Virtualized Controller Placement for Multi-Domain Optical Transport Networks

Sabidur Rahman*, Tanjila Ahmed*, Sifat Ferdousi*, Partha Bhaumik*, Pulak Chowdhury*,
Massimo Tornatore*†, Goutam Das‡, and Biswanath Mukherjee*
*University of California, Davis, USA †Politecnico di Milano, Italy ‡Indian Institute of Technology, Kharagpur, India
Email: {krahman, tanahmed, sferdousi, pbhaumik, pchowdhury, bmukherjee}@ucdavis.edu; gdas@gssst.iitkgp.ac.in

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 their 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, latency requirements, and costs. Simulation studies considering practical scenarios show significant cost savings and delay reductions compared to standard placement approaches.**

Index Terms—**SDN controller; optical controller; optical transport network; cost savings; network function virtualization; edge computing.**

I. 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 layer 0 (optical) and layer 1 (SONET/ SDH and OTN). T-SDN supports multi-layer, multi-vendor, circuit-oriented networks that are different from packet-based SDN-controlled networks [2]. The control plane for optical transport networks employs a hierarchical distributed architecture [3] comprising heterogeneous (often vendor-specific) Optical Network (ON) Controllers (ONC) and SDN Controllers (SDNC).

Our study considers that T-SDN controllers can be deployed as virtualized controller instances (as in [5]). Virtualized controller placement has many benefits. First, manually deploying SDN and ON controllers 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.) in the 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 [6] 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 [7] [8], and dynamic [9] controller placement problems mostly focused on packet-based SDN controllers and DC networks [10] [11]. But, as we discuss in Section II, 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 ON and SDN controllers, considering the complexity due to virtual instances hosted jointly on DCs and Edge (e.g. NFVi-PoPs), and inter-domain and intra-domain latency constraints. In addition, we observe that [2] [4] both fixed-grid and flex-grid technologies might be required to co-exist (i.e., mixed-grid) with seamless interoperability (see Fig. 1). In this context, our proposed method can support the deployment of different controller types, hence we enforce 'controller-type constraint' (e.g., flex-grid controllers for flex-grid domains, fixed-grid controllers for fixed-grid domains, etc.).

In our previous study [13], we discussed state-of-the-art control-plane architectures for transport network and motivations behind virtualization of controllers. In this study, we explore the technical details of the dynamic controller placement problem (e.g., latency requirements, resource limitations at Edge-DCs, heterogeneous types of controllers, controller capacity limitations, etc.), propose the Virtualized Controller Deployment Algorithm (VCDA), and report illustrative results comparing with prior studies.

This study is organized as follows. Section II reviews prior work on controller placement problems in both SDN and transport networks. Section III discusses the control-plane architecture and describes how virtualization enables to exploit spatial variations of load. Section IV provides a formal problem statement and describes the proposed solution method. Section V discussed numerical results on cost savings and delay minimization. Section VI concludes the study.

II. Background and Related Work

In the context of SDN, both static [7] [8], and dynamic [9] placement problems have been explored. But most studies on SDN Controller Placement Problems (CPPs), e.g., [7]-[9], 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.

On the other hand, control-plane architectures [12] [16] in T-SDN paradigm are designed to accommodate multiple heterogeneous network domains and associated domain-specific ON controllers. But, as recent studies [2] [16] suggests, there is still no consensus on the design of the T-SDN control plane. Ref. [5] was among the first to propose a virtualized control plane architecture for transport networks, but it did not cover several virtualization aspects such as compute and memory resource allocation at host locations, VM consolidation and migration, limitation in VM hosting capacity, etc.

Recent studies [10] [11] on Elastic Control Place (ECP) for SDN controllers discuss threshold-based methods to dynamically resize the ECPs. Ref. [10] focuses on DC networks managed through homogeneous SDN controllers, minimizing control-plane resizing delay. Refs. [10] [11] 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, which introduces new constraints such as host location capacity, and inter-domain and intra-domain communication delays (which [10] [11] do not consider).

III. Virtualized Control Plane for Heterogeneous Transport Networks

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.) using 'north-bound interfaces'.
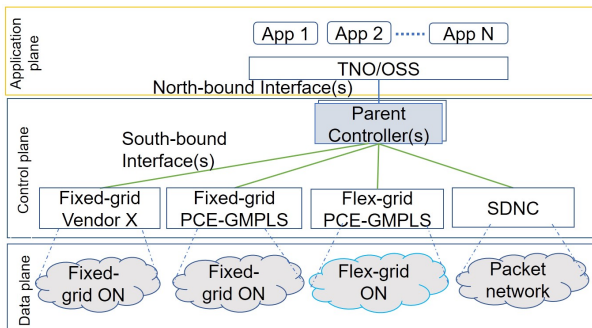

Fig. 1: Control-plane architecture for heterogeneous transport networks.

'Domain Controllers' are responsible for the communication between 'control plane' and 'data plane'. Different autonomous domains, depending on the underlying ONC/SDNC, use specific type of controllers and protocols to control the 'data plane' switches. Fig. 1 shows three types of domains. This example architecture can be extended to support more variations of domain controllers and associated technologies.

IV. Problem Statement and Solution Method

A. Problem Statement

The controller-placement problem is known to be NP-hard [1]. Traditional dynamic controller placement methods [9] [10] focus 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.

In addition to considerations from prior studies, we consider that heterogeneous network domains require specific controller instances, compute and memory resources, and constraints from the Edge-DC vs. DC hosting locations (more details in Section IV.B). In addition, our method uses consolidated placement (placing more interacting modules closer together) from cloud computing. Our proposed method places the controller instance from the same domain together. Also, the 'controller instances' are deployed dynamically to serve the new incoming traffic requests (i.e., higher load requires more controllers and lower load requires less controllers).

The dynamic 'on-demand' controller deployment 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.

B. Input Parameters and Variables

- $G(V, E)$: Optical transport network topology where V is set of network domains and E is set of core network links connecting the domains in V.
- M_v: set of controllers serving domain v.
- S_v: set of switches in domain v.
- T_v: domain-specific controller type.
- 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 compute capacity at v.
- T_v^μ: service capacity limit (i.e., maximum number of 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 arrival of flow routing request. S_r is the switch at which the request has arrived, and M_v gives the set of domain controllers the switch (S_r) is connected to.
- α: latency constraint (maximum allowed delay) from users' Service Level Agreement (SLA).

- C_T: variable containing total cost of running the controllers in all the domains (more details in Section IV.D).

C. 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:

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

where function $D(x,y)$ represents transmission, propagation, and processing delay between points x and y, s is origin, h is controller hosting location, and D_p is processing delay at controller instances M_v (depends on load).

2) **Controller type constraint**: To reflect the impact of controller heterogeneity, we must ensure the following controller type constraints:

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

where, the constraint enforces that all controller instances of v, (M_v^t) match the required controller type T_v.

3) **Controller capacity constraints**: Deployed controllers must have enough capacity to support domain switches:

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

4) **Controller host capacity limit**: Hosting location $(v.h)$ must have both compute and memory capacity to host the controllers placed in that location:

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

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

D. Cost Models

1) Leasing cost for virtual controller instances: We consider that network operators lease hosting capacity from DC operators. Virtual instance leasing cost C_C can be stated as:

$$C_C = \sum_{v \epsilon V} |M_v| * \gamma * d) \quad (6)$$

where γ is per-unit compute per unit-time cost of leasing virtual instances, and d is duration of operation.

2) Network capacity cost: Cost of network usage is often ignored in prior studies focusing on DC SDN scenarios. But the distributed nature of virtualized controller deployment for transport networks can add significant communication cost. First, switch-to-controller communication cost, C_N^S, is:

$$C_N^S = (\sum_{v \epsilon V} (B(S_v, M_v))) * \pi \quad (7)$$

where B(x,y) gives bandwidth consumption due to communication between x and y, and π gives the per-GBps per unit-time bandwidth price.

Similarly, controller-to-controller (same domain) communication cost, C_N^M, is:

$$C_N^M = \sum_{v \epsilon V} (B(M_v(i), M_v(j)); i \neq j)) * \pi \quad (8)$$

where $B(M_v(i), M_v(j)); i \neq j$ gives bandwidth consumption between controllers instances of the same domain.

Similarly, controller-to-controller (different domains) communication cost, C_N^P, is:

$$C_N^P = \sum_{p,q \epsilon V} (B(M_p, M_p; p \neq q)) * \pi \quad (9)$$

In addition, controllers may require to be migrated from one hosting location to other. This live VM migration process adds to the network cost (C_N^V) as follows:

$$C_N^V = \sum_{v \epsilon V} (B_{VM}(v)) * \pi \quad (10)$$

where $B_{VM}(.)$ is the bandwidth consumption due to VM migrations.

Hence, total network cost, C_N, is:

$$C_N = C_N^S + C_N^M + C_N^P + C_N^V \quad (11)$$

3) Delay Cost: We also consider the impact of controller delays on user experience (higher delay means unhappy user, leading to revenue penalty). This cost adds delay factor in decision making and encourages the algorithm to minimize delays. Similar to network cost, cost of switch-to-controller delay (C_U^S), controller-to-controller (same domain) communication delay (C_U^M), and controller-to-controller (different domains) delay (C_U^K) can be calculated by replacing B(x,y) with D(x,y) and π with σ in Eqns. (7), (8), and (9), respectively. Here, D(x,y) gives delay due to communication between sets x and y, and σ is cost (\$) associated to per unit-time delay. Also, our model considers cost due to processing delay at controllers, C_U^P.

Hence, total delay cost becomes:

$$C_U = C_U^S + C_U^M + C_U^K + C_U^P \quad (12)$$

Now, total cost is modeled as:

$$C_T = C_C + C_N + C_U \quad (13)$$

The objective of the proposed method is to minimize the cost of controller deployment:

$$Minimize(C_T) \quad (14)$$

E. 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 make the network unstable, we introduce a decision epoch (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 manages the controller turn-on/off activities, as well as live VM migration in Edge-DC and DC hosting locations.

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:           ▷ Calculate required number of controllers
4:     $c \leftarrow (count(r_v))/T_v^\mu$;
5:     **if** $|M_v| == c$ **then**
6:         Consolidate and load balance switches and traffic
7:         flows among the M_vs using Eqns. (1-5);
8:               ▷ deploy more controllers
9:     **else if** $c > |M_v|$ **then**
10:         $h \leftarrow H_v$;
11:         $\delta \leftarrow c - M_v$;
12:               ▷ enough resources at h
13:         **if** $(\chi_h^t - \chi_h^u) > \delta * T_v^\chi$ & $(\omega_h^t - \omega_h^u) > \delta * T_v^\omega$ **then**
14:             Turn on additional δ controllers (T_v type) at
15:             location h;
16:             Load balance and re-route switches and traffic
17:             flows among the M_vs using Eqns. (1-5);
18:               ▷ not enough resources at h
19:         **else**
20:             $h' \leftarrow$ find optimum location to host c;
21:             using Eqns. (1-5) and Eqns. (6-14);
22:             Allocate c controllers (T_v type) at h'
23:             via **DCM**;
24:             Migrate all M_v instances to h' via **DCM**;
25:             Turn on δ additional controller instances via
26:             **DCM**;
27:             Load balance and re-route switches and traffic
28:             flows among the controller instances;
29:             Turn off M_v controllers at h via **DCM**;
30:         **end if**
31:               ▷ remove extra resources
32:     **else if** $c < |M_v|$ **then**
33:         $\delta \leftarrow c - M_v$;
34:         DCM finds optimum δ controllers to turn off;
35:         Reroute and load balance switches and traffic flows
36:         among the M_v using Eqns. (1-5);
37:         Turn off δ controllers;
38:     **end if**
39: **end for**
40: **if** e is expired **then**
41:     go to line 2;
42: **end if**

Our algorithm ensures that, for each domain, enough controllers are deployed to serve current load, observing the constraints. At a given load, 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 migraion (line 24) to relocate the already-running controller instances to the new location with least interruption of services.

The algorithm turns off the extra controllers (line 32-37) to save operational cost. 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 $O(|V| * max(|M_v|)*max(|S_v|)+|V|*|H_v|*max(|M_v|)*max(|S_v|)$.

V. ILLUSTRATIVE NUMERICAL EXAMPLES

We present illustrative results on a US-wide 14-domain topology (see Fig. 2), 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. For example, we consider six fixed-grid domains (4, 5, 7, 9, 12), two 'packet network' domains (8 and 11), and six flex-grid domains (1, 2, 3, 10, 13, 14). Later, our study also considers larger domains in Fig. 5. We consider two DC locations (2 and 13) and three Edge-DCs (6, 8, 10) to host controller instances.
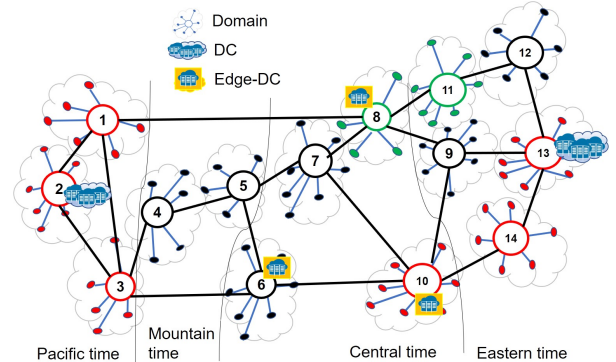

Fig. 2: Topology 1: Example optical network topology with controller host locations and 14 heterogeneous domains (red denotes flex-grid, black denotes fixed-grid, and green denotes packet).

This topology shows a heterogeneous multi-domain optical transport networks with both Edge-DC and DCs. Capacities of DCs, racks, and servers vary significantly in practice. For Edge-DCs (6, 8, 10), we assume total compute capacity (χ_v^t) is 30 units and total memory capacity (ω_v^t) is 60 GB. For DCs, we consider 15000 compute and 30000 memory capacity (to represent virtually infinite capacity).

Per census data, we have 47.6% of US population in Eastern time zone, 29.1% in Central, 6.7% in Mountain, and 16.6% in Pacific time zone. Our study uses this data to generate spatial variation of incoming load $\lambda(v)$. For example, full load ($\lambda = 1$) for the Eastern domains is 30,615 requests per seconds vs. 10,910 requests for Mountain domains. For illustrative examples, let $\alpha = 15$ ms, $T_v^\mu = 2500$ requests per second [9], per-controller instance compute requirement (T_v^χ) = 2 compute units [10], memory requirement (T_v^ω) = 4 GB, γ = \$0.01 per unit per hour, σ = \$0.0001 per minute, and π = \$70 per GBps per month [14].

Fig. 3 compares the normalized cost (Eqn. (14)) among three different methods: (1) DC-Only method mimics controller placement methods which focus on dynamic placement of controllers inside DCs only; (2) Edge-Greedy method considering both DCs and Edge-DCs, but instead of consolidated VM placement considering delays, this method places VMs in a greedy way ignoring network cost (Eqn. 7-11) and delay cost (Eqn. 12) (similar to [9], evolved to host controllers at Edge); and (3) our proposed VCDA method considering consolidated VM placement, delay and capacity constraints (Eqns. (1-5)), and cost minimization (Eqns. (6-14)).
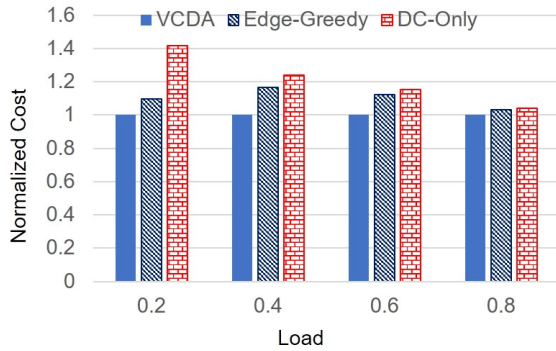

Fig. 3: Normalized cost versus load for Topology 1.

Fig. 3 shows that our algorithm has lowest normalized cost among all three approaches. As expected, placing controllers only at DCs results in very high delays (see Fig. 4), causing higher delay cost. At lower load ($\lambda = 0.2$), Edge-Greedy placement saves cost compared to DC-Only (10% extra cost for Edge-Greedy vs. 41% extra cost for DC-Only), by placing the domain controllers closer to Edge. VCDA benefits more from Edge resources using consolidated placement and reducing inter-controller communication delays, resulting in cost saving. But, at higher load ($\lambda = 0.8$), VCDA saves less due to tighter capacity limits at Edge-DCs, as consolidated controllers need to move to DC locations, resulting in higher network and delay costs. Still, VCDA's cost savings is higher than the other two methods.

A major limitation of virtualized controller placement in transport networks is the additional delays. Edge-DCs helps to reduce those delays. But, as shown in Fig. 4, if the placement method is not aware of the delays (DC-Only) or takes a greedy placement approach (Edge-Greedy), the controllers will experience significant additional delays (resulting in higher

delay cost). At lower load ($\lambda = 0.2$), DC-Only placement method places the controllers far from the domains, resulting in very high delays (60% extra delays). Even Edge-Greedy placement method (13.5% extra delays) reduces significant delays compared to DC-Only. But, as load increases ($\lambda = 0.6$), VCDA experiences higher delays as more controllers are now being placed at DCs (due to Edge-DC hosting capacity limit). Edge-Greedy also experience more delays at higher loads as, in addition to delays due to DC locations usage, more scattered controller instances lead to higher communication delays.
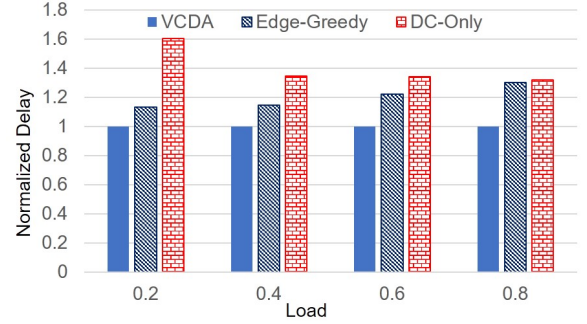

Fig. 4: Normalized delay versus load for Topology 1.

We also observe that changes in topology can impact the results. To illustrate the impact of domain size, let us consider larger domains (see Fig. 5).
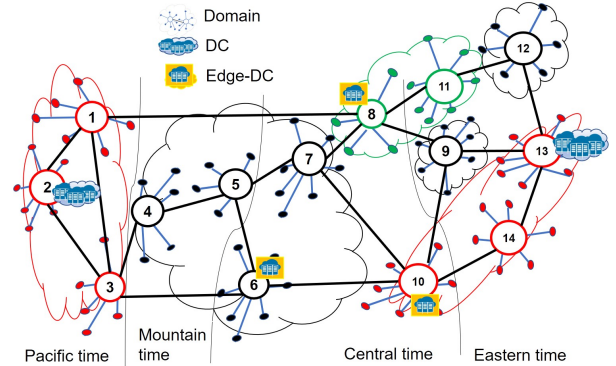

Fig. 5: Topology 2: Example optical network topology with controller host locations and six heterogeneous domains (red denotes flex-grid, black denotes fixed-grid, and green denotes packet).

Fig. 5 show six domains. Nodes 1, 2, and 3 create a flex-grid domain: 1-2-3. Similarly, the other domains are: 4-5-6-7 (fixed-grid), 8-11 (packet), 10-13-14 (flex-grid), 9 (fixed-grid), and 12 (fixed-grid). Fig. 6 shows normalized cost for Topology 2. At lower load ($\lambda = 0.2$), cost savings is 30% compared to 'DC-Only'. VCDA looses some cost savings (compared to 41% for the same load in Fig. 3). This observation can be explained as follows. When node 10 was a separate domain, it's controllers were placed in the Edge-DC at 10. But, with the bigger domain 10-13-14, controllers for this domain are hosted in DC at 13, adding to the switch-to-controller network and delay cost. As load grows cost savings reduces and gets close to 'DC-Only' (0.358% cost savings at load $\lambda = 0.8$). This observation can be explained with the following example: for

domain 4-5-6-7, initially at lower load, controllers were placed at Edge-DC at 6, but as load grew, requiring more resources for controllers, the controller instances were migrated to DC (reducing the cost gap).
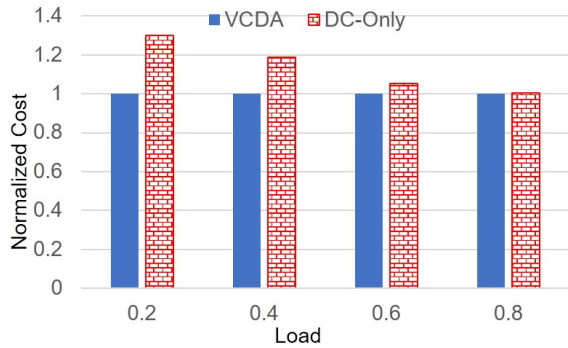

Fig. 6: Normalized cost versus load for Topology 2.

More Edge-DC and DC locations can significantly change the cost and delay. More Edge-DC locations will allow to save more cost and reduce delay even in higher loads ($\lambda = 0.8$). In our study, we consider two DC locations, in two corners of the topology, from these intuitions: (1) DC operators in USA tend to place DCs in more populated east coast and west coast regions; and (2) Placing DCs apart allows us to demonstrate the impact of Edge-DCs. But if we add more DCs or change the DC locations, that will impact the cost and delays as well.

Our study also considers operational cost of controllers. A large portion of DC operational costs comes from electricity cost from servers' electricity consumption [6]. In Fig. 7, we borrow the operational cost model for servers from Ref. [6], and consider four different virtualization technologies (Xen, KVM, Docker, and LXC) [17] to compare the operational cost (in terms of electricity consumption in KWH). We observe that container-based technologies (Docker and LXC) use significantly lower electricity compared to Xen and KVM. Using a more detailed cost model, this electricity consumption can be converted to electricity cost (including power consumption from heating, cooling, ventilation, lighting, maintenance activities, network electricity usage, etc.).
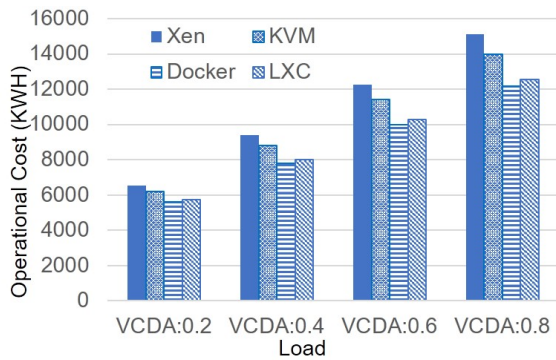

Fig. 7: Operational cost in terms of server electricity consumption over load.

VI. 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 optical controller types, resource limitations at edge-hosting locations, cost from additional delays, etc. Illustrative examples show that our proposed method saves cost and reduces delays significantly, compared to prior studies. Future studies should explore variation of compute/memory requirements, variation of Edge-DC capacities, variation of Edge-DC and DC locations, temporal variation of load, and more detailed cost models.

Acknowledgement

REFERENCES

[1] G. Wang et al., "The controller placement problem in software defined networking: a survey," *IEEE Network*, vol. 31, no. 5, pp. 21-7, 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] T. Ahmed et al., "Dynamic Routing and Spectrum Assignment in Co-Existing Fixed/Flex Grid Optical Networks," *Proc. IEEE Advanced Networks and Telecom Systems*, Indore, India, Dec. 2018.

[5] R. Munoz 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. 60-70, Nov. 2017.

[6] 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.

[7] B. Heller et al., "The Controller Placement Problem," *Proc. 1st Wksp. Hot Topics in Software Defined Networks*, pp. 712, 2012.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[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,* pp. 1-6, 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] R. Morabito, "Power Consumption of Virtualization Technologies: an Empirical Investigation," *8th IEEE/ACM Intl. Conf. on Utility and Cloud Computing*, pp. 522-527, 2015.