Optimizing the Flexibility of SDN Control Plane

Mu He, Mei-Yuan Huang, Wolfgang Kellerer
Chair of Communication Networks, Technical University of Munich
{mu.he,meiyuan.huang,wolfgang.kellerer}@tum.de

***Abstract*—Facing fast-changing network traffic, Software Defined Networking (SDN) endows the capability of efficient traffic forwarding and control plane resource management, which results in flexibility in comparison to networking with legacy rigid hardware. Meanwhile, flexibility has become an implicit target of many novel network algorithms and designs. This paper answers an interesting research question: "Can we optimize the flexibility as an objective?" We study the impact of Data Centers (DCs) location on the flexibility of dynamic control plane. The flexibility is revealed when the control plane can adapt itself with controller migration and switch re-assignment in a timely manner for a new group of flows to satisfy the requirements of flow setup. We propose a model, named FLEXDC, for static DC placement and dynamic controller placement to optimize the control plane's flexibility. We also design heuristics to speed up the decision process. Our simulation over real network topology with synthetic flows shows the improved flexibility of the dynamic control plane. Furthermore, we can save up to 2 DCs while achieving the same flexibility, compared with a naive approach.**

I. INTRODUCTION

In recent years, the concept of softwarization has emerged to provide more flexibility in computer networks [1]. As one realization of this concept, Software Defined Networking (SDN) splits the data plane and control plane of legacy networks to enable fast adaptation in the face of fast-changing networking traffic with the target of shorter forwarding latency and higher throughput [2]. The adaptation comes with two perspectives. *(i)* The forwarding path of traffic can be updated with several flow-mods sent from the controller, compared with the path update of autonomous routing that can take very long to converge. *(ii)* The resource of a software controller (i.e., compute power) can be scaled up or down in accordance with dynamic traffic flows, compared with legacy hardware routers that are hard to scale. With the introduction of the distributed control plane, SDN can be applied to Wide Area Networks (WAN) while still enjoying the benefit of centralized control logic, efficient path computation [3] and high reliability of control [4]. Meanwhile, the distributed control plane has one more degree of freedom for adaptation, i.e., the controllers can be migrated among all possible locations.

Formally, given a changing traffic input, the adaptation of distributed control plane means to place the controllers dynamically so that particular performance metrics such as average flow setup time or load imbalance [5], [6] can always be optimized. The optimization problem is first introduced as the Controller Placement Problem (CPP) by Heller et al. [7]. Existing CPP literature, though, invariably assumes that there is one Data Center (DC) at each node in the topology to host a controller, which is far from realistic, especially for WAN with dozens of nodes. Further, it would be economically irrational to build or rent all DCs, as many of which would seldom be used during the operation.

Moreover, there is no existing metric that evaluates the DC placement as a design choice for distributed control plane. We advocate in our previous work [1], [8], [9] that a measure of flexibility would be beneficial when comparing different design choices of a network. Such a measure is derived from the definition of flexibility as a system's ability to satisfy external changing requirements in a timely and cost-efficient manner. That is, in addition to the ability to dynamically adapt, the adaptation time (i.e., how fast the adaptation can take) and cost (e.g., the induced extra latency and bandwidth) matter when evaluating flexibility. With the flexibility measurement framework [8] in mind, we proceed with *optimizing* the flexibility of our network, which in a nutshell involves deciding the network's design parameters, e.g., the number of DCs and their locations, to deliver the highest possible flexibility value.

We consider in this paper the scenario that network operators have the freedom to decide where to place a limited number of DCs. Once decided, the locations of the DCs are not allowed to change anymore. This problem sounds trivial: a naive approach could place DCs close to large cities where most of the traffic originate and terminate. However, in a multi-controller scenario, inter-domain flows would also trigger flow setup requests at intermediate switches [5] which are not necessarily in the proximity of large cities, not to mention the flows variate from time to time. Fig. 1 illustrates an example of two DCs, one controller, and two time slots with changing traffic. The DC placement supports the control placement in both time slots that can satisfy the flow setup requirements with a timely control plane reconfiguration.

In this paper, we propose a mathematical programming model for static DC placement and dynamic controller placement in SDN networks. Our main objective is to maximize the flexibility of the control plane under the constraints of adaptation time and cost. The control plane adaptation consists of controller migration and switch re-assignment [5]. Implemented as VM migration, the overhead of controller is proportional to the size of the VM and the available bandwidth for migration. The switch-reassignment involves rerouting switch-to-controller messages whose latency overhead depends on the distance between the switch and the controller. Due to the problem's non-linear feature, we apply auxiliary constraints and translation of if-conditional in order to use the linear optimizer. To improve efficiency, we also design two heuristics: one borrows the idea of the legacy

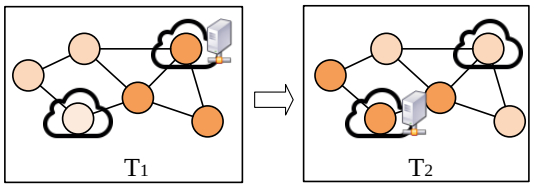

Fig. 1: Illustration of an optimal static DC placement. A single controller can satisfy the flow setup requirement in two time slots. Because two DCs are close to each other, control plane reconfiguration can also happen within a short time. This paper aims at finding the optimal DC placement, as well as the controller placement in each time slot, given varying flows in a network.

controller placement problem, and the other extracts traffic features with less problem input. We evaluate our proposals over real-world topology with synthetic traffic. Evaluation results suggest that the *(i)* flexibility depends on the adaptation time and cost constraints that need to be met, and *(ii)* our proposed heuristics can achieve near-optimal performance in some cases with apparent runtime saving.

This paper is organized as follows. Sec. II describes the literature related to our work. Sec. III introduces the basics of designing a flexible control plane. Sec. IV formulates the mathematical model for optimizing the flexibility of control plane and proposes efficient methods. We evaluate our model and discuss the results in Sec. V, and conclude the paper in Sec. VI.

II. Related Work

In this section, we first discuss the literature that is related to the topic of CPP and flexible network design. Afterward, we clarify the contribution of our work in this paper.

A. CPP in SDN

Upon its introduction by Heller et al. [7], the research of CPP has evolved rapidly and contributed diverse optimization models and solution techniques. Initial literature focuses on static objectives, e.g., average control latency [7], [10] and average inter-controller latency [11], which depend only on the topology and the number of controllers. Dynamic objectives, however, depend on varying networking conditions such as traffic in the network [5], [12], and node/link failures [13]. From the solution's perspective, the existing works are classified into two categories: optimal solution and heuristic algorithms. The first category includes the naive Brute force method [7] and Integer Linear Programming (ILP) or Mixed ILP (MILP) that can be solved by linear solvers. For the second category, state-of-the-art research analyzes the problem's structure and proposes various heuristic algorithms [14], [15]. General meta-heuristic frameworks also show their promising performance in solving CPP problems, giving rise to simulated annealing [16] and evolutionary algorithms [17]. The problem we consider in this paper is a variation of the multi-period CPP, which stems from the multi-period facility location problem in supply chain management [18]. We need to solve different CPP instances over multiple time slots, where the movement of controllers are subject to certain constraints. Online optimization techniques are preferred for such a problem [5], [19]. A comprehensive survey of CPP can be found in the work of Das et al. [2].

B. Flexibility Envisioned by Network Softwarization

Network softwarization can achieve significant cost savings not only because of resource pooling and multi-tenancy, but also due to flexible placement of network functions, direction of traffic flows, and orchestration of network resources [20]. With virtualized network functions in mobile core network [21], some of the functions offered by hardware middleboxes can be moved to DCs and therefore benefit from dynamic resource scaling. Proposals like SIMPLE [22] leverage SDN to flexibly balance the traffic among different middleboxes by the dynamic installation of traffic splitting flow rules in the switches. Further, combining both legacy and SDN data plane for a flexible forwarding model [23] is also advocated due to the performance concern of flow setup in SDN. A detailed discussion of flexibility envisioned by softwarized networks can be found in our survey [1].

C. Our contribution

To the best of our knowledge, this paper is the first one that discovers the design space of the SDN dynamic control plane, which takes a more realistic assumption, i.e., limited DC resources, into consideration. We optimize static DC locations to guarantee the highest flexibility value under adaption time and cost constraints over the whole time horizon. We model the adaptation time as the time needed to migrate the control plane with controller migration and switch re-assignment and the cost as the performance metrics we focus on after migration, including average flow setup time and load of controller.

III. Toward Designing Flexible Control Plane

This section clarifies the definition of flexibility to provide the necessary background knowledge of what do we mean by "optimizing the flexibility". It also elaborates the cost and time aspects, two important factors that we consider in the optimization. Further, it introduces the *measure* and *design* phase that corresponds to controller placement and DC placement respectively.

A. Basics of Network Flexibility

1) Definition: Flexibility has become a significant target of network design in recent years. However, as a high-level term, its interpretation was different across various research fields and use cases. For example, in DC traffic engineering, the authors in [24] consider it as the additional throughput performance for increasing traffic, in comparison to a proportional behavior. In [8], we propose a general measure of flexibility that is abstracted from different use cases. We interpret *flexibility* of a network as its ability to adapt its state to changing requirements promptly and with little effort in order to satisfy specific requests. Formally, we have the following definition: Flexibility is the ratio of the number of requests that can be satisfied (adapted) over the total number of requests, under predefined *time* and *cost* constraint [8].

2) Request: The request is specific to the use case; its definition depends on the functionality of the system and what factors can trigger the system's adaptation. In this paper, we consider the dynamic SDN control plane that is responsible for the flow setup. With varying data plane traffic (defined for each time slot), the adaptation of the control plane can be triggered by the changing pattern of the flow setup requests. Without adaptation, the network (i.e., system) would suffer from *(i)* longer average flow setup time and *(ii)* higher controller load.

3) Time and Cost Constraints: The time constraint reflects how fast the system can adapt. We argue that a system that takes too long to react to new requests would not be deemed as flexible. The adaptation of the control plane includes controller migration and switch re-assignment [5], and both times should be considered. The definition of cost constraint, on the other hand, is broader and use case-specific. It can reflect the induced cost of adaptation, which is the extra mechanism to enable the adaption, the overhead to implement the adaptation, and even a function of the system's performance after and/or before the adaptation [8]. For a flexible system, we would expect both constraints as small as possible. In the optimization model, we pre-define thresholds of both aspects and admit a failure of adaptation when at least one of the thresholds is violated.

B. Design and Operation Phase

Two phases are involved in managing a flexible networking system during its life-cycle: design and operation phase. In the design phase, we decide the parameters of the system, which are hard to change once the system starts to operate. In the operation phase, we optimize the system-specific performance metric at runtime, but under the constraints of system design and operation. In our use case, the design parameters include the number of DCs and their locations. The constraints of the operation phase cover the parameters of DCs, the relation between controllers and DCs, the properties of the control plane, and the time of control plane adaptation. The possible performance metrics are flow setup time, control plane reliability and overhead, etc.

IV. Mathematical Model

Based on the background of the previous section, we propose our optimization model to decide on that DC placement inside the given topology which leads to the maximal flexibility of the dynamic control plane. In other words, with the static DC locations as candidates, the controllers can be placed differently in each time slot among them to better satisfy the requests under time and cost constraints. The number of satisfied requests is defined as the optimization objective that represents flexibility. Our model is an extension of the multi-period facility location problem [18]. Further, we propose three efficient heuristic methods to decide the DC locations.

A. DC Placement Model (FlexDC)

We consider a physical network represented by $G(V, E)$. Every node (i.e., switch) in V has a switch, and is a candidate for DC and controller. Network traffic during a certain time slot, i.e., a *flow profile* F_t, is represented as a set of flows with each flow f defined by source $f[s]$ and destination $f[d]$. There are T time slots, and in each time slot t, the controllers can only be placed on the nodes where DCs locate. The set of possible controller locations is always a subset of the set of possible DC locations, i.e., $D \subseteq C$. Note that for the sake of simplicity, the set of possible DC locations D is the same as the set of nodes V, but in reality, the two sets can be different. We have three parameters to reveal our thresholds of flow setup performance ($\mathcal{N}_{\text{setup}}$), controller's node ($\mathcal{N}_{\text{load}}$), and latency of control plane adaptation ($\mathcal{N}_{\text{time}}$). All the sets and parameters defined in this paper are summarized in Tab. I. The notations and meanings of the variables are described in Tab. II. Except for the variable $\mathcal{P}^d_{\text{dc}}$ that denotes the static locations of the DCs, all the variables are defined per time slot with superscript t. To better explain the model, we split our constraints into different categories: the basic placement model, the fulfillment of adaptation cost and time threshold, and the outcome of adaptation in each time slot.

TABLE I: Sets and Parameters

Notation	Description
V	Set of nodes
E	Set of links with $E \subseteq V \times V$
$G(V, E)$	Network topology with node set V and link set E
D	Set of possible DC locations with $D \subseteq V$
C	Set of possible controller locations with $C \subseteq V$ and $C \subseteq D$
F_t	Flow profile in time slot t with $f[s]$ and $f[d]$ as source node and destination node for $f \in F_t$
p_f	Ordered set of node pairs from source to destination on flow path of $f \in F$ with $p_f \subseteq E$
T	Set of time slots
Π_{ctr}	Number of controllers
Π_{dc}	Number of DCs with $\Pi_{\text{dc}} \leq \Pi_{\text{ctr}}$
$\mathcal{N}_{\text{setup}}$	Threshold of average flow setup time
$\mathcal{N}_{\text{load}}$	Threshold of controller's load
$\mathcal{N}_{\text{time}}$	Threshold of adaptation time

1) Basic Placement Model: This set of constraints constitute the basic model of placing controllers and DCs, as well as assigning switches to the controllers in each time slot. The major difference of this model and the other CPP models is the additional layer of DCs between the nodes and the controllers.

$$\sum_{c \in C} \mathcal{P}^{c,t}_{\text{ctr}} = \Pi_c, \ \forall t \in T; \tag{1}$$

$$\sum_{d \in D} \mathcal{P}^{d}_{\text{dc}} = \Pi_{\text{dc}}; \tag{2}$$

$$\mathcal{P}^{c,t}_{\text{ctr}} \leq \mathcal{P}^{d}_{\text{dc}}, \ \forall c \in C, \ \forall d \in D, \ \forall t \in T; \tag{3}$$

$$\sum_{d \in D} \mathcal{A}^{c,d,t}_{\text{ctr}} = \Pi_c, \ \forall c \in C, \ \forall t \in T; \tag{4}$$

$$\sum_{c \in C} \mathcal{A}^{v,c,t}_{\text{sw}} = 1, \ \forall v \in V, \ \forall t \in T; \tag{5}$$

$$\sum_{v \in V} \mathcal{A}^{v,c,t}_{\text{sw}} \leq |V| \times \mathcal{P}^{c,t}_{\text{ctr}}, \ \forall c \in C, \ \forall t \in T. \tag{6}$$

TABLE II: Variables

Notation	Description
$\mathcal{P}_{\text{dc}}^{d}$	Binary variable representing if a DC is placed on $d \in D$
$\mathcal{P}_{\text{ctr}}^{t}$	Binary variable representing if a controller is placed on $c \in C_t$ in time slot t
$\mathcal{A}_{\text{sw}}^{v,c,t}$	Binary variable representing if a node $v \in V$ is assigned to a controller $c \in C$ in time slot t
$\mathcal{A}_{\text{ctr}}^{c,d,t}$	Binary variable representing if a DC $d \in D$ hosts a controller $c \in C$ in time slot t
$\mathcal{L}^{v,t}$	Non-negative variable representing the control latency of a node $v \in V$
$\mathcal{D}_{\text{sw}}^{u,v,t}$	Binary variable representing if two nodes $u, v \in V$ are assigned to different controllers
$\mathcal{S}_{\text{d}}^{c,u,v,t}$	Binary variable representing if both switches $u, v \in V$ are assigned to controller $c \in C$
$\mathcal{T}_{\text{time}}^{t}$	Non-negative variable representing the overall adaptation time in time slot t
$\mathcal{L}_{\text{dd}}^{u,v,t}$	Non-negative variable representing control latency if a flow is forwarded from u to v
$\mathcal{T}_{\text{ctr\_tran}}^{t}$	Non-negative variable representing the transmission part of controller migration delay in time slot t
$\mathcal{T}_{\text{ctr\_prop}}^{t}$	Non-negative variable representing the propagation part of controller migration delay in time slot t
$\mathcal{T}_{\text{sw\_assign}}^{t}$	Non-negative variable representing the switch re-assignment delay in time slot t
$\mathcal{L}_{\text{sos}}^{c_{\text{old}},c_{\text{new}},t}$	Binary SOS variable representing if a migration happens between two locations c_{old} and c_{new}
$\mathcal{R}_{\text{ctr}}^{c_{\text{old}},c_{\text{new}},t}$	Binary variable representing the location of two controllers in two consecutive time slots
$\mathcal{L}_{\text{prop}}^{c,t}$	Non-negative variable representing the propagation delay of migrating a controller $c \in C$
$\mathcal{R}_{\text{sw}}^{v,c,t}$	Binary variable representing if the assigned controller $c \in C$ of node $v \in V$ in time slot t is different from the controller in time slot $t-1$
$\mathcal{M}_{\text{afst}}^{t}$	Non-negative variable representing the average flow setup time in time slot t
$\mathcal{M}_{\text{acl}}^{t}$	Non-negative variable representing the average weighted control latency in time slot t
$\mathcal{S}_{\text{ctr}}^{c,t}$	Non-negative variable representing the amount of flow setup requests served by a controller c
$\mathcal{B}_{\text{setup}}^{t}$	Binary variable representing if the flow setup performance threshold is reached with $\mathcal{M}_{\text{afst}}^{t} \leq \mathcal{N}_{\text{setup}}$
$\mathcal{B}_{\text{load}}^{t}$	Binary variable representing if the controller's load threshold is reached with $\mathcal{S}_{\text{ctr}}^{c,t} \leq \mathcal{N}_{\text{load}}$
$\mathcal{B}_{\text{time}}^{t}$	Binary variable representing if the control plane adaptation time threshold is reached with $\mathcal{T}_{\text{time}}^{t} \leq \mathcal{N}_{\text{time}}$

Eq. (1) ensures that there are only Π_{ctr} controllers in each time slot t. Eq. (2) defines Π_{dc} as the number of DCs available throughout the whole time horizon T. Eq. (3) ensures that a controller can only be placed on the node that has a DC. Eq. (4) ensures that a DC can only host one controller. A switch is assigned to only one controller in time slot t, which is enforced by Eq. (5) and Eq. (6).

$$\mathcal{L}^{v,t} = \sum_{c \in C} \mathcal{A}_{\text{sw}}^{v,c,t} \times d(v,c), \forall v \in V,\ \forall t \in T; \tag{7}$$

$$\mathcal{S}_{\text{d}}^{c,u,v,t} = \mathcal{A}_{\text{sw}}^{v,c,t} \times \mathcal{A}_{\text{sw}}^{u,c,t},\ \forall u, v \in V,\ \forall t \in T,\ \forall c \in C; \tag{8}$$

$$\mathcal{D}_{\text{sw}}^{u,v,t} = 1 - \sum_{c \in C} \mathcal{S}_{d}^{c,u,v,t},\ \forall u, v \in V,\ \forall t \in T; \tag{9}$$

$$\mathcal{L}_{\text{dd}}^{u,v,t} = \mathcal{L}^{v,t} \times \mathcal{D}_{d}^{u,v,t},\ \forall u, v \in V,\ \forall t \in T; \tag{10}$$

Eq. (7) defines the control latency between a switch and its controller. If the location of the switch and its controller coincide, the control latency is 0; otherwise, the latency defined as the shortest-path latency. If two nodes are adjacent, $d(,)$ returns the latency between them that equals to their geographical distance divided by the speed of light in fiber [5]; otherwise, $d(,)$ returns the shortest-path routing latency between them.

Eq. (8) ensures that two switches are in the same control domain if and only if they are assigned to the same controller, where the binary variable $\mathcal{S}_{\text{d}}^{c,u,v,t}$ represents if two switches u and v are assigned to the same controller c at time slot t. Meanwhile, Eq. (9) defines the binary variable $\mathcal{D}_{\text{sw}}^{u,v,t}$ which represents if two switches u and v are in the different control domains. Eq. (10) defines the induced latency $\mathcal{L}_{\text{dd}}^{u,v,t}$ between a pair of switches if a flow is forwarded from switch u to switch v in time slot t. Note that $\mathcal{L}_{\text{dd}}^{u,v,t} = 0$ if u and v are in the same control domain or there is a controller on v.

2) Flow Setup Performance (Cost Aspect): We consider two cost aspects in this paper, i.e., flow setup performance and load of controller. The flow setup performance is evaluated by the *average flow setup time* $\mathcal{M}_{\text{afst}}^{t}$ which depends on the flow profile F_t and the controller placement [25]. The flow setup time represents the time it takes to forward the first packet of a flow to its destination. It impacts the end-to-end latency experienced by end-users if their applications create mostly short-lived flows. We module it with the following equation:

$$\mathcal{M}_{\text{afst}}^{t} = \frac{1}{|F_t|} \sum_{f \in F_t} \Big[2 \times \mathcal{L}^{f[s],t} + 2 \times \sum_{(u,v) \in p_f} \mathcal{L}_{\text{dd}}^{u,v,t} + d(f[s], f[d]) \Big],\ \forall t \in T \tag{11}$$

For each time slot t, we average the end-to-end flow setup time T_f of all flows in a flow profile F_t with $f[s]$ and $f[d]$ as source and destination node, and p_f is the ordered set of node pairs from source to destination on flow path of f. The exact modeling detail of end-to-end flow setup time can be found in [25], which is omitted here because of the space limitation.

Our previous work [5], [25] demonstrates that optimizing $\mathcal{M}_{\text{afst}}^{t}$ as the objective for a single time slot is much time-consuming, not teqo mention a multi-period model. Eq. (11) can be simplified to only consider the average control latency of each node weighted by the number of new flows originate from it, which is defined in the following constraint:

$$\mathcal{M}_{\text{acl}}^{t} = \frac{1}{|F_t|} \sum_{f \in F_t} \mathcal{L}^{f[s],t},\ \forall t \in T. \tag{12}$$

3) Load of Controller (Cost Aspect): If we model a controller as an $M/M/1$ queue [26], the expected sojourn time of flow setup request from a switch will surge if the arrival rate of setup requests approximates the controller's capacity. A

new flow profile with different traffic distribution can overload a particular controller if the previous controller placement retains and therefore, should trigger the adaptation of the placement.

We consider the load of controller $\mathcal{S}_{\text{ctr}}^{c,t}$ as the second cost aspect. It is defined as the total number of flow setup requests, including both initial and intermediate ones, that the controller needs to process in time slot t.

$$\mathcal{S}_{\text{ctr}}^{c,t} = \sum_{f \in F_t} \mathcal{A}_{\text{sw}}^{f[s],c,t} + \sum_{f \in F_t} \sum_{(u,v) \in p_f} \mathcal{A}_{\text{sw}}^{v,c,t} \times \mathcal{D}_d^{u,v,t}, \quad \forall t \in T,\ \forall c \in C. \tag{13}$$

Similarly, we can also simplify it and only count the load in terms of the initial flow setup requests, as followed,

$$\tilde{\mathcal{S}_{\text{ctr}}}^{c,t} = \sum_{f \in F_t} \mathcal{A}_{\text{sw}}^{f[s],c,t},\ \forall t \in T,\ \forall c \in C. \tag{14}$$

4) Adaptation Time (Time Aspect): Control plane adaptation triggered by new flow profile consists of *(i) controller migration* and *(ii) switch re-assignment* [5], [9]. We assume that switches can only be re-assigned after controllers are migrated to their new locations and have the following equation:

$$\mathcal{T}_{\text{time}}^{t} = \mathcal{T}_{\text{ctr\_prop}}^{t} + \mathcal{T}_{\text{ctr\_tran}}^{t} + \mathcal{T}_{\text{sw\_assign}}^{t}, \tag{15}$$

where $\mathcal{T}_{\text{ctr\_prop}}^{t}$, $\mathcal{T}_{\text{ctr\_tran}}^{t}$, and $\mathcal{T}_{\text{sw\_assign}}^{t}$ denotes the controller propagation, transmission delay, and switch re-assignment delay respectively.

SDN controller is a software (e.g., ONOS [27] and OpenDaylight [28]) running in a VM that can be migrated across different DCs with live-migration techniques [29]. The propagation delay of a migration is the shortest-path latency between its old (at $t-1$) and new (at t) location. Each controller in time slot t selects the closest location of a controller in time slot $t-1$ and calculates its own propagation delay.

To model this selection procedure, we need to find the one-to-one mapping of the controllers of the previous time slot and current time slot. For this purpose, we leverage Spatial Ordered Set (SOS) variables, which are supported by modern linear solvers. In a set of SOS variables, at most *one* item can take a non-zero value, whereas the remaining items must be set to 0. We use binary SOS variable $\mathcal{L}_{\text{sos}}^{c_{\text{old}},c_{\text{new}},t}$ to denote if a migration happens between old and new controller location. For example, $\mathcal{L}_{\text{sos}}^{1,5,3} = 1$ means that the controller on node 5 in time slot 3 is migrated from the controller on node 1 in time slot 2. With the following constraint, we enforce only one migration between a pair of old and new locations:

$$\sum_{c_{\text{old}} \in C} \mathcal{L}_{\text{sos}}^{c_{\text{old}},c_{\text{new}},t} = 1,\ \forall c_{\text{new}} \in C,\ t \in T. \tag{16}$$

As indicated in Eq. (17), the helper variable $\mathcal{R}_{\text{ctr}}^{c_{\text{old}},c_{\text{new}},t}$ will be set to 1 if there is one controller at c_{old} in slot $t-1$ and one controller at c_{new} in slot t.

$$\mathcal{R}_{\text{ctr}}^{c_{\text{old}},c_{\text{new}},t} = \mathcal{P}_{\text{ctr}}^{c_{\text{old}},t-1} \times \mathcal{P}_{\text{ctr}}^{c_{\text{new}},t},\ \forall c_{\text{old}}, c_{\text{new}} \in C,\ t \in T. \tag{17}$$

If there is a controller moving from c_{old} to c_{new}, the propagation delay $\mathcal{L}_{\text{prop}}^{c_{\text{new}},t}$ is the latency between c_{old} and c_{new}, otherwise it is set to a large number $\mathbb{M}$ (meaning an illegal case), as described by the following constraint:

$$\mathcal{L}_{\text{prop}}^{c_{\text{new}},t} = \sum_{c_{\text{old}} \in C} \mathcal{L}_{\text{sos}}^{c_{\text{old}},c_{\text{new}},t} \times \Big[\mathcal{R}_{\text{ctr}}^{c_{\text{old}},c_{\text{new}},t} \times d(c_{\text{old}}, c_{\text{new}}) + (1 - \mathcal{R}_{\text{ctr}}^{c_{\text{old}},c_{\text{new}},t}) \times \mathbb{M}\Big],\ \forall c_{\text{new}} \in C, t \in T \tag{18}$$

Because the migrations of different controllers occur in parallel, only the longest migration matters, whose delay is defined by the following constraint:

$$\mathcal{T}_{\text{ctr\_prop}}^{t} \geq \mathcal{L}_{\text{prop}}^{c,t} \times \mathcal{P}_{\text{ctr}}^{c,t},\ \forall c \in C,\ \forall t \in T. \tag{19}$$

The controller transmission delay $\mathcal{T}_{\text{ctr\_tran}}$ is defined as the time of transmitting the VM of a controller with the migration bandwidth. It is evident that $\mathcal{T}_{\text{ctr\_tran}}$ does not become dependent on the controller placement and is always a constant. Therefore we can ignore it in Eq. (15).

Regarding the switch re-assignment, protocols like [30] need to be designed to reroute all the switch-to-controller messages. We model this time overhead as the propagation delay between the switch and its new controller. Variable $\mathcal{R}_{\text{sw}}^{v,c,t}$ represents if switch v is assigned to a different controller other than c in slot $t-1$ and assigned to c in slot t, as followed,

$$\mathcal{R}_{\text{sw}}^{v,c,t} = \mathcal{A}_{\text{sw}}^{v,c,t} \times (1 - \mathcal{A}_{\text{sw}}^{v,c,t-1}),\ \forall v \in V,\ \forall c \in C,\ \forall t \in T. \tag{20}$$

We also assume parallelism of re-assignment, therefore the overall delay $\mathcal{T}_{\text{sw\_assign}}^{t}$ is defined as the maximum of all induced delays in time slot t.

$$\mathcal{T}_{\text{sw\_assign}}^{t} \geq \sum_{c \in C} \mathcal{R}_{\text{sw}}^{v,c,t} \times d(v,c) \times \mathcal{A}_{\text{sw}}^{v,c,t},\ \forall v \in V,\ \forall t \in T. \tag{21}$$

Note that $\mathcal{T}_{\text{time}}^{t}$ is defined only for valid $t-1$ and t. For the time slot $t = 0$, since $t-1$ does not exist, we have $\mathcal{T}_{\text{ctr\_prop}}^{0} = 0$. But for the switches, we define $T_{\text{sw\_assign}}^{0}$ as the maximum control latency as followed,

$$\mathcal{T}_{\text{sw\_assign}}^{t} \geq \sum_{c \in C} d(v,c) * \mathcal{A}_{\text{sw}}^{v,c,t},\ \forall v \in V,\ t = 0. \tag{22}$$

5) Optimization Objective: As introduced in Sec. III, flexibility is defined as the number of requests that can be successfully adapted under both time and cost constraints. In each time slot t, we define three binary variables $\mathcal{B}_{\text{setup}}^{t}$, $\mathcal{B}_{\text{load}}^{t}$ and $\mathcal{B}_{\text{time}}^{t}$ to denote the fulfillment of two cost and one time constraint respectively. For the cost in terms of flow setup performance, if the average flow setup time $\mathcal{M}_{\text{afst}}^{t}$ (or average weighted control latency $\mathcal{M}_{\text{acl}}^{t}$ for simplicity) is not greater than the predefined threshold $\mathcal{N}_{\text{setup}}$, we set the variable $\mathcal{B}_{\text{setup}}^{t}$ to 1, as defined below:

$$\mathcal{B}_{\text{setup}}^{t} = \begin{cases} 1, & \text{if } \mathcal{M}_{\text{afst}}^{t} \leq \mathcal{N}_{\text{setup}} \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

As if-condition expression cannot be implemented by linear solvers, we translate it into its equivalence as follows,

$$\mathcal{N}_{\text{setup}} - \mathcal{M}_{\text{afst}}^{t} \leq \mathcal{B}_{\text{setup}}^{t} \times \mathbb{M},\ \forall t \in T, \tag{24}$$

$$\mathcal{B}_{\text{setup}}^{t} \times (\mathcal{N}_{\text{setup}} - \mathcal{M}_{\text{afst}}^{t}) \geq 0,\ \forall t \in T, \tag{25}$$

where $\mathbb{M}$ is a very big number.

We apply the same to the constraint of load of controller and adaptation time, and have the following linear equivalences:

$$\mathcal{N}_{\text{time}} - \mathcal{T}^t_{\text{time}} < \mathcal{B}^t_{\text{time}} \times \mathbb{M},\ \forall t \in T, \tag{26}$$

$$\mathcal{B}^t_{\text{time}} \times (\mathcal{N}_{\text{time}} - \mathcal{T}^t_{\text{time}}) \geq 0,\ \forall t \in T, \tag{27}$$

$$\mathcal{N}_{\text{load}} - \mathcal{S}^{c,t}_{\text{ctr}} \leq \mathcal{B}^t_{\text{load}} \times \mathbb{M},\ \forall t \in T, \tag{28}$$

$$\mathcal{B}^t_{\text{load}} \times (\mathcal{N}_{\text{load}} - \mathcal{S}^{c,t}_{\text{ctr}}) \geq 0,\ \forall t \in T. \tag{29}$$

Our single-objective optimization problem formulation is as follows:

$$\mathbb{P}:\ \arg_{\mathcal{P}^d_{\text{dc}}} \max \sum_{t \in T} \mathcal{B}^t_{\text{time}} \times \mathcal{B}^t_{\text{setup}} \times \mathcal{B}^t_{\text{load}},\ \forall t \in T, \tag{30}$$
$$\text{s.t. (1)-(11), (13), (15)-(22), (24)-(29).}$$

The simplified version is as follows:

$$\mathbb{P}':\ \arg_{\mathcal{P}^d_{\text{dc}}} \max \sum_{t \in T} \mathcal{B}^t_{\text{time}} \times \mathcal{B}^t_{\text{setup}} \times \mathcal{B}^t_{\text{load}},\ \forall t \in T, \tag{31}$$
$$\text{s.t. (1)-(10), (12), (14)-(22), (24)-(29).}$$

Given a flow profile F_t of each time slots $t \in T$, our objective is to decide the DC locations which maximize the total number of successful adaptations. Each successful adaptation corresponds to the case when $\mathcal{B}^t_{\text{time}} = \mathcal{B}^t_{\text{setup}} = \mathcal{B}^t_{\text{load}} = 1$.

It's obvious that the formulation of $\mathbb{P}$ includes non-linear constraints such as Eq. (8), Eq. (10) and Eq. (18). In this regard, we apply two linearization techniques [25] and create auxiliary constraints to make them supported by linear solvers. If we have a variable c that is the product of two binary variables a and b, i.e., $a = b \times c$, we could replace it with following three constraints *(i)* $c \leq a$, *(ii)* $c \leq b$, and *(iii)* $c \geq a + b - 1$. Likewise, a variable c that is the product of a binary variable a and non-negative variable b ($b \in [0, \infty)$) could be replaced by four constraints and a big constant number $\mathbb{M}$, i.e., *(i)* $c \leq a \times \mathbb{M}$, *(ii)* $c \leq b$, *(iii)* $c \geq b - (1-a) \times \mathbb{M}$, and *(iv)* $c \geq 0$. The choice of $\mathbb{M}$ dependents on the range of b: for example for Eq.(10), we use the upper bound of $\mathcal{L}^{v,t}$ which is the maximum inter-node forwarding latency in the topology.

B. Heuristic Methods for DC Selection

The optimization's runtime of FLEXDC increases exponentially for large-size topology and more time slots. Therefore, we propose three methods to decide DC locations efficiently with less knowledge of flow profiles.

*1) DC Selector with Random Selection (*RANDOMDC*):* This is a vanilla method that selects DC locations in a purely random manner. We only need the number of DCs and the topology as input. It serves as a baseline to evaluate the performance of other methods.

*2) DC Selector toward Minimum Average Control Latency (*ACLDC*):* Our first heuristic borrows the output of the legacy controller placement problem, optimizing the average control latency of switches in the network. For huge topology size, heuristic algorithms with acceptable optimization gaps can be applied, such as [31].

*3) DC Selector with Half input (*HFDC*):* One factor that contributes to the complexity of the original model $\mathbb{P}$ is the number of time slots. With this method, we optimize $\mathbb{P}$ considering only the first half of the time slots. Note that this number can be amended depending on the traffic input.

C. Measurement of Flexibility

For FLEXDC, the achieved flexibility is indicated by the objective of problem $\mathbb{P}$ divided by the number of adaptations $|T|$. However, the above three methods do not output flexibility directly; thus, we need a process to measure it. The measurement process can be performed by optimizing the problem $\mathbb{P}$ or $\mathbb{P}'$ but with DC placement variable $\mathcal{P}^d_{\text{dc}}$ fixed according to the DC locations. For example, if the three DC locations returned by RANDOMDC are 0, 3, and 8. We need to explicitly set $\mathcal{P}^0_{\text{dc}} = \mathcal{P}^3_{\text{dc}} = \mathcal{P}^8_{\text{dc}} = 1$ and the others to 0 in the linear optimizer. The output is the maximum flexibility value that can be achieved by those DC locations and the controller placement in each time slot.

V. EVALUATION AND DISCUSSION

This section reports the evaluation of the proposed FLEXDC model and efficient methods. It first describes the setup and parameters of our evaluation. Afterward, it compares the performance in terms of both flexibility and runtime, as well as derive take-away messages for SDN network design.

A. Evaluation Setup and Parameters

We evaluate two real-world topologies Abilene and AttMpls from the Topology Zoo [32] for 5 and 10 slots. For each time slot, we create a random number of flows that follow a uniform distribution from 1 to 10 (Abilene) and 1 to 100 (AttMpls). We variate the number of DCs from 2 to 6, fix the number of controllers to be 2, and optimize the problem formulation $\mathbb{P}'$. Regarding the thresholds, we envision different combinations would lead to different flexibility values. Therefore, for $\mathcal{N}_{\text{setup}}$ and $\mathcal{N}_{\text{adapt}}$, we multiply the maximal inter-node forwarding delay with the list of [0.125, 0.15, 0.175, 2.0] and [0.6, 0.7, 0.8, 0.9, 1.0] respectively, whereas for $\mathcal{N}_{\text{load}}$, we use the list of [30, 40, 50, 60, 70].

B. Results

We use two main performance metrics for our evaluation: flexibility and runtime. The flexibility is the ratio of the number of successful adaptations to the total number of new flow profiles as requests. The runtime is the time it takes to output the DC locations, which includes the time of optimizing controller placement and DC locations for FLEXDC. For a better demonstration, we compress most of the results and demonstrate them with 2-D heatmaps: we fix one of the three constraints with a high value and variate the coefficients of the other two (values as indicated above) on the two axes. The brighter the color is, the higher the value would be. Because of the space limit, we only show the cases with $\Pi_{\text{dc}} = 2$, 4, and 6 from left to right.

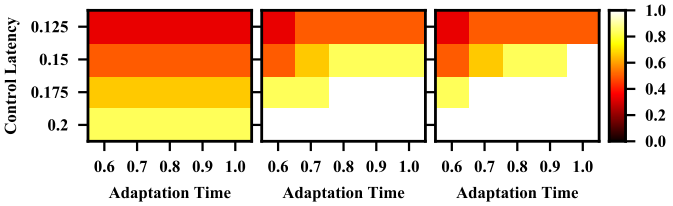

Fig. 2: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (FLEXDC, Abilene, $|T| = 5$). The plot from left to right illustrates the case of $\Pi_{dc} = 2$, 4, and 6 respectively. Same applies to the following figures.

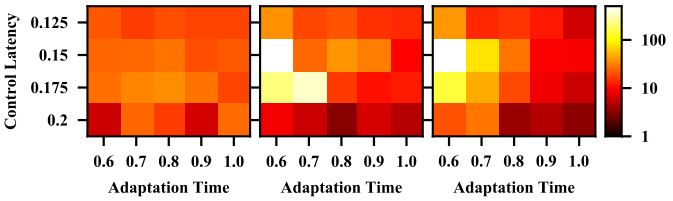

Fig. 3: Heatmap of runtime with different control latency and adaptation time constraint coefficients (FLEXDC, Abilene, $|T| = 5$)

1) FLEXDC*:* Fig. 2 illustrates the impact of adaptation time and average control latency constraints. For 4 and 6 DCs, an increase of the adaptation time constraint coefficient (ATC for short) with fixed weighted control latency constraint coefficient (CLC for short) results in an increase of the flexibility, which is not shown for 2 DCs. This is because 2 DCs would not allow any migration of 2 controllers; thus, higher ATC does not help. However, fixing ATC and increasing CLC leads to higher flexibility for all three cases. Another observation from the figure is that more DCs tends to increase the flexibility for the same coefficient combination. However, when we compare $\Pi_{dc} = 5$ (not shown in the figure) and $\Pi_{dc} = 6$, the increase is not apparent compared with the smaller number of DCs. This complies the effect of *diminishing return* [7], [25]. For most of the coefficient combinations, 6 DCs would be an *over-design* that does not bring much benefit but induces an obvious higher cost of investment.

As for the runtime in Fig. 3 (note the logarithmic scale), the increase of runtime goes hand-in-hand with the increase of Π_{dc} for most coefficient combinations. For fixed Π_{dc}, there is a loose relation between the coefficients and runtime.

Fig. 4 shows the topology of Abilene with node IDs. Fig. 5 illustrates the normalized distribution of DC locations of 20 coefficient combinations (corresponding to 20 grids in the heatmap): the sizes of the circles denote the empirical distribution of the possibility that a node is selected to host a DC. In general, some nodes (e.g., 0 and 3) are more likely to be selected than others (e.g., 5 and 8). When fewer DCs are available, their location distribution is more concentrated on some nodes. The most selected DC locations for $\Pi_{dc} = 2$ are (2, 3) and (0, 4), which are pushed toward two sides in the topology as controller migration is not allowed to happen.

Fig. 6 shows the impact of controller load constraint (short for LC) and ATC, with fixed CLC. Similar to Fig 2, ATC does not impact flexibility for $\Pi_{dc} = 2$. For all cases, increasing LC results in higher flexibility. The observation of runtime (not

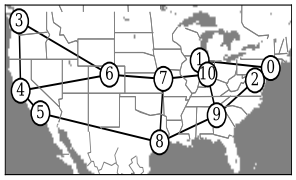

Fig. 4: Topology of Abilene

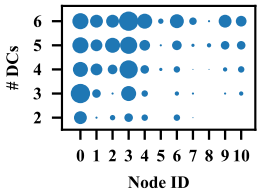

Fig. 5: Summary of DC locations of 20 parameter combinations

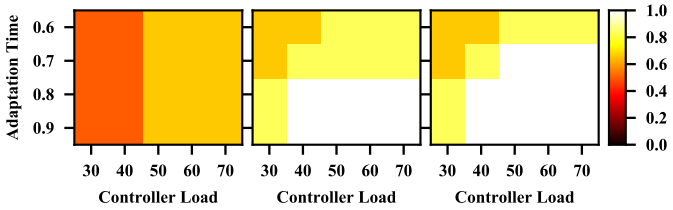

Fig. 6: Heatmap of flexibility with different adaptation time and controller load constraint coefficients (FLEXDC, Abilene, $|T| = 5$)

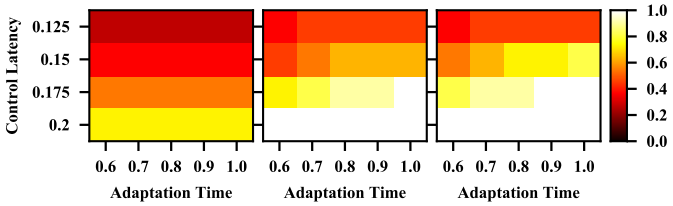

Fig. 7: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (FLEXDC, Abilene, $|T| = 10$)

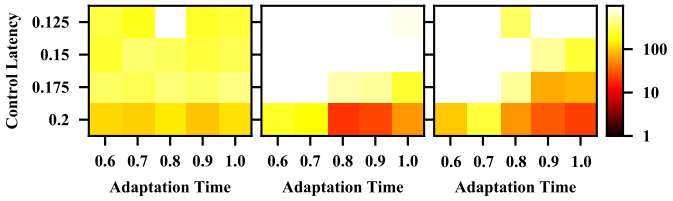

Fig. 8: Heatmap of runtime with different control latency and adaptation time constraint coefficients (FLEXDC, Abilene, $|T| = 10$)

shown) is also similar to Fig. 3.

We report the flexibility and runtime of $|T| = 10$ in Fig. 7 and Fig. 8. The general observation of different coefficient combinations is similar to $|T| = 5$. Longer runtime is intuitive because of a larger input set. Regarding the flexibility, since the flows of different time slots are randomly generated without interdependence, it becomes less likely to find an optimal DC placement to satisfy the constraints. Nevertheless, if the flows follow a diurnal pattern or are predictable, the flexibility would increase for both $|T| = 5$ and $|T| = 10$.

2) RANDOMDC*:* As a vanilla approach, RANDOMDC selects DC locations randomly from the topology with negligible runtime. Because we create new DC locations for each coefficient combination, Fig. 9 looks rather irregular. For several parameter combinations, the flexibility is nearly 0. Only higher Π_{dc} tends to lead to higher flexibility.

3) ACLDC*:* This method takes Π_{dc} as input and optimizes the locations towards the minimum average control latency as if all DCs are home to controllers. The runtime does not depend on the constraint combination, which is in the

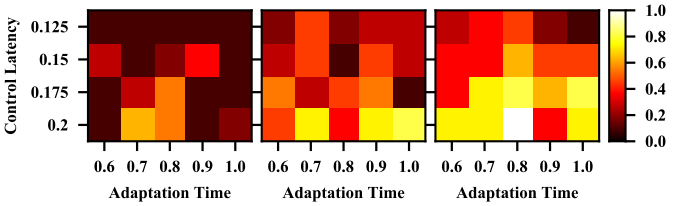

Fig. 9: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (RANDOMDC, Abilene, $|T| = 10$)

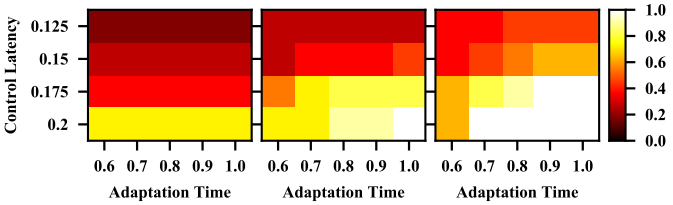

Fig. 10: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (ACLDC, Abilene, $|T| = 10$)

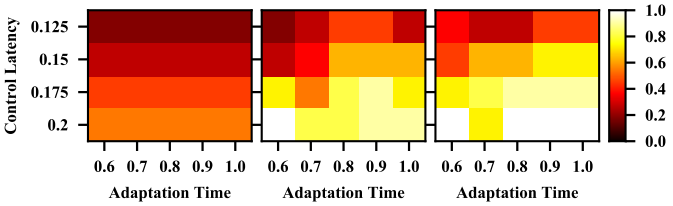

Fig. 11: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (HFDC, Abilene, $|T| = 10$)

magnitude of seconds for both topologies. Fig. 10 depicts the achieved flexibility. Comparing with FLEXDC, the gap is minor for relaxed constraints. In addition, ACLDC can achieve at least the same flexibility for almost all coefficient combinations with 4 DCs (see the heatmap in the middle in Fig. 10) with RANDOMDC with 6DCs (see the heatmap on the right in Fig. 9), which is a significant cost saving.

4) HFDC*:* HFDC only optimizes the first 5 time slots and output the optimal DCs, which are evaluated on all 10 time slots. Its runtime is similar to Fig. 3, which is obviously higher than the other two methods. Fig. 11 illustrates that the flexibility is worse in most cases compared with ACLDC. This is because of the traffic pattern we have mentioned before. We envision that for traffic that is periodic or has a specific trend, HFDC would be a competent candidate to save runtime.

5) Attmpls Topology: Because the runtime of FLEXDC on AttMpls takes more than 10 hours for a single coefficient combination, we prefer using ACLDC and showing its result, as depicted in Fig. 12. We can observe that increasing Π_{dc} and relaxing CLC result in higher flexibility, whereas varying ATC does not have an obvious impact.

C. Key-Takeaways

In summary, the coefficient of constraints plays a crucial role in restricting the maximum achievable flexibility. The runtime of FLEXDC can hinder its usability for large $|T|$ and topology size, giving way to our proposed heuristic methods. The optimal DC locations of FLEXDC depends on the cost factors that we employ. When the constraints are not tight,

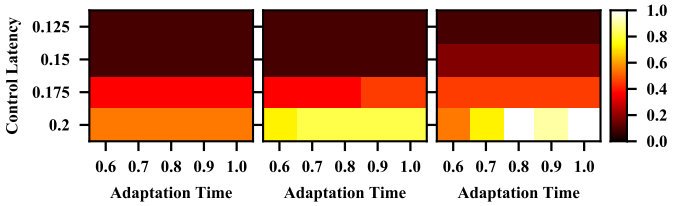

Fig. 12: Heatmap of flexibility with different control latency and adaptation time constraint coefficients (ACLDC, AttMpls, $|T| = 10$)

ACLDC is the right candidate for FLEXDC, whereas when the traffic has a cyclic pattern, HFDC shows its advantage by only taking the flow profiles of the first cycle. Furthermore, both ACLDC and HFDC can achieve the same flexibility but with less number of DCs, comparing with RANDOMDC.

VI. CONCLUSION AND FUTURE WORK

Network softwarization envisions higher flexibility of networks, including flow routing, function orchestration, and resource scaling. A flexible network endows the ability to cover future new demands without the need to jump into a new system, which implies a huge cost. While comparing different system designs, a flexibility measure would benefit us in making a decision, together with other performance metrics. With the knowledge of how a flexible would look, it is more promising to optimize for higher flexibility by a careful selection of the system's parameters.

In this paper, we initiate the study of DC placement towards a flexible control plane in the face of dynamic traffic. We propose a framework for optimizing the flexibility of an SDN network. Under the threshold of adaptation cost and time, the optimization model FLEXDC optimally decides the static DC locations and dynamic controller placement for each time slot. Evaluation results show that the coefficients play a vital role in the flexibility value. Our proposed efficient methods ACLDC and HFDC can solve DC locations with acceptable optimality gap. To further increase the flexibility, one additional factor, besides the system's performance, we need to consider while proposing novel techniques is the time it takes for the system to adapt in the face of changing requests.

For future work, it would be interesting to model other cost factors that are induced by control plane adaptation, such as the signaling overhead. Besides, designing better heuristic algorithms to decide DC locations and even dynamic controller placement is worthy of investigation.

ACKNOWLEDGMENT

This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 647158 - FlexNets). The authors alone are responsible for the content of the paper.

REFERENCES

[1] M. He, A. M. Alba, A. Basta, A. Blenk, and W. Kellerer, "Flexibility in softwarized networks: Classifications and research challenges," *IEEE Communications Surveys & Tutorials*, 2019.

[2] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in sdn," *IEEE Communications Surveys & Tutorials*, 2019.

[3] M. A. Togou, D. A. Chekired, L. Khoukhi, and G.-M. Muntean, "A hierarchical distributed control plane for path computation scalability in large scale software-defined networks," *IEEE Transactions on Network and Service Management*, 2019.

[4] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 31–36.

[5] M. He, A. Varasteh, and W. Kellerer, "Towards a flexible design of sdn dynamic control plane: An online optimization approach," *IEEE Transactions on Network and Service Management*, 2019.

[6] Y. Hu, T. Luo, W. Wang, and C. Deng, "On the load balanced controller placement problem in software defined networks," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2016, pp. 2430–2434.

[7] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[8] M. Klügel, M. He, W. Kellerer, and B. Péter, "A mathematical measure for flexibility in communication networks," in *IFIP Networking*, 2019.

[9] M. He, A. Basta, A. Blenk, and W. Kellerer, "How flexible is dynamic SDN control plane?" in *Proceedings of the IEEE INFOCOM Workshops*. IEEE, 2017, pp. 689–694.

[10] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A k-means-based network partition algorithm for controller placement in software defined network," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[11] T. Zhang, P. Giaccone, A. Bianco, and S. De Domenico, "The role of the inter-controller consensus in the placement of distributed sdn controllers," *Computer Communications*, vol. 113, pp. 1–13, 2017.

[12] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-scale dynamic controller placement," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 63–76, 2017.

[13] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: An enhanced controller placement strategy for improving sdn survivability," in *2014 IEEE Global Communications Conference*. IEEE, 2014, pp. 1909–1915.

[14] M. Obadia, M. Bouet, J.-L. Rougier, and L. Iannone, "A greedy approach for minimizing sdn control overhead," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–5.

[15] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[16] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[17] Q. Zhong, Y. Wang, W. Li, and X. Qiu, "A min-cover based controller placement approach to build reliable control network in sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 481–487.

[18] S. Nickel and F. S. da Gama, "Multi-period facility location," in *Location science*. Springer, 2015, pp. 289–310.

[19] T. Wang, F. Liu, and H. Xu, "An efficient online algorithm for dynamic sdn controller assignment in data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, 2017.

[20] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, "How will nfv/sdn transform service provider opex?" *IEEE Network*, vol. 29, no. 3, pp. 60–67, 2015.

[21] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "A virtual sdn-enabled lte epc architecture: A case study for s-/p-gateways functions," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. IEEE, 2013, pp. 1–7.

[22] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[23] R. D. Vencioneck, G. Vassoler, M. Martinello, M. R. Ribeiro, and C. Marcondes, "Flexforward: Enabling an sdn manageable forwarding engine in open vswitch," in *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, 2014, pp. 296–299.

[24] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 281–294.

[25] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.

[26] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, "Performance evaluation of openflow-based software-defined networks based on queueing model," *Computer Networks*, vol. 102, pp. 172–185, 2016.

[27] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[28] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 2014, pp. 1–6.

[29] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[30] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. R. Kompella, "Elasticon; an elastic distributed sdn controller," in *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2014, pp. 17–27.

[31] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, 2017.

[32] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.