee (TPC) member of international

conferences and workshops. Dr. Chowdhury received the Ph.D. degree from University of California, Davis, the M.A.Sc. degree from McMaster University, Canada, and the B.Sc. Engineering degree from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2011, 2005, and 2002, respectively. His research interests cover a variety of topics in optical networks, hybrid wireless optical networks, and energy efficiency in next-generation networks. He is currently with Ennetix inc.



Massimo Tornatore is currently an Associate Professor in the Department of Electronics, Information and Bioengineering at Politecnico di Milano, Italy, where he received a PhD degree in Information Engineering in 2006. He also holds an appointment as adjunct associate professor in the Department of Computer Science at the University of California, Davis, where he served as a postdoc researcher between 2007 and 2009.

He is author of about 200 peer-reviewed conference and journal papers and his research interests include performance evaluation, optimization and design of communication networks (with an emphasis on the application of optical networking technologies), cloud computing, and energy-efficient networking. He is member of the editorial board of Springer journal “Photonic Network Communications”. He is active member of the Technical Program Committee of various networking conferences such as INFOCOM, OFC, ICC, Globecom. He was a co-recipient of six best-paper awards from IEEE conferences.



Goutam Das has obtained his M.Tech. degree from the Indian Institute of Technology, Kharagpur (India) in 2001, and his Ph.D. degree from the University of Melbourne (Australia) in 2008. He has also worked as a senior research engineer in General Electric R&D center for 3.5 years from the year 2001–2004. He has also worked as a post-doctoral fellow at Ghent University (Belgium) from 2009 to 2011. Currently, he is

working as an Associate Professor in the Indian Institute of Technology, Kharagpur. His research interest is in the area of both optical as well as wireless networking.



Biswanath Mukherjee is Distinguished Professor at University of California, Davis, where he was Chairman of Computer Science during 1997–2000. He received the BTech degree from Indian Institute of Technology, Kharagpur (1980) and PhD from University of Washington, Seattle (1987). He was General Co-Chair of the IEEE/OSA Optical Fiber Communications (OFC) Conference 2011, Technical Program Co-Chair

of OFC’2009, and Technical Program Chair of the IEEE INFOCOM’96 conference. He is Editor of Springer’s Optical Networks Book Series. He has served on eight journal editorial boards, most notably IEEE/ACM Transactions on Networking and IEEE Network. In addition, he has Guest Edited Special Issues of Proceedings of the IEEE, IEEE/OSA Journal of Lightwave Technology, IEEE Journal on Selected Areas in Communications, and IEEE Communications. He has supervised 77 PhDs to completion and currently mentors 6 advisees, mainly PhD students. He is winner of the 2004 Distinguished Graduate Mentoring Award and the 2009 College of Engineering Outstanding Senior Faculty Award at UC Davis. He is co-winner of Optical Networking Symposium Best Paper Awards at IEEE Globecom 2007, 2008, and 2019. He is author of the graduate-level textbook Optical WDM Networks (Springer, January 2006). He served a 5-year term on Board of Directors of IPLocks, a Silicon Valley startup company (acquired by Fortinet). He has served on Technical Advisory Board of several startup companies, including Teknovus (acquired by Broadcom). He is an IEEE Fellow.